Možnosti widgetů v Elementor Pro a tvorba pluginů v redakčním systému Wordpress

Bc. Jakub Josef Forman

Diplomová práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jakub Josef Forman**
Osobní číslo:	**A19695**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Forma studia:	**Prezenční**
Téma práce:	**Možnosti widgetů v Elementor Pro a tvorba pluginů v redakčním systému Wordpress**
Téma práce anglicky:	**Widget Options in Elementor Pro and the Creation of Plugins in Wordpress CMS**

Zásady pro vypracování

1. Prostudujte a stručně představte redakční systém (CMS) Wordpress.
2. Popište hlavní možnosti jeho úprav promocí šablon, pluginů a dalších možností customizace.
3. Seznamte se s pluginy Elementor a Elementor PRO a popište jejich hlavní funkcionalitu.
4. V rámci praktické části popište způsob návrhu webu pomocí CMS Wordpress a pluginu Elementor na vybraný způsob užití.
5. Navrhněte a implementujte vlastní plugin sloužící k úpravě či rozšíření stávajících Elementor widgetů.

Forma zpracování diplomové práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. MCNULTY, Scott. WordPress: efektivní publikování na webu. Brno: Zoner Press, 2009. Encyklopedie webdesignera. ISBN 9788074130427.
2. WILLIAMS, Brad. Professional WordPress : design and development. Third edition. Indianapolis: Wrox a Wiley Brand, 2015. ISBN 978-1-118-98724-7.
3. ŠESTÁKOVÁ, Lucie. WordPress: vlastní web bez programování. Brno: Computer Press, 2013. ISBN 9788025138328.
4. Elementor Developer Resources: Developers [web]. [cit. 2020-12-01]. Dostupné z: https://developers.elementor.com
5. WordPress Developer Resources: The freedom to build. [web]. [cit. 2020-12-01]. Dostupné z: https://developer.wordpress.org
6. The History of WordPress, its Ecosystem and Community [web]. [cit. 2020-12-01]. Dostupné z: https://kinsta.com/learn/wordpress-history/
7. WordPress Files and Directory Structure [web]. [cit. 2020-12-01]. Dostupné z: https://www.interserver.net/tips/kb/wordpress-files-directory-%20structure/
8. WORDPRESS PLUGIN BOILERPLATE GENERATOR [web]. [cit. 2020-12-01]. Dostupné z: https://wppb.me

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- ☐ že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- ☐ že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Jakub Josef Forman v.r.

podpis studenta

ABSTRAKT

Cílem diplomové práce je demonstrovat postupy pro tvorbu grafického rozhraní a šablon v Elementor a Elementor Pro a jejich možné využití při výuce webových stránek. Praktickým výstupem bude plugin sloužící k upravení a rozšíření stávajících Elementor Widgetů a také bude zveřejněn ke stažení na GitHub. Součástí výstupu bude také popis postupů pro vytvoření základních grafických prvků jako je Header, Footer, tělo stránky (šablony) a práce se základními Widgety pluginu Elementor.

Klíčová slova: Elementor, WordPress, redakční systém, UI/UX, web, učitelství

ABSTRACT

The goal of the diploma thesis is to demonstrate the procedures for creating graphical interfaces and templates in Elementor and Elementor Pro and their possible use in web design lessons. The practical output will be a plugin used to modify and extend existing Elementor Widgets and the plugin will be published for download on GitHub. The output will also include a description of the procedures for creating basic graphic elements such as Header, Footer, page body (templates) and working with the basic Widgets of the Elementor plugin.

Keywords: Elementor, WordPress, cms, UI/UX, web, teaching

Touto cestou bych rád poděkoval vedoucímu Ing. Radku Valovi Ph.D., za rady, které mi během vypracování poskytl a vedení této práce. Dále také všem, kteří se na mé práci jakkoliv podíleli včetně rodiny a nejbližších, chci poděkovat za ochotu a pomoc.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Internet a online prostředí se stalo nedílnou součástí našich každodenních životů. Děti a dospívající ve věku 9–17 let na internetu tráví více jak 4 hodiny denně [32], s technologiemi vyrůstají a budou je používat celý svůj život. Propagovat svou značku, práci atp. v online prostředí je jedním ze základních pilířů úspěchu dnešní a nejspíše i budoucí doby. Tvorba webů a internetové prezentace může vyjít na desítky až stovky tisíc v případě malých a středně velkých projektů, a to v začátcích propagace a podnikání většina lidí nemá. Existují různé řešení, které mohou být téměř nebo zcela zdarma a web, portfolio nebo e-shop si může uživatel vytvořit během víkendu. Práce si klade za cíl seznámit čtenáře se systémem WordPress, prací v pluginu Elementor a rozšířením tohoto systému spolu s pluginem Elementor. Dále také rozvržením designu pro dnes standartní počítače, tablety a chytré telefony, možnostmi Widgetů v Elementor Pro při tvorbě webové prezentace a tvorbou jednoduchého rozšiřujícího pluginu pro Elementor.

WordPress je jedním z nejrozšířenějších a nejpoužívanějších redakčních systémů, který každý rok stoupá na popularitě. Elementor je jedním z nejpoužívanějších a nejrozšířenějších nástrojů pro tvorbu obsahu pro WordPress, který aktivně využívá přes 5.000.000 stránek. V diplomové práci je vysvětlena základní práce s Elementor a Elementor Pro, použití jeho Widgetů a rozšíření některých Widgetů vytvořením vlastního pluginu pro WordPress Elementor.

I. TEORETICKÁ ČÁST

1 WORDPRESS

WordPress (WP) je jeden z nejpopulárnějších open source redakčních systémů (CMS) na světě. Na začátku roku 2021 byl tento systém rozšířený na 39.5 % webů z celého internetu. Díky tomu je tento systém nejpoužívanějším CMS s tržním podílem 64.1 % ze všech webů, využívajících některý z CMS systémů. WordPress je dostupný ve 160 jazycích, z čehož je jich pouze 65 kompletních a do českého jazyka je téměř plně přeložen. [1][2]

Obrázek 1: Logo WordPress [33]

WordPress vznikl v roce 2001 jako projekt několika nadšenců a dobrovolníků, kteří chtěli vytvořit dobře strukturovanou, snadno upravitelnou a rozšiřitelnou platformu pro malé weby a osobní blogy. Z tohoto malého projektu postupem času vyrost nejpoužívanější redakční systém vůbec, který dnes využívají i velké firmy, jako například: BBC America, oficiální blog Hvězdných válek, New York Times, Apple či Marks & Spencer. [1]

Tento sytém nabízí možnosti **šablon**, které jednotlivé weby vizuálně odlišují. To znamená, že každý web má stejnou vnitřní strukturu, ale jeho design je tvořený s pomocí rozdílných šablon a existujících funkcí poskytovanými rozhraním WordPressu. Existují buď šablony zdarma, které si lze stáhnout a nastavit anebo pak placené, jejichž cena se pohybuje od několika stovek, až po tisíce korun českých, a nakonec šablony na zakázku, které vytváří nejeden student nebo firmy, u kterých se vývoj pohybuje od desítek tisíc korun nahoru.

Kromě šablon nabízí systém integraci modulů třetích stran, které se nazývají Pluginy. **Pluginy** jsou jednou z nejpodstatnějších a nejpopulárnějších částí celého systému. Jsou jich desetitisíce a lze je chápat jako rozšíření funkcionality webu. Umožňují přidat další

možnosti do jakéhokoliv WordPress webu během několika málo kliknutí. Tento fakt stojí za obrovským vlivem popularity této platformy. WordPress nabízí také možnost **multisite**, což je funkce provozování více stránek na jedné doméně. A v neposlední řade je jeho **jednoduchost používání** dalším klíčovým prvkem, který umožňuje uživatelům ovládat intuitivní administrativní rozhraní, které se neustále inovuje. [1]

WordPress má také určité nevýhody, které se mohou projevit díky jeho struktuře nebo rozšířenosti na internetu.

- Riziko napadení webu – právě díky své popularitě je WordPress snadnějším cílem hackerů, protože jeho jádro je všude stejné a také veřejné. Tomuto riziku lze výrazně zabránit správným zabezpečením a pravidelnými zálohami celého webu. [1]
- Spam – ze stejného důvodu jako v předchozím bodě se často weby WordPressu (i dalších CMS) často setkávají se spamem. Jelikož někteří roboti na internetu zkouší vpisovat do formulářů to, co mají předvolené (reklamy, nepodstatné slovní spojení atd…) je spam velmi častý. Nejčastěji se jedná o spam na Blogových webech, kde jsou povoleny komentáře, e-shopech nebo odesílání kontaktních formulářů. [1]
- Rychlost webové stránky – způsob, jakým byl a je WordPress vytvořený, neumožňuje, aby byl tou nejrychlejší dostupnou platformou. Bohužel tímto nešvarem trpí více či méně všechny redakční systémy. Existuje řada možností, jak WP web zrychlit, například: optimální velikost multimediálních souborů, kešování nebo odebraní nepotřebných Pluginů a Schémat. [1]

Požadavky hostingu

Pro správnou funkci systému WordPressu, musí vybraný webhosting podporovat minimálně:

- PHP verze 7.4 nebo novější,
- MySQL verze 5.6 nebo novější, nebo MariaDB verze 10.1 nebo novější,
- HTTPS

V oficiální dokumentaci je uvedeno, že je doporučeno mít server běžící na Apache nebo Nginx pro nejrobustnější a nejvýkonnější provoz WordPressu, nicméně jakýkoliv jiný server podporující PHP a MySQL, by měl tento systém taktéž zvládat, avšak týmem WordPressu jsou odzkoušeny tyto dva systémy. [3]

1.1 Šablony

Šablony, někdy nazývané motivy, umožňují zobrazení informací na frontendu stránky, tedy koncovému uživateli, který konzumuje obsah. Vše, co tyto šablony umožňují, je jen zobrazení obsahu bez zásahu do hlavních funkcí WordPressu. Každá šablona se skládá z několika PHP souborů, které sestavují HTML za pomocí funkcí nebo tříd poskytnutých vývojářům WordPressem. Šablony jsou vždy doplněné o CSS a JavaScript, a proto lze vytvořit mnoho rozdílně vypadajících šablon, využívající stejný základ WordPressu. [7]

Volitelnou možností každé šablony je soubor **functions.php**, který je uložený ve složce šablony. Tímto souborem autor šablony WordPressu říká, že jeho vzhled nabízí určitou formu rozšíření a WordPress tento soubor bere jako plugin. Pokud tento soubor existuje, je automaticky načten při inicializaci systému – jak administrace (backend), tak frontend části a umožňuje spouštět tzv. **Hooky** pro tento systém. Často se tento soubor využívá pro zařazení kaskádových stylů (CSS) či JavaScriptu do šablony nebo pro funkce, které se vyskytují na více místech v autorově šabloně. [4]

Každá šablona by měla obsahovat takzvanou WP smyčku – WordPress Loop. Za pomocí této smyčky systém rozhoduje, který obsah se vykreslí a na jakém místě uvnitř PHP souboru. Základní smyčka kontroluje, zda pro danou zobrazenou stránku existuje v systému obsah, který může zobrazit a také jestli tento obsah splňuje globální nastavení – počet příspěvků na stránku a jiné nastavení. [5] Proces rozhodování, který obsah se má zobrazit je řízen WordPressem podle dosažené URL adresy. WordPress prohledá strukturu šablony, zda je pro tuto adresu odpovídající soubor šablony. [6] Každá funkce zapsaná do WP smyčky je označována pojmem Template Tag a na stránkách dokumentace WordPressu lze nalézt kompletní dokumentaci těchto tagů i s jejich popisem a funkcionalitou.

Template Tags jsou používány pro dynamické zobrazování obsahu, informací nebo nabídky úprav obsahu. Některé tagy jsou omezené a lze je použít pouze uvnitř smyčky šablony, jiné zase kdekoliv uvnitř šablony, a dokonce i v pluginech. [5] Systém WordPress nabízí velké množství těchto značek, které se rozlišují na dvě základní skupiny:

- Include Tags,
- Conditional Tags.

Include Tags, jak už napovídá jejich název, se využívají pro načítání souborů šablony, které mohou vracet obsah. Příkladem tohoto tagu je **get_header()**, který vrátí obsah souboru představující hlavičku šablony. Podobně je to i s tagem **get_footer()**.

Druhá skupina tagů zvaná Conditional Tags je podmiňovací. Obsahují funkce, které se používají v šablonách či pluginech pro změnu obsahu závislou na podmínkách stránky. Příkladem tohoto tagu je **is_page(ID)**. Tento tag kontroluje, zda je právě zobrazovaný příspěvek s daným ID načítán. Tohoto se často využívá v podmínkách pro zobrazení dynamických částí webu, které se pro každého uživatele nebo stránku mohou zobrazit jinak a můžou vykreslit jiný obsah stránky.

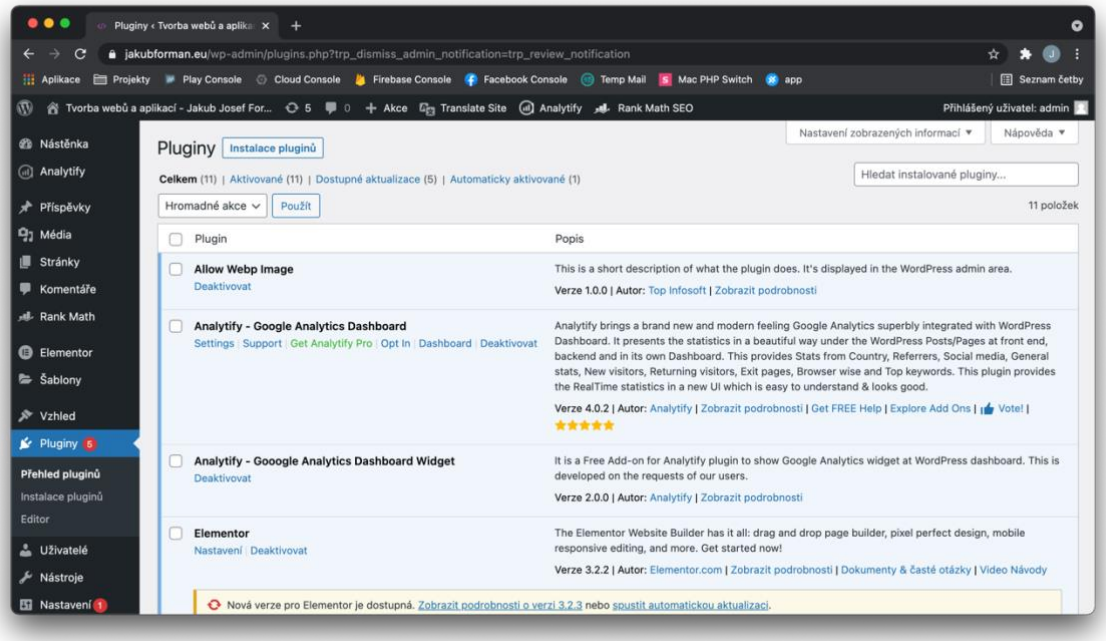
1.1.1 Vlastní CSS

WordPress v posledních verzích přidal editor šablony, ve kterém si lze dopsat vlastní CSS a tím ovlivnit vizuální vjem zobrazované šablony. Tento způsob upravování šablony je vhodný pouze pro drobné úpravy, nikoli úpravu celé šablony nebo její tvorbu. WordPress ukládá toto vlastní CSS do databáze jako příspěvek s typem **custom _css**. Toto CSS je závislé na šabloně, která je aktuálně používána v systému a autor tohoto CSS by si na to měl vždy pamatovat. Pokud by se rozhodl v budoucnu změnit šablonu nebo by aktuálně používaná šablona vydala aktualizaci, která ovlivňuje HTML tagy nebo atributy ID a CLASS z CSS, bylo by možné, že vlastní CSS nebude fungovat správně a může kolidovat s jinou částí obsahu nebo nebude fungovat vůbec. [9]

Jde však o užitečnou formu, jak upravit drobnosti nebo nepřesnosti v šabloně bez fyzických změn v kódu šablony. Toto řešení nikdy neupravuje HTML, pouze přepisuje již existující CSS (pokud existuje).

1.2 Pluginy

Pluginy jsou malé skupiny kódu, které rozšiřují základní funkcionalitu redakčního systému WordPress. Pluginy ve WordPressu jsou psané s využitím *WordPress Plugin API*, které je sestavené na událostmi řízené architektuře. [7] Událostmi řízená architektura – EDA, je architektonický styl, ve kterém je jedna nebo více částí systému vykonána v reakci na obdržení jedné nebo více notifikací o provádění události [8]. Systém WordPress tyto události označuje výrazem **Hook**.

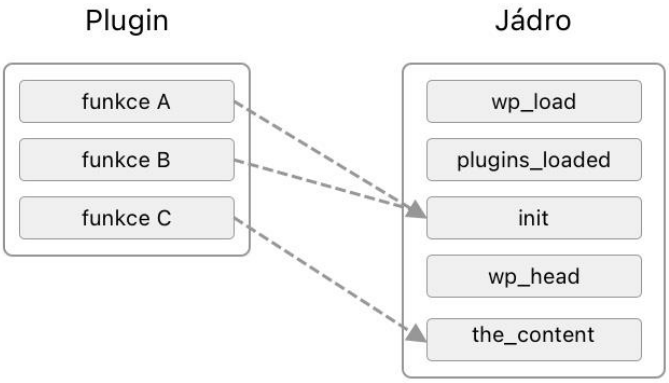


Obrázek 2: Přehled pluginů v administraci WordPress

1.2.1 Hooky

Hooky jsou specifické události jádra systému, ke kterým lze připojit vlastní funkce. umožňují vývojářům pluginů primární metodu interakce s WordPressem, aniž by vývojář upravil kód jádra nebo hledal vlastní řešení. Použitím Hooků lze spouštět definovanou vlastní funkcionalitu v určitý čas. To, že jsou Hooky jen jakýmsi „konektorem“ k jádru aplikace, umožňuje vývojářům a uživatelům WP aktualizovat jak pluginy, tak systém bez ztráty konektivity mezi pluginy a WordPressem. Výjimkou jsou aktualizace, kdy je stará funkcionalita z WP odebrána, nebo plugin vyžaduje novější verzi WP skrze navázání na nové funkce obsažené v určité verzi WP. Tyto vlastnosti většina pluginů kontroluje právě za pomocí Hooků. Před instalací, nebo při načtení základních vlastností WP. [10][11]

Tím, že vývojář může tyto Hooky upravovat, lze docílit potřebné funkcionality a získat tím požadované výstupy z WordPressu. K jednomu Hooku lze přiřadit více PHP metod. Každá tato metoda musí být registrována k Hooku zvlášť.



Obrázek 3: Znázornění práce s Hooky v pluginu

Hooky se podobně jako u šablon Template Tags rozdělují na dva druhy:

- Action – akce,
- Filters – filtry.

Rozdílem mezi akcemi a filtry je styl zpracování Hooku uvnitř WordPressu. U metod přiřazených k Hookům s filtry musí tato metoda přijímat vstupní parametr/y a vracet nějaký výstup. Takto označené Hooky jsou určeny k modifikování nebo filtrování výstupu, který může být zaslán do databáze, nebo je zobrazen uživateli při generaci obsahu. Naopak u metod, které jsou přiřazeny k Hookům s typem akce, nejsou předány žádné vstupní parametry a jakýkoliv výstup z metody je systémem ignorován. [11]

K přiřazení vlastní metody k Hooku slouží dvě vestavěné funkce **add_action** a **add_filter**. Tyto funkce obsahují 4 parametry, z nichž první 2 jsou povinné a druhé 2 jsou nepovinné. První parametr je název Hooku a druhý je název metody, která se má volat z vlastního kódu. Pokud by nastala situace, že je k jednomu Hooku přiřazeno více metod, jádro automaticky přiřadí metodě pořadí právě podle priority, která je této metodě nastavena. Posledním parametrem je počet argumentů, které může volaná metoda přijmout. [10]

WordPress v současné době nabízí přes 2500 Hooků, které může vývojář při tvorbě pluginu nebo šablony využívat. Kromě toho WordPress nabízí i možnost vytváření vlastních Hooků. Ty umožňují dalším vývojářům rozšiřovat a upravovat jiné pluginy nebo šablony, které nejsou obsaženy v jádru systému. Tuto možnost využívá například Elementor, WooCommerce a další známé pluginy. Pro vytváření vlastních Hooků, je nutné myslet na možné konflikty, kterým je třeba se vyhnout. Pokud vznikne duplicitní název,

mohou vznikat chyby, které jsou velmi těžko odhalitelné a následně opravitelné. Seznam existujících Hooků lze nalézt na *https://adambrown.info/p/wp_hooks/hook*. [10][11]

1.2.2 Struktura souborů pluginu

Plugin je v základu tvořený jednou složkou a jedním PHP souborem, které jsou stejně pojmenované nebo pouze jedním souborem. Název souboru pluginu (i adresáře) by měl být vždy unikátní, aby nedocházelo ke konfliktům s jiným pluginem ve stejném adresáři. Nejčastěji je plugin tvořen více jak jedním souborem. Může obsahovat CSS, JavaScript, PHP soubory, konfigurační a jazykové soubory a další. [10]

Každý plugin musí obsahovat speciálně formátovaný **DocBlock** v záhlaví hlavního souboru pluginu. Není třeba, aby byla tato hlavička umístěna ve všech souborech pluginu, stačí v hlavním inicializačním souboru. Pomocí této hlavičky se docílí toho, že WordPress bude k tomuto adresáři nebo souboru přistupovat jako k pluginu. Pokud je plugin tvořen jen jedním souborem bez adresáře, pak se bude chovat stejně, jako by byl uvnitř složky jediný soubor. Hlavička by v sobě měla obsahovat [10]:

- Plugin Name – název pluginu,
- Description – popis pluginu,
- Version – verzi pluginu,
- Author – autora pluginu,
- Author URI – autorův web,
- License – licenci (většinou GPL).

```php
<?php

/**
 * Plugin Name: Section Column Booster
 * Plugin URI: https://jakubforman.eu/personal-projects/section-column-booster/
 * Description: An extension plugin used in sections & columns for Elementor. It's enabling custom width in columns & horizontal align columns in a section.
 * Version: 1.2.1
 * Author: Jakub Josef Forman
 * Author URI: http://jakubforman.eu
 * Domain Path: /languages/
 * License: GPLv2 or later
 * License URI: http://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain: section-column-booster
 * Domain Path: /languages/
 */
```

Obrázek 4: Ukázka DockBlock v praxi

Jelikož je jediným povinným prvkem hlavičky Plugin Name, který nese název pluginu, je doporučeno vyplnit i další prvky hlavičky. Doporučení je zde hlavně z důvodu rozlišení mezi ostatními pluginy v administrační části správce pluginů, kde lze jednotlivá rozšíření přidávat, povolovat nebo odebírat. Pokud budou tyto náležitosti splněny, bude tato

složková struktura nebo soubor zobrazený v přehledu pluginů v administrační části pro jejich správu. Pro to, aby byl plugin zaveden do systému, je nutné jej aktivovat v již zmiňovaném administračním rozhraní. Uložení složky/souboru do adresáře **/wp-content/plugins/** totiž neznamená, že je systém automaticky aktivuje. [10][11]

1.2.3 Bezpečnost pluginů

Stěžejním bodem všech pluginů, vzhledů i WordPresu je obecně bezpečnost. Každý, kdo se podílí na vývoji, by měl na bezpečnost myslet na prvním místě a zajistit vývoj tak, aby zabránil nežádoucím útokům a zneužití systému skrze vyvíjený plugin. V případě, že by se v pluginu či vzhledu objevila „bezpečností díra“, která by mohla ohrozit bezpečnost celéhos systému, mohl by být takový systém velmi zranitelným. WordPres vývojářům nabízí v základu některé zabudované bezpečnostní prvky, které je doporučeno aplikovat. [10]

Jednou cestou, jak řešit bezpečnost, je kontrola uživatelských pravomocí pro provedení požadované akce. Tuto kontrolu lze provádět vždy, když je třeba spustit nějakou akci nebo Hook v pluginu. Plugin před spuštěním akce zkontroluje, zda uživatel, který chce tuto akci použít, má na tuto akci pravomoc a pokud tuto pravomoc nemá, tak akci zamítne.
S oprávněními se lze v systému setkat na několika úrovních. Nejčastěji využívanou je forma rolí, které uvnitř sebe obsahují oprávnění k různým akcím. Tyto role jsou ve WordPressu již předdefinované:

- Administrátor – má přístup ke všem administračním funkcím,
- Šéfredaktor – může spravovat a publikovat veškeré příspěvky,
- Redaktor – může spravovat a publikovat pouze vlastní příspěvky,
- Spolupracovník – může vytvářet a spravovat vlastní příspěvky, bez možnosti publikace,
- Návštěvník – může spravovat pouze svůj vlastní účet.

WordPress nabízí i možnost vytvářet vlastní role s vlastními oprávněními. Pro zvýšení bezpečnosti jádro WordPressu nabízí i možnost kontrolovat tyto role a oprávnění. Jde o metody, které vrací boolean hodnoty toho, zda má uživatel požadovanou roli nebo oprávnění. Role a oprávnění se do těchto metod předávají jako parametry. Taktéž jim lze nastavit výchozí a vlastní role a oprávnění. [10]

WordPress také ve svých interních metodách využívá celou řadu zabezpečení proti CSRF (XSRF) útokům, které může vývojář během vývoje pluginu využívat. Toto je zabezpečeno pomocí WP Nonce – ze slova **N**umber Used **Once**. Tyto Nonce se využívají u odesílání požadavků (jako ukládání nastavení), formulářů a AJAX požadavků, pro ochranu před neautorizovaným přístupem pomocí dříve generovaného náhodného tokenu, který slouží pro jeden request. [12]

Vývojář nesmí zapomenout ani na kontrolu veškerých dat, která jsou uživateli vkládána a ukládaná do databáze. Považuji za důležité zmínit zabezpečení proti SQL Injection a dalším možným napadením a chybám, které by vývojář měl mít vždy na paměti. Kromě vestavěné bezpečnosti formou rolí, oprávnění a dalších interních technik, je k dispozici celá řada pluginů, které zabraňují různým typům útoků.

1.2.4 Doporučení

WordPress je již vyvíjen nějaký čas a za tu dobu se styl psaní kódů vyvinul. Všechny doporučení však radí dodržovat stejné praktiky jako používá WordPress, pro dosažení lepší organizace kódu a zamezení možným kolizím. [13]

Vše, co se ve WordPressu nachází – funkce, proměnné a třídy jsou implicitně definovány v globálním jmenném prostoru zvaném namespace. V praxi to znamená, že je možné jedním pluginem volat metody z jiného pluginu, nebo vyvolat kolizi stejným pojmenováním metod. Nejčastěji nastane jmenná kolize, pokud plugin využívá stejné jméno pro třídu, proměnnou nebo metodu. Existuje však několik technik, pro zamezení výskytu jmenných kolizí. [13]

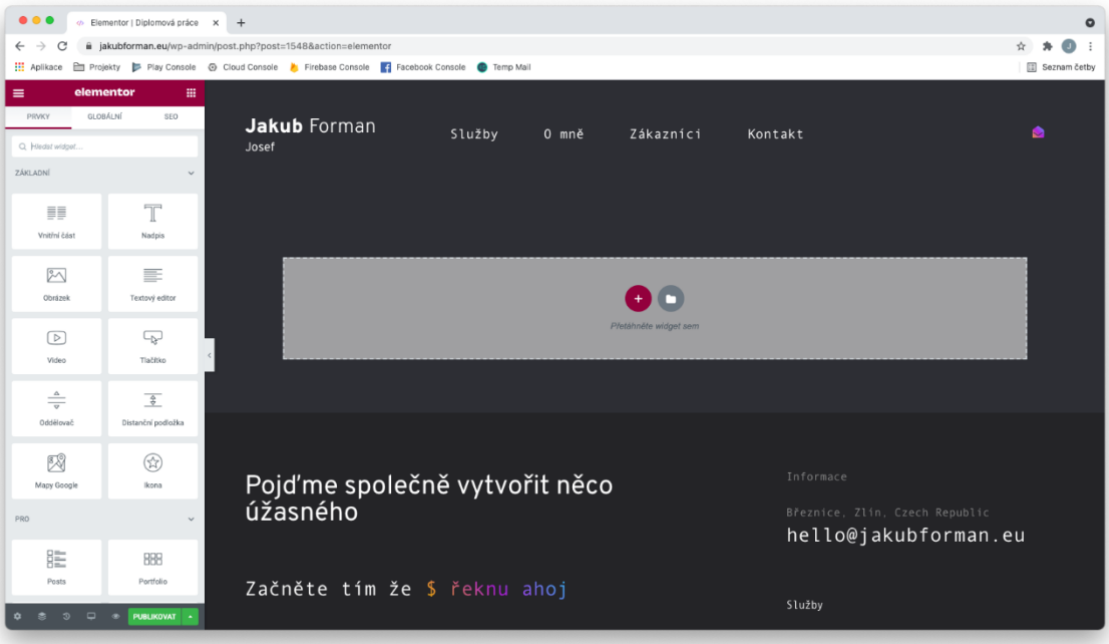
1. WordPress uvádí, že by všechny metody měly být lidsky čitelné a neměly by být pojmenovány nečitelně jako `xxyyghzz2()`. Ovšem v určitých případech lze použít speciální identifikátor jako předponu metody. Identifikátor by měl být opět lidsky čitelný a měl by se využívat všude, kde by mohla nastat jmenná kolize. [13]
2. Používat kontrolu existence již existující proměnné, třídy nebo metody, která dokáže detekovat přítomnost již existující metody a zabránit tak v implicitním názvu. Využívají se funkce `isset()` pro kontrolu proměnných, `function_exists()` a `class_exists()` pro kontrolu existence metod a tříd. [13]

Mezi hlavní doporučení patří i složková struktura pluginu, která by měla být tak, jak uvádí dokumentace WordPressu. Pokud by tak nebyla, může nastat situace, kdy se nepovede

vyvíjený plugin publikovat do centrální knihovny pluginů WordPress nebo může plugin značně zpomalit celkovou dobu načítání webu.

2 ELEMENTOR

Elementor je jedním z nejpokročilejších „drag and drop“ editorů obsahu pro vytváření webů na redakčním systému WordPress bez nutnosti mít znalosti CSS, HTML, JavaScriptu nebo PHP. Tento editor slouží pro vytváření nejen stránek, ale i archivů, produktů, košíků a další typů příspěvků. Elementor v sobě nabízí kompaktní a elegantní řešení především pro e-shopy, blogy, novinkové portály, školní a další weby, které požadují flexibilně upravovat nebo vytvořit design ve velmi krátkém čase. Základní verze je dostupná zdarma a nabízí řadu možností, jak vytvářet obsah na webu. Elementor je z 90 % přeložen do češtiny a aktuálně je nainstalovaný na více jak 5.000.000 stránkách. [16]



Obrázek 5: První pohled na editor Elementor

Elementor je postaven na tvoření designu z tzv. Widgetů, které se jednoduše přetahují na plátno a vytváří tak obsah a design stránek, navíc řada těchto Widgetů umožňuje dynamické propsání obsahu, což zjednodušuje například připsání názvů stránek, náhledových obrázků nebo vložení vlastních dynamických polí a šablon u archivů. Kromě tvorby stránek za pomocí tzv. Widgetů nabízí další nástroje, které se mohou při vytváření obsahu hodit a jsou dostupné v jeho placené verzi PRO. Mezi tyto nástroje patří [15]:

- Custom Fonts – vlastní fonty (výchozí knihovnou fontů je knihovna Google Fonts),
- Custom Icons – vlastní ikony (výchozí knihovnou je FontAwesome),
- Custome Code – umožňuje vložit vlastní kód HTML, CSS, JavaScript do stránky na určené místo s prioritou podobnou tomu, jak fungují WP Hooks v pluginech. Administrátoři již nemusí vkládat Google Analytics do hlavičky Šablony, která by se mohla v budoucnu přepsat.

Elementor je po instalaci ihned připraven k použití. [16] Existuje ale několik možných komplikací, které mohou bránit v jeho využívání. Pokud se Elementor instalujete, na již existující web, je možné, že se objeví problémy týkající se nastavení. Druhým častým problémem je používaná šablona, která je buď zastaralá nebo neumožňuje využít plný potenciál Elementoru a brání tak v jeho správné funkci.

2.1 Nastavení Elementoru

Elementor má nastavení rozděleno do 2 kategorií. **Nastavení**, které by se dalo označit za základní, se dělí na několik sekcí:

- Obecné – slouží pro nastavení typů příspěvků, kde lze Elementor využívat a zakázání výchozích barev a fontů z výchozí používané šablony. [17]
- Integrace – zde lze nastavit integrace s dalšími SDK a knihovnami třetích stran, které lze za pomocí Elementoru používat (nikoli však jiných pluginů). Dostupnost knihoven je více jak dostatečná a v základu již nabízí integraci s reCAPTCHA, Facebook SDK nebo MailChip a dalšími službami. [15][17]
- Pokročilé – nastavení, jak se bude CSS soubor vygenerovaný Elementorem vkládat do stránky, povolení nebo zakázání nahrávání nebezpečných souborů na web a další rozšířené možnosti pro vývojáře. [17]
- Experimenty – je nově přidaná funkcionalita aktuálně ve verzi Beta, která umožňuje rozšířit Elementor o další funkce jako je ukládání všech dat z Elementor formulářů do interní databáze, aby nedošlo k jejich možné ztrátě, vytváření nových typů Landing Pages, lepší načítání obrázků a souborů a další.

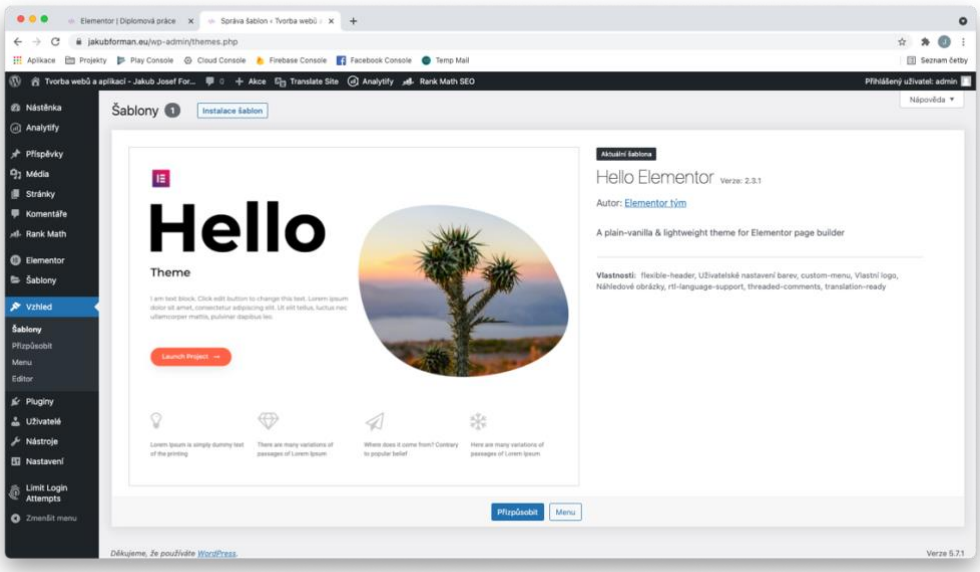
Druhý typ nastavení, který v Elementoru existuje jsou **Nástroje**. Tyto nástroje se využívají většinou při problémech, které občas mohou vzniknout. Umožňují povolit režim ladění nebo nouzový režim, obnovit synchronizaci knihoven a znovu vygenerovat CSS, pokud něco nefunguje tak, jak by mělo. Podstatnou funkcí je nahrazení URL adres v databázi,

které odkazují na nevalidní adresu (třeba bez HTTPS) na novou URL adresu. *Tato funkce je nově přidána také ve WordPress 5.7.1*. Další nastavení je možnost zvolit si libovolnou verzi Elementoru, která se na stránce bude využívat – tato možnost je ale nedoporučována běžným uživatelům, jelikož může změnit již existující obsah. Posledním nastavením je režim údržby, kde lze vybrat libovolnou šablonu, která se zobrazí na všech stránkách, pokud uživatel není přihlášen a umožní nastavit celý web do režimu výstavby. [18]

2.2 Šablony, Vzhled

Elementor zvládá pracovat s téměř všemi šablonami, které se aktuálně pro WordPress používají a nejsou zastaralé. Pomocí této vlastnosti lze Elementor nasadit na každý web i v průběhu jeho existence. Tým Elementoru také připravil čistou šablonu pojmenovanou **Hello Elementor**, která je velmi minimalistická a může posloužit jako základ pro vytváření osobního blogu nebo e-shopu úplně na začátku. Tato šablona je přizpůsobena tak, aby nedocházelo k případům, kdy se každá část musí nastavit v jiné části systému WordPressu nebo se tyto případy minimalizovali. [19]

Také je tato šablona menší než většina vzhledů bez stylů a scriptů. Šablona je vytvořena tak, aby splňovala nejpřísnější požadavky na webové stránky a fungovala se všemi pluginy bez konfliktních situací. Šablona podporuje RTL čtení textu a vícejazyčnost, je plně responzivní a je Open Source. V testech dosáhla nejrychlejšího načtení celé stránky do 503 ms a celková velikost stránky byla pouhých 475.1KB s 37 požadavky, což z ní dělá jednu z nejrychlejších šablon pro WordPress. [19]



Obrázek 6: Doporučená šablona Hello Elementor

2.3 Elementor Šablony

Elementor Šablony jsou určené pro Elementor a nejedná se o šablony jako takové. V českém jazyce je zde zavádějící překlad, který může evokovat výchozí šablony WordPressu, ale jedná o úplně odlišnou skupinu funkcí.

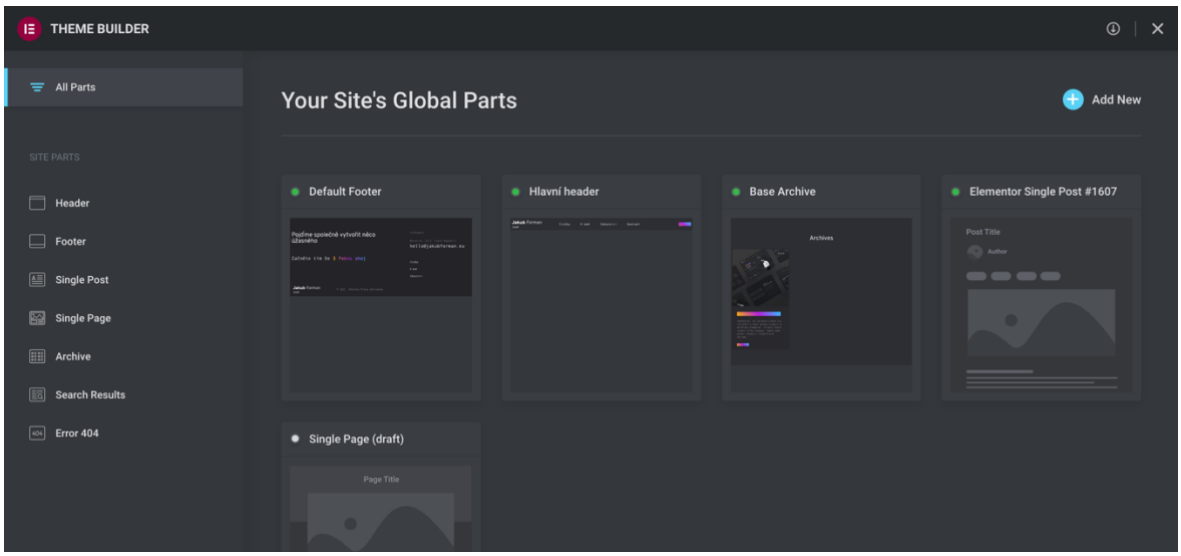
2.3.1 Typy šablon

Elementor nabízí možnost vytvořit několik typů těchto šablon, které lze rozdílně používat, ale také pro orientaci jsou lépe uspořádané skrze jejich kategorii. Tento seznam se může lišit na verzi používaného Elementoru, Elementoru Pro a jiných pluginů jako je třeba WooCommerce. Theme Builderu (Šablony) se v nejnovějších verzích začíná přesouvat přímo do editoru Elementoru, momentálně lze tento Theme Builder zobrazit jak v administraci pod položkou Šablony v postranním menu, tak v editoru Elementor skrze hamburger menu a položku Theme Builder. [20][21] Základní typy šablon jsou:

- Stránka – čistá stránka, která může stát samostatně podobně jako klasické stránky WordPressu.
- Section – značí typ příspěvku jako sekci, slouží pro lepší orientaci při vytváření globálních sekcí, které se využívají na více místech.
- Popup – nabízí možnost zobrazení Modal (vyskakovacího) okna na jakékoliv stránce nebo po určité akci.

- Header / Footer – umožňuje vytvořit specifickou hlavičku nebo zápatí a vložit jej na specifické stránky nebo globálně na všechny stránky.
- Single Post – slouží pro nastavení designu pro každý jeden příspěvek, určitou taxonomii nebo jiné nastavení.
- Single Page – slouží pro nastavení designu pro každou jednu stránku, určitou taxonomii nebo jiné nastavení.
- Archive – zobrazuje archiv, nejnovější příspěvky, blog, kategorie a další taxonomie.
- Search Result – zobrazuje výsledky z vyhledavače, typ je podobný typu Archive.
- Error 404 – umožňuje vytvořit design pro 404 stránku.

Všechny typy příspěvků lze libovolně vytvářet a aplikovat jejich použití podle potřeb. Existuje speciální šablona **Default Kit**, kterou Elementor automaticky vygeneruje i v případě jejího smazání. Tato šablona obsahuje veškeré nastavení vzhledu Elementoru a je pro správnou funci podstatná. [14] Šablony lze navíc exportovat ve formátu .json a přenášet je tak mezi různými weby, které se na Elementoru vytvářejí. [21].



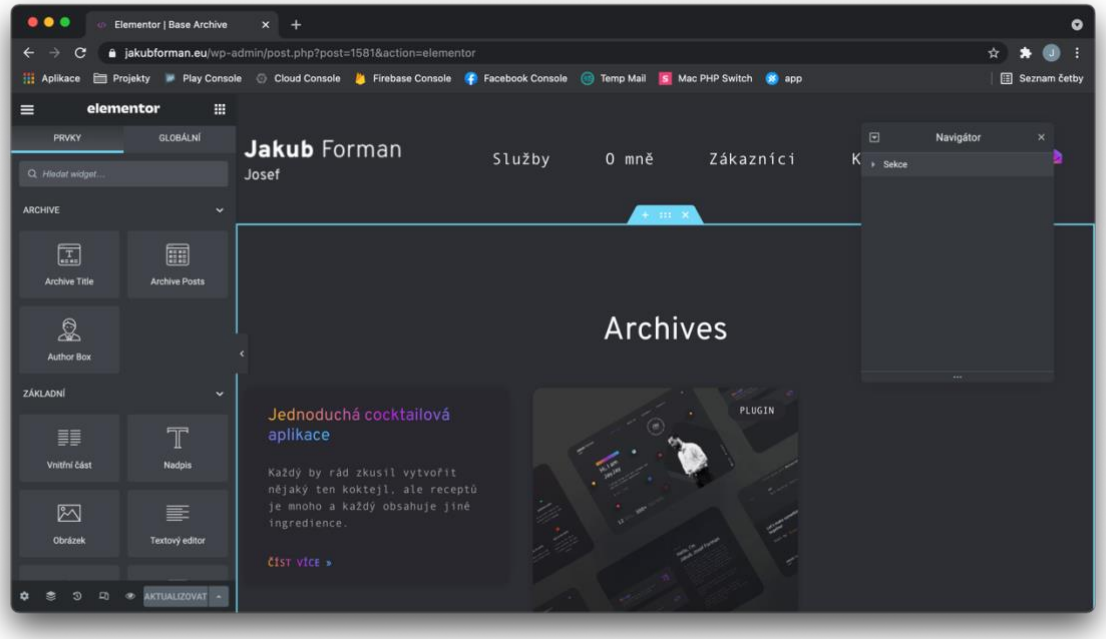
Obrázek 7: Elementor Theme Builder

2.4 Globální nastavení vzhledu

Elementor nabízí možnost vytvořit si globální nastavení, které bude aplikováno na vizuální prvky. Toto nastavení lze zobrazit při kliknutí na hamburger menu v levé části obrazovky při editaci libovolné šablony nebo příspěvku v Elementoru a zvolení položky Site Settings. Stránka se přepne do režimu globálního nastavení, kde lze nastavovat výchozí barevnou paletu, výchozí paletu fontů, specifikace hlavních textových HTML tagů, design tlačítek a

obrázků, formulářových prvků, faviconu, logo, pozadí tagu <body>, rozvržení zalamování obsahu u responzivního vzhledu, výchozí nastavení šířky obsahu uvnitř sekcí a mezeru mezi Widgety, základní vlastnosti Popupů a Lightbox, vložit vlastní CSS (nejedná se o stejné CSS které lze vkládat v přizpůsobení výchozí šablony). [22]

2.5 Orientace v Elementoru



Obrázek 8: Orientace v Elementoru

Editor Elementoru nabízí uživatelům přehledný a intuitivní design, pomocí kterého se lze v editoru zorientovat v řádu minut a je rozdělen na 2 části:

- levý sidebar – hlavní navigační prvek s galerií Widgetů a přizpůsobením stránky nebo šablony, ukládáním a dalšími funkcemi,
- hlavní obsah – plátno, kde lze přetahovat prvky (Widgety) a kontrolovat zobrazení v responzivním režimu,
- pravý sidebar – navigátor, ten je zobrazen pouze pokud si jej uživatel připne k pravému okraji, ve výchozím stavu je skrytý a po zobrazení poletuje (plave) na obrazovce.

2.5.1 Nastavení editované stránky, šablony

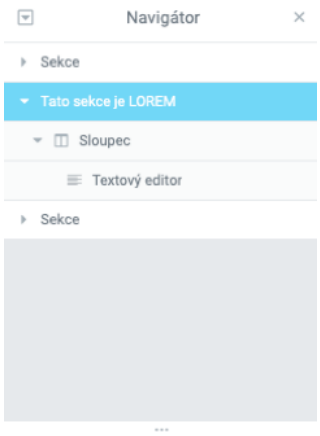
Základní nastavení stránky nebo šablony je podobné tomu, které lze nalézt u Widgetů. Obsahuje základní nastavení jako je: titulek stránky, stav stránky, náhledový obrázek, styl zobrazení stránky, přizpůsobení vlastností CSS a někdy i pokročilé volby. [24]

Obrázek 9: Navigační lišta – nastavení

2.5.2 Navigátor

Navigátor umožňuje rychlou manipulaci s Widgety, jejich přemisťování, orientační pojmenování a přehlednost. Navigátor lze připnout k pravé části obrazovky místo plovoucího okna. [23]

Obrázek 10: Navigační lišta – Navigátor



Obrázek 11: Navigátor

2.5.3 Historie

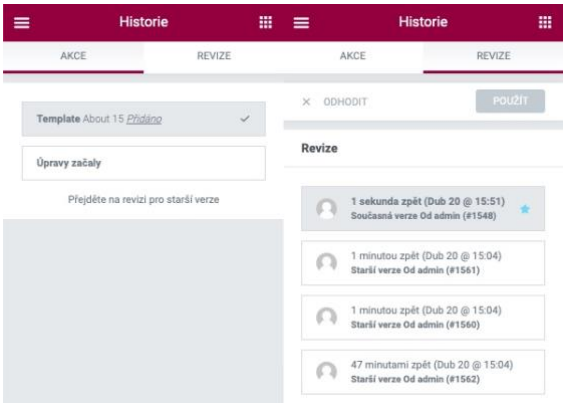
Elementor už ve své free verzi obsahuje vestavěnou Historii revizí obsahující dvě karty [25]:

- Karta Akce zaznamenává každou akci, kterou uživatel v Elementoru provede. Akce také umožňují procházet seznam akcí, zpětně sledovat kroky uživatele a libovolně se mezi těmito kroky pohybovat. Každá akce obsahuje kompletní informace – o jaký

prvek šlo, jaký to byl typ změny nebo informace o doplnění textu. Pro starší záznamy a možnost vracet se v čase dále do historie je zde karta Revize.

Revize je bod obnovení vždy při znovunačtení stránky, mezi kterými se lze libovolně přepínat. Revize již neobsahuje jednotlivé akce jako v předchozím případě, ale lze pomocí revizí „vracet čas" na jednotlivé uložené záznamy.

Obrázek 12: Navigační lišta – Historie



Obrázek 13: Historie – panel

2.5.4 Responzivní mód

Responzivní návrh je dnes součástí každého webu a jelikož existuje celá řada různých šířek obrazovky, je žádoucí mít co největší kontrolu nad velikostí a proporcionalitou grafických a textových prvků. Elementor přichází s jednoduchým přepínačem mezi Desktop, Tablet a Mobil zobrazením, který vychází z roky ověřených zkušeností nejčastěji používaných šířek a výšek obrazovky. Ve zvoleném módu lze editovat všechny Widgety tak, aby bylo dosaženo požadovaného zobrazení. Pokud je nastavení editovaného Widgetu možné upravit na rozdílných zařízeních, je u editované části zobrazena ikona znázorňující responzivní režim . Tato hodnota se bude upravovat pouze pro zvolený responzivní režim. Pokud u volby tato ikona není, provede se změna globálně na všechny šířky displejů. [26]

Některé prvky lze na jiných šířkách obrazovky skrýt, například zobrazit jiné rozložení webu pro telefony a jiné pro desktopy. V editoru se tato změna projeví tak, že je prvek „utlumený“. [26]

Obrázek 14: Navigační lišta – Responzivní mód

2.5.5 Náhled

Elementor tak jako WordPress umožňuje zobrazit náhled změn dříve, než se změny publikují na produkční verzi stránek. Jde o WordPress review režim, ve kterém také pracují klasické příspěvky a stránky WordPressu.

V případě, že se jedná o editaci Šablony, může být po kliknutí zobrazeno ještě jedno tlačítko s popisem Settings. Toto tlačítko umožňuje zobrazit tuto šablonu nad libovolným dynamickým obsahem. Tento náhled může sloužit jako vizuální porovnání na konkrétní stránce, kam by uživatel umisťoval Šablonu typu Popup a jiný. Toto nastavení lze také nalézt při kliknutí na ikonu nastavení a zobrazení položky Preview Settings. Z vlastní zkušenosti doporučuji na tento Preview dynamického obsahu nespoléhat, jde o jedinou neduhu Elementoru. Občas nezobrazí žádný dynamický obsah, což může vypadat „že se něco rozbilo“. Ale tak to není. Vše funguje, jen se nenačetl dynamický obsah.

Obrázek 15: Navigační lišta – Náhled

2.5.6 Akční tlačítko publikace

Akční tlačítko slouží pro uložení a zveřejnění, uložení jako konceptu, uložení jako šablony a v případě editace některých šablon i nastavení podmínek pro zobrazení šablony. Jde o univerzální tlačítko, které se přizpůsobuje podle toho, co právě uživatel v editoru upravuje.

Obrázek 16: Navigační lišta – Akční tlačítko

2.6 Uživatelské role a zabezpečení nepozornosti uživatelů

Bezpečnost systému WordPresu a jeho doplňků je tlačena na první místo a není tomu jinak aniv pluginu Elementor. WordPress nabízí možnost rolí uživatelů, které byly vytvořeny za účelem přiřazení rolí lidem v závislosti na tom, k čemu jim chce administrátor poskytnout přístup, ale nejsou optimalizovány pro potřeby webových designérů. Role předdefinované systémem jsou omezeny na základní práci s oprávněními. Například, jestli má uživatel přístup k určitému příspěvku nebo může vytvářet nové uživatele a tak podobně. Tento přístup by v Elementoru nebyl možný použít skrze jeho zaměření a rozsah jeho možností. Typickým příkladem, proč nelze tento přístup použít, může být jakákoliv stránka vytvořená designérem v Elementoru, na které se klient chystá změnit texty, ale nedopatřením pohne s nadpisy nebo obsahovou částí, což vede k rozhození designu a následnému hovoru administrátorovi s tím, že web nefunguje.

Žádný klient se nesnaží design ani ostatní věci úmyslně pokazit. Ale pokud klienti získají přístup k nástrojům, s nimiž nevědí, jak pracovat, mohou rychle nastat problémy. Koneckonců, s velkou mocí přichází i velká odpovědnost. Je snadné udělat obrovskou chybu, i když se právě klient snaží změnit nějaký text nebo trochu pohnout věcmi. Například klient si často neuvědomí, že pokud mění pozici titulku na počítači, musí tuto změnu zkontrolovat i na tabletu a telefonu, případně je upravit také.

Role jsou řešením, které vyřeší částečně tento problém. Jelikož role WordPress jsou obecně definované a nejsou natolik flexibilní pro pluginy typu Elementor vyžadují, aby je tyto pluginy implementovaly mírně rozdílnou formou. V Elementoru se tato část nazývá **Správce rolí Elementor**. Tento nástroj, vytvořený právě pro designéry, nabízí další funkce a rozšiřuje možnosti základních rolí. Například lze určité uživatele zcela vyloučit z editoru Elementor nebo jim poskytnout přístup pouze k obsahu. Tímto způsobem jde snížit počet nehod ze strany klientů a chránit tak jedinečný design.

2.6.1 Jak fungují role ve WordPressu

Role WordPressu byly vytvořeny, aby řešily běžný problém, který mají správci webů a blogů, když potřebují přivést více autorů a editorů: Jak jim můžou zabránit v tom, aby neměli autorizaci k nastavením, která by neměla být pro ně přístupná nebo je nepotřebují?

Za pomocí uživatelských rolí je snadné přidat lidi na web nebo e-shop a administrátor se nemusí bát, že by se tito lidé dostali do částí webu, kam nepatří. Tyto role jsou vytvořeny tak, aby omezovaly lidi ve funkcích, ke kterým nepotřebují mít přístup.

2.6.2 Správce rolí Elementor

Tento správce rolí byl týmem Elementor navržen tak, aby umožnil designérům povolit uživatelům možnost vytvářet obsah bez úprav citlivých částí designu webu. Elementor implementuje již existující základní role uživatelů z WordPressu a také s nimi pracuje již v základním nastavení.

Do správce rolí lze vstoupit skrze administračním menu **Elementor ⇢ Správce rolí**. Zobrazí se seznam s výpisem všech rolí obsažených ve WordPressu. Každá role bude mít rozbalovací nabídku. Kliknutím na tuto roli se zobrazí jedno nebo dvě zaškrtávací pole:

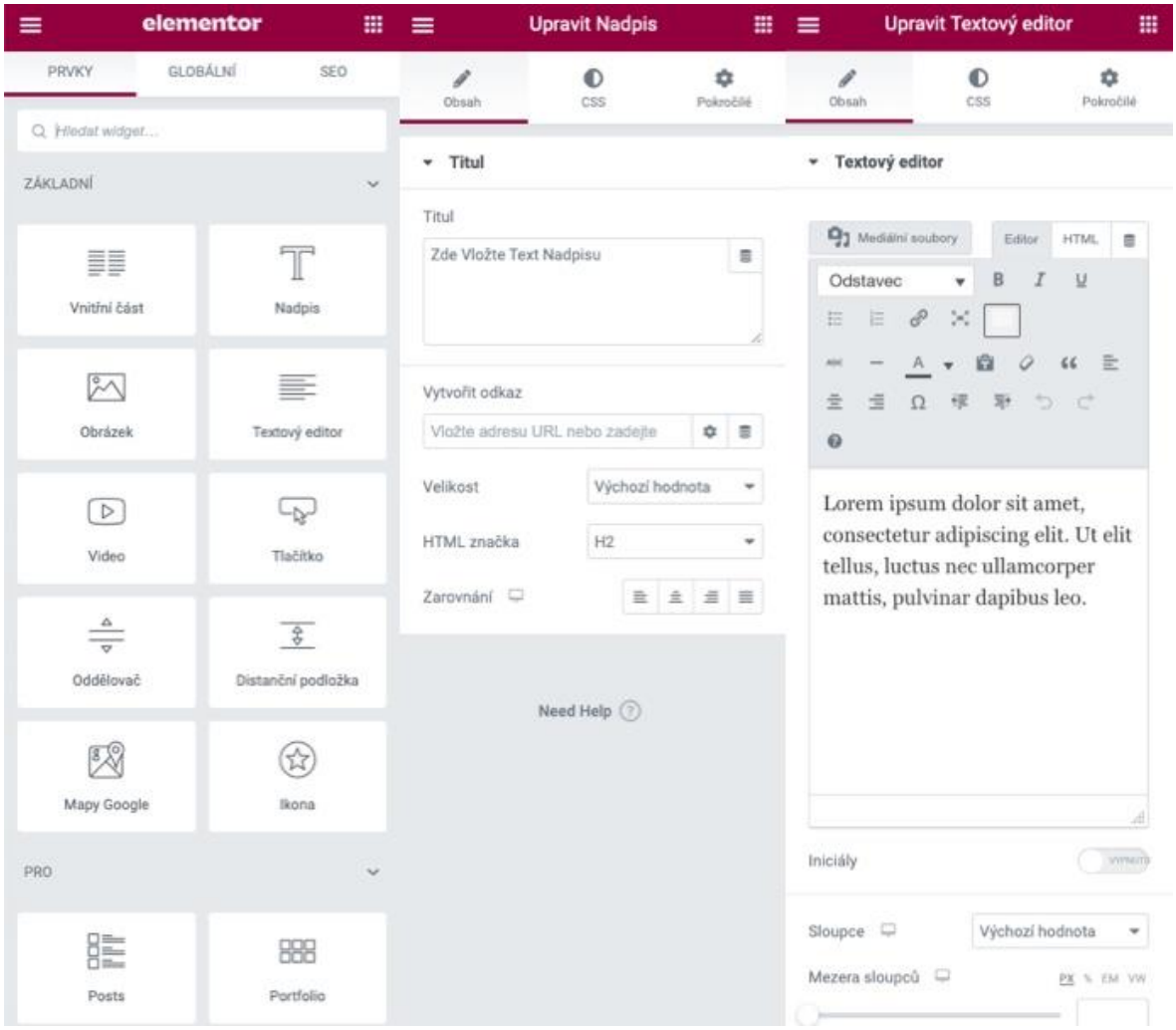
- Žádný přístup k editoru
- Přístup pouze k úpravám obsahu (Access to edit content only)

Zaškrtnutím prvního pole všichni uživatelé s touto rolí neuvidí modré tlačítko *Upravit pomocí Elementoru*. Nebudou mít absolutně žádný přístup k Elementoru ani k žádnému obsahu na stránce s ním navržené. Pro mnoho klientů tato možnost nic nemění, jelikož potřebují editovat obsah, který je vytvořený v Elementoru. V případě Elementor PRO, je zobrazena i druhá možnost: Přístup pouze k úpravám obsahu (Access to edit content only). Tato možnost umožní této roli otevřít editor Elementor a přistupovat k obsahu na této stránce, například k textu a obrázkům. Nebudou však moci přesouvat sloupce, přidávat widgety ani jiné funkce, které by mohly změnit vzhled vašeho webu. Tím se minimalizuje spousta problémů, které by se mohly objevit. Zároveň umožní klientům měnit text a obrázky, aniž by museli neustále žádat o pomoc správce webu.

Tým Elementoru doporučuje tyto role projít a nastavit jim oprávnění podle svého uvážení tak, aby přístup k editačním funkcím editoru měli opravdu jen spolehliví uživatelé webu, kteří nemohou poškodit strukturu designu.

2.7 Widgety

Widgety jsou jádrem celého Elementoru. Jejich funkce je takřka neomezená. Základní Elementor obsahuje desítky widgetů, které jsou zdarma [27]. Pro rozšíření této knihovny o další Widgety a funkce je třeba použít Elementor Pro nebo jiný alternativní plugin.

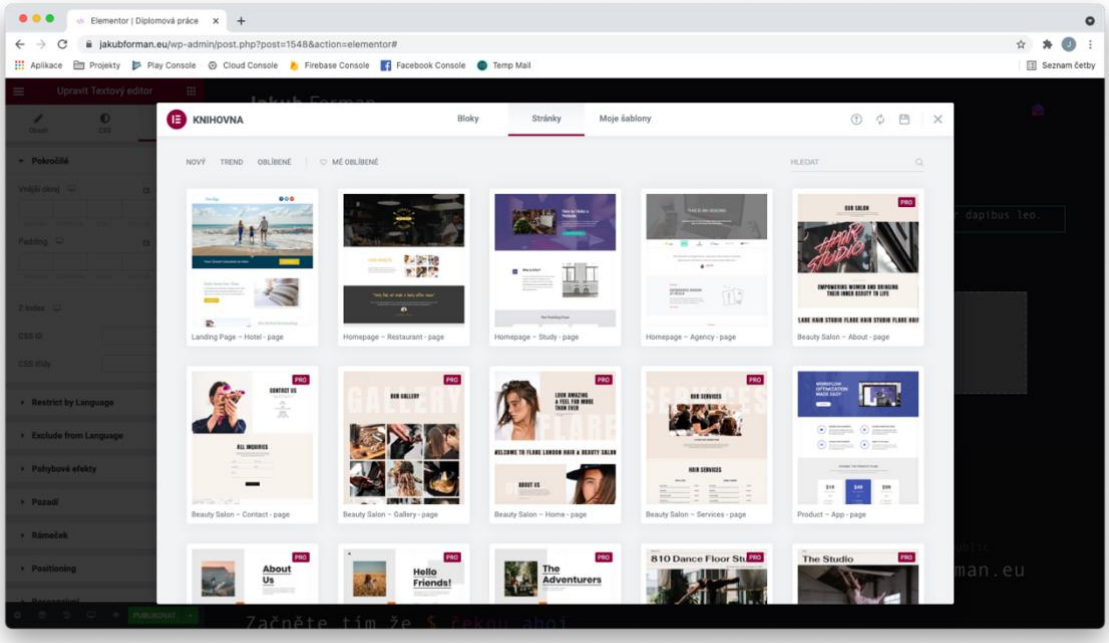


Obrázek 17: Panel widgetů, nastavení widgetů

Na obrázku 4 je zobrazena galerie Widgetů – **PRVKY,** z nichž lze Widgety přetahovat do plátna – sekcí nebo sloupců na webové stránce. Po kliknutí na Widget v editačním okně nebo je umístěn na plátno webové stránky je zobrazeno jeho nastavení. V první záložce je obsah, většinou je zde to, co se bude vykreslovat na plátno a jakým stylem, tedy texty, obrázky a zarovnání obsahu. Druhá záložka je CSS, kde lze upravit některé **CSS** vlastnosti podle zvoleného Widgetu. Záložka **POKROČILÉ** obsahuje pokročilou editaci Widgetu. Umožňuje nastavit vnitřní a vnější odsazení, z-index, CSS ID a CLASS, pozadí, pozicování, vlastní CSS a další pokročilé funkce. [27]

2.8 Knihovna Elementoru

Knihovna nabízená Elementorem je nejen přehledná, ale ve formě PRO nabízí velmi rozsáhlou databázi bloků a stránek, které lze do stránky vložit a vytvořit tak stránku během pár kliknutí. [16]



Obrázek 18: Knihovna Elementor

Elementor umožňuje po vložení Bloku nebo Stránky z knihovny do plátna stránky vše upravovat. Vše, co je připravené týmem Elementor je vždy responzivní a plně připravené pro produkční použití. [16]

2.8.1 Vlastní šablony

Tak, jak je zmiňováno v bodě **2.3** lze vytvářet vlastní šablony. Tyto šablony lze opakovaně používat, vkládat do stránek a upravovat je dle potřeb daného vzhledu.

2.9 Rozdíl mezi Elementor a Elementro Pro

Elementor nabízí mnoho jednoduchých a pokročilých funkcí. Klasický Elementor slouží pro tvorbu Landing Pages, příspěvků a stránek a dalších typů příspěvků včetně responzivity. Naopak verze Elementor PRO nabízí rozšíření Widgetů, rozšířenou možnost vytváření šablon a archivů, integrace s třetími stranami, dynamické vkládání obsahu z jiných částí WordPressu nebo pluginů, nativní podporu WooCommerce, formuláře pohybové a animační efekty a další. [15]

Existují však alternativy, které umí Elementor Pro částečně nahradit, avšak nedosahují tak plné integrace jako Elementor Pro.

2.10 Možnosti rozšíření funkcionality

Ve WordPressu je často využíván styl dvou verzí pluginů. Jedna verze je dostupná z oficiální knihovny pluginů WordPress – tato verze je základní verzí pluginu a je zdarma. Druhá verze je placená a možnosti první verze pluginu rozšiřuje anebo jej plně nahrazuje. V případě Elementoru je základní plugin rozšířen o jeho Pro verzi, která prochází stejně jako verze zdarma pravidelnými aktualizacemi. Kromě oficiálního Pro pluginu od Elementoru je možnost rozšíření nebo upravení funkcionality dalšími pluginy třetích stran.

2.10.1 Existující rozšíření třetích stran

Již přímo v knihovně pluginů WordPress se nachází celá řada rozšíření, které funkčnost Elementoru upravují nebo rozšiřují. Velká část z nich je opět dodávána ve dvou verzích Free a Pro. Na oficiálních stránkách Elementoru (*https://elementor.com/addons/)* je jich zmíněno 20+. Většina z těchto pluginů přidává nové Widgety do Elementoru a některé rozšiřují funkčnosti Elementoru skrze optimalizaci, propojení s dalšími pluginy a službami.

2.10.2 Vlastním rozšířením

Další možností, jak rozšířit funkčnosti Elementoru, je vytvořit vlastní plugin, který bude splňovat přesně ty požadavky, které je pro web nutné mít. Tvorbou tohoto pluginu se zabývám v praktické části, kde rozšiřuji funkčnost sloupců a sekcí, která je u některých designů dosti nepřesná a nepodporuje určité funkce.

3 POUŽITÉ TECHNOLOGIE

Při vývoji pluginu, který je součástí praktické části, byly použity další technologie spolu s WP a Elementorem. Stručně zde vysvětluji ty technologie, které lze použít a také se s nimi lze setkat při práci s WP a Elementorem.

3.1 HTML, CSS, JavaScript

Jde o skupinu, která neodmyslitelně patří k tvorbě webových stránek. **HTML** je značkovací jazyk používající se při tvorbě webů. Jeho struktura je podobná struktuře XML. Elementy jsou přesně dané a začínají otevíracím tagem a končí tagem ukončovacím (například). Některé elementy nejsou párové a mohou existovat sami o sobě bez uzavíracího tagu (například
). Tyto elementy mohou obsahovat celou řadu atributů, které upravují jejich vlastnosti, vizuální vjem a chování. [28]

```
<h1 id="base-title">Toto je základní nadpis</h1>
```

CSS – kaskádové styly jsou používané pro tvorbu grafických prvků za pomocí úpravy HTML atributů. Aktuální standard CSS3 nabízí mnoho dynamických možností, mezi které patří barevné přechody, animace, transformace, proměnné a spoustu dalšího. Před příchodem tohoto standardu se tyto funkce musely vytvářet JavaScriptem a dopočítávat matematickými funkcemi.

```
h1 {
  color: white;
  text-align: center;

  transform: rotateX(150deg);
}
```

JavaScript je skriptovací jazyk používaný v HTML a na některých serverech. Chod jazyku zajišťuje prohlížeč, ve kterém je stránka s tímto JavaScriptem a HTML načtena. Nejčastěji se používá spolu s CSS, za pomocí kterého lze vytvářet dynamicky měnící se obsah s přesným designem.

```
document.getElementById(base-title).innerHTML = Date()
```

3.2 PHP

PHP je podobně jako JavaScript programovací jazyk, ale tento jazyk pracuje naopak na serveru. Je tedy plně odstíněn od uživatele, který konzumuje webový obsah a uživatel s ním nepřijde do styku. PHP je hojně rozšířený jazyk, jelikož jej poskytují všechny webhostingy za minimální provozní cenu. Tento jazyk je velmi jednoduchý a lze se v něm rychle zorientovat a naučit se jeho základní principy. Kromě základních sčítacích operací lze v tomto jazyku vytvořit statické generátory obsahu například pro diskusní fórum, počítadlo, anketu, graf nebo lze celý obsah webu smazat. Pro ty, kteří chtějí web propojit s databází PHP nabízí řadu konektorů z nichž nejpoužívanější je MySQL. [29] Funkce PHP lze rozšířit již existujícími balíčky nebo knihovny, které jsou většinou dostupné skrze Composer.

```
<?php
echo "<h1 id=\"base-title\">Toto je základní nadpis</h1>";
?>
```

3.3 Composer

Composer je nástroj pro správu závislostí v PHP. Umožňuje programátorovi deklarovat knihovny, na kterých je projekt závislý a bude je spravovat (instalovat / aktualizovat) za programátora. Tento „balíčkovací systém“ je dodáván formou jednoho souboru a lze jej spustit bez instalace. Pro svůj běh vyžaduje PHP a někdy i povolení administrátora pro instalaci globálních balíčků. Composer využívá jako zdroj knihoven Packagist. Jde o repozitář, ze kterého Composer stahuje knihovny podobně jako z GITu. [30] V projektu je uložen jako soubor `composer.json`, který obsahuje všechna potřebná nastavení pro import knihoven, verzi projektu, autora, Autoloading složek a spoustu dalšího.

II. PRAKTICKÁ ČÁST

4 VYUČOVÁNÍ VÍCE NEŽ HTML A CSS

Na většině dnešních škol jsou ŠVP postavené tak, že práci s redakčními systémy do výuky nezahrnují, nebo je zmiňují studentům středních škol jen okrajově. Téměř na každé střední škole se lze setkat s tvorbou webových stránek pomocí HTML, CSS a JavaScriptu, kterou většinou následuje PHP, objektové programování v C# (.NET) nebo jiné vývojové nástroje se zaměřením na algoritmizaci.

Spolu s tvorbou webových stránek se studenti v jiném (nebo stejném) předmětu často učí grafiku, ale prakticky ji s kódem propojují jen velmi omezenou formou. Spojení webů a grafiky může být jedna z nejrychlejších cest, jak studenty navnadit na budoucí studium informatiky právě díky rychlé názornosti. Pomocí redakčních systémů a nástrojů pro editaci obsahu jako je Elementor lze žákům jednoduše, a hlavně rychle objasnit základy, které by se psaním kódu prodlužovaly na několik vyučujících hodin a namotivovat je tak k jejich studiu.

V praxi se čím dál tím častěji setkávám s firmami, které hledají externí pracovníky pro tvorbu prezentačních webů za pomocí nástrojů pro editaci obsahu, za účelem ušetření nákladů vynaložených na ruční psaní webových stránek nebo šablon pro redakční systémy. Pro studenty by taková příležitost mohla být další motivací k tomu, aby se studiem zabývali více hlouběji a byli tak lépe připraveni k budoucímu uplatnění v praxi. Důvodem, proč Elementor využívá přes 5.000.000 webů na světě a toto číslo stále roste je fakt, že tvorba webové prezentace se z ručního psaní kódu zkracuje na několik hodin maximálně dní – v případě existence prototypu designu a všech podkladových materiálů.

5 PROČ VYVÝJET VLASTNÍ PLUGIN

Do této kapitoly jsou zahrnuty možnosti, proč vyvíjet vlastní plugin pro WordPress, kdy se může hodit a kdy je naopak plugin zbytečné vyvíjet. V následujících kapitolách je popsán vývoj rozšíření pro WordPress, který rozšíří již existující plugin Elementor o několik užitečných funkcí.

5.1 Kdy plugin vyvíjet

Každý, kdo si WordPress vyzkouší zjistí, že se bez patřičných pluginů neobejde. Systém je v základu dostatečný na psaní textu, vytváření blogových příspěvků a stránek. Pokud by chtěl administrátor webu funkce rozšířit, má v podstatě svázané ruce. Jedinou cestou jsou pluginy, ale ani ty někdy nenabízí vše, co administrátor na webu požaduje. V takovém případě je nedílnou součástí tvorby webových stránek na WordPressu vytvoření vlastního pluginu.

Ten může být navržen několika způsoby, nejlepší je však dodržovat strukturu, jakou uvádí WordPress v oficiální dokumentaci. Před vytvořením vlastního pluginu většinou předchází analýza podobných, již existujících pluginů nebo částí kódu podle požadavků. Je totiž možné, že plugin s podobnou funkcionalitou již existuje, a proto nebude nutné jej psát od čistého souboru a může se jednat jen o rozšíření již existujícího pluginu a do značné míry ulehčení práce.

5.2 Kdy plugin nevyvíjet

Pokud se při analýze požadavků na plugin zjistí, že existuje podobný nebo nabízí velmi podobnou funkcionalitu tomu, jaká by se očekávala od nově vytvořeného pluginu je většinou zbytečné jej vyvíjet.

Ve WordPressu existuje dříve zmíněný soubor **functions.php**, který je většinou součástí všech šablon a chová se podobně jako plugin. Pokud se jedná o drobnou úpravu, která nevyžaduje velkou část kódu, je možné ji zapsat pomocí Hooku do tohoto souboru. Je ale nutné pamatovat na to, že při aktualizaci vzhledu se tento kód může přepsat.

6 NÁVRH WEBU POMOCÍ WORDPRESS

V této kapitole bych chtěl představit základní návrh webu pomocí WordPressu, základní nastavení, které je dobré udělat hned na začátku, vytvoření struktury webu a instalaci a práci s pluginem Elementor.

6.1 Základní nastavení

Před začátkem vytváření jakéhokoliv obsahu je dobré si zvolit nastavení WordPressu. Je možné, že základní nastavení bude dostačující, ale i tak je nutné jeho nastavení zkontrolovat. Toto nastavení lze nalézt v administraci v sekci **Nastavení**.

6.1.1 Obecné

V této sekci je dobré určit si název webu, který bude mít do 60 znaků a obecný popis stránky, který by měl být v rozsahu 156–300 znaků. *Tento popis se může měnit podle použitého SEO pluginu na jednotlivých stránkách*.

Instalace WordPressu se mění jen v případě změny hostingu nebo domény. Mezi nejčastější změny je zavedení HTTS (SSL certifikátu). Ve výchozím stavu je URL nastavena s *http://*, ale pro přesměrování na HTTPS je nutné změnit URL na https://. Podrobnosti, jak kompletně přesměrovat domény z HTTP na HTTPS jsou popsány u každého hostingu a většinou jsou popsány také pro redakční systém WordPress.

E-mailová adresa administrátora je výchozí adresa emailů. Na tuto adresu chodí veškeré upozornění z redakčního systému a měla by být plně funkční. WordPress se pravidelně na tuto adresu dotazuje jestli je aktuální při přihlašování do administračního rozhraní.

Registrace (členství) je dobré mít zakázané, předejde se tím nechtěným útokům ze strany nežádoucích návštěvníků v případě, že by se někde ve WP nebo některém pluginu objevila chyba. Stejně tak je doporučeno mít výchozí úroveň nových uživatelů jako návštěvník. Ostatní nastavení v sekci obecné jsou převážně časové nastavení a zobrazení formátu data a času, které lze přizpůsobit i v pluginu Elementor.

6.1.2 Publikování

Zde stačí nastavit výchozí rubriku (někdy označovanou taky jako kategorii) pro příspěvky. Tuto rubriku lze vytvořit v **Příspěvky ⇢ Rubriky**. Výchozí formát příspěvků se mění, jen v případě, že je stránka zaměřena na blogové psaní v určitém formátu, například se často

píšou příspěvky, které jsou skutečné rozhovory. Více o těchto formátech si lze přečíst zde: *https://www.wpbeginner.com/glossary/post-formats/*.

6.1.3 Zobrazování

Jde o sekci, na které lze nastavit základní styl zobrazení stránky. Jestli se jako úvodní stránka zobrazí nejnovější příspěvky rozložení (využívané u novinkových serverů) nebo se bude zobrazovat specifická statická stránka, která již existuje a novinky (příspěvky) budou zobrazeny na jiné statické stránce tomu určené. Pro novinky, kategorie a vyhledávání lze nastavit maximální počet těchto příspěvků na stránku (v Elementoru lze nastavit jinou hodnotu). Taktéž zde lze obecně přizpůsobit RSS a soubor robots.txt.

6.1.4 Nastavení komentářů

Pokud se administrátor webu rozhodne využívat vestavěné komentáře WordPressu, je nutné nastavit, jak se tyto komentáře mají chovat. Pokud je správce na webu používat nechce, lze je vypnout a případně povolit jen u určitých příspěvků. Nastavením komentářů se předejde spamovým komentářům odkazující na nežádoucí weby, opakujícím se komentářům a dalším problémům s komentáři.

6.1.5 Média

Média obsahují nastavení velikosti obrázků využívající se na webu – toto nastavení se mění jen podle specifického zaměření webu a ve výchozím nastavení je dostačující. Existují pluginy, které velikosti obrázků umí rozšířit o další možnosti, ale v základu si každý vystačí s tím, co WordPress nabízí.

6.1.6 Trvalé odkazy

Jsou jednou z částí SEO na webech – měly by být uživatelsky přívětivé. Odkazy se jinak formátují u stránek a jinak u příspěvků. Základní nastavení slouží obecně pro odkazy a jaký styl si administrátor vybere je na něm. Mnou používané je často vlastní struktura s **/%category%/%postname%/** za pomocí ní se všechny příspěvky řadí do kategorií. Nedochází tak ke kolizi jmen s názvy stránek.

Jak vypadají odkazy stránek a příspěvků se pokusím vysvětlit v následujících ukázkách, které jsou vytvořeny podle mé vlastní struktury, kterou na webech využívám.

Příklad 1:

Stránka O nás

Název stránky: O nás

https://exmaple.com/o-nas

Příklad 2:

Stránka historie s nadřazenou stránkou (rodičovskou)

Název stránky: Naše historie

Nadřazená stránka: O nás

https://exmaple.com/o-nas/nase-historie

Příklad 3:

Příspěvek zařazený v kategorii léto

Název příspěvku: Kam v létě na vodu

Kategorie: léto, dovolená …

https://exmaple.com/leto/kam-v-lete-na-vodu

Z hlediska dostupnosti odkazů v budoucnu, není vhodné často měnit nastavení trvalých odkazů. Pokud by se tyto odkazy časem (v rozmezí let) měnily, nastane situace, kdy staré odkazy budou neplatné. Pokud tedy na sociálních sítích jsou odkazy ve struktuře http://example.com/archiv/123 a následně se tyto odkazy změní na jinou strukturu, budou s největší pravděpodobností odkazovat na stránku 404 obsah nenalezen a zákazníci nebo čtenáři blogu budou zaskočeni a pravděpodobně stránku opustí.

6.1.7 Ostatní nastavení WordPress

Ostatní sekce nastavení nejsou tak podstatná pro používání webu a můžou se lišit podle aktuálně nainstalované verze WordPressu. Nově WordPress do nastavení přidal sekci Soukromí, která navazuje na zásady ochrany osobních údajů a provede administrátora nastavením těchto zásad.

6.2 Stránky a příspěvky

WordPress nabízí možnost vytvářet Stránky a Příspěvky, které se ve své podstatě moc neliší. Na databázové úrovni jsou příspěvky, stánky, produkty a další uloženy ve stejné tabulce – všechny tyto možné příspěvky se nazývají **post**. Rozdíl je však v typu postu, který se do této tabulky ukládá, za pomocí něhož lze rozlišit data na stránky, příspěvky, produkty a další. Proto se lze u vytváření nového obsahu setkat většinou se stejnou strukturou typu: Titulek, obsah, tagy, náhledový obrázek, trvalý odkaz, stav publikování apod.

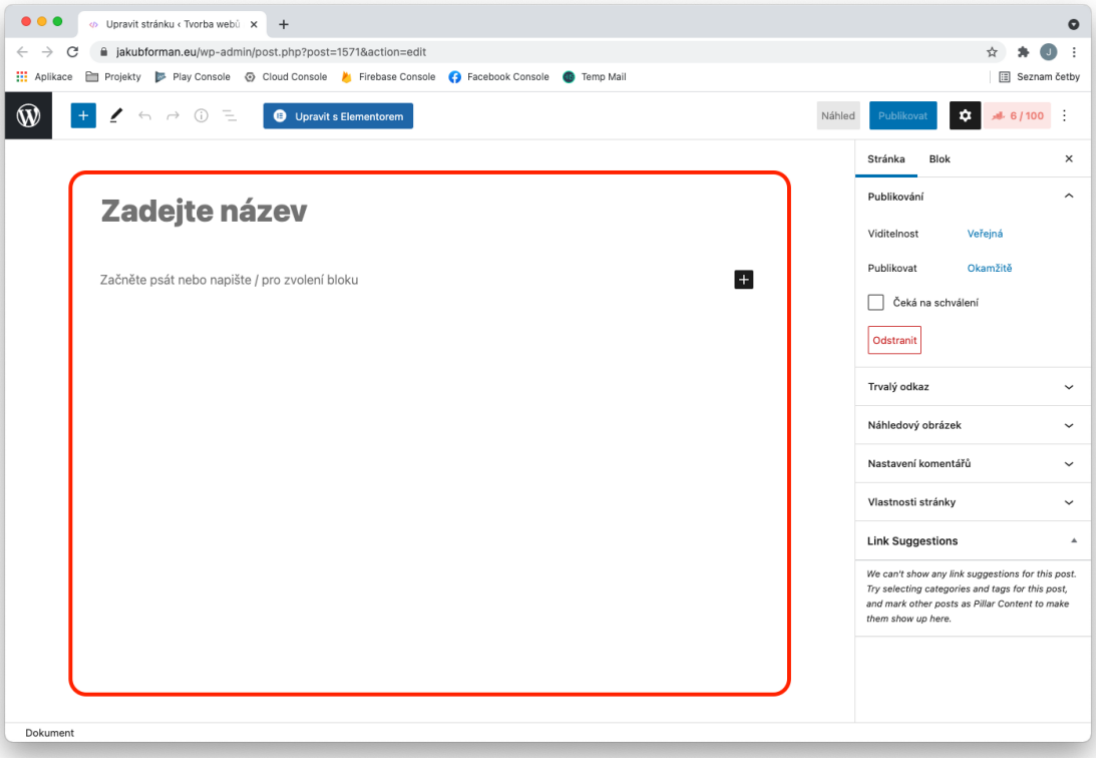
Aby bylo možné ke každému typu postu přidat rozdílné informace je v databázi druhá tabulka, která tato data ukládá k postům. V důsledku toho lze vytvářet „neomezené“ typy postů a mít tak u každého jiné vlastnosti a nastavení.

6.2.1 Rozdílná funkcionalita

Rozdíl v hlavních postech Příspěvky a Stránky je v jejich zobrazení a možnosti dodatečných informací při vytváření. Stránkami se chápe statický obsah, který je v čase neměnný (nebo minimálně) a jeho aktuálnost je po dlouhé časové období neměnná (například stránka O nás). Rozdílně jsou na tom příspěvky, které již nejsou „statické“ a jejich aktuálnost časem ztrácí na hodnotě (například: počasí na příští týden). Příspěvky jsou zobrazovány na stránce blogu (novinek) a na stránkách kategorií, navíc lze určit příspěvky, které jsou připnuté a jejich pozice na blogu je tak na prvním místě nad všemi příspěvky, i když byly přidány před měsícem.

6.2.2 Editor

Pro vytváření obsahu je WordPress vybaven základními editory obsahu, kde lze psát texty. Původní editor, který byl dlouhá léta WordPressem využíván je postupně nahrazován editorem Gutenberg, který si klade za cíl nejen úpravu a formátování textu, ale možnost vytvářet obsah za pomocí uživatelsky editovatelných bloků. U některých pluginů, kde se tvoří obsah, je možné stále nalézt starý editor, ale i zde vývojáři postupně přechází na nový editor.



Obrázek 19: Ukázka editoru Stránek a Příspěvků

Kromě základních editorů obsahu WordPress nabízí dodatečné nastavení postu jako je publikace, upravení trvalého odkazu, náhledový obrázek, nastavení komentářů a další nastavení. V případě příspěvků jsou to pak rubriky, štítky (tagy), stručný výpis příspěvku a další nastavení podobné jako u stránek.

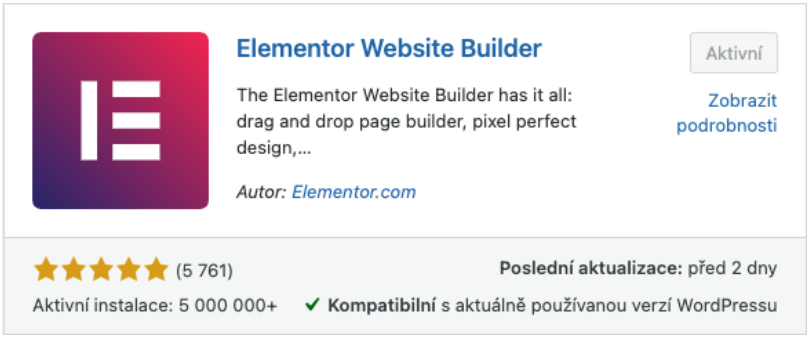
Kromě editace Příspěvku nebo Stránky jako celku umožňuje nový editor editovat jednotlivé bloky, které byly do editoru vloženy přes modré tlačítko +. Po tom, co je blok vložen do editoru je možné jej v pravém sloupci, kde je nastavení postu editovat, přepnutím na záložku blok. Možnosti bloků a jejich editace se liší podle vývojáře a podle použití bloku.

6.3 Instalace Elementoru

Podobně jako Gutenberg od WordPress funguje i Elementor, který pro tvorbu stránek budu využívat. Tento editor využiji pro tvorbu stránky pro přehled mých projektů, které budou

vkládány jako příspěvky se specifickou kategorií. Elementor pro svou funkci staví na tzv. Widgetech, které reprezentují bloky, které lze uživatelsky přizpůsobit.

Pro možnost využití Elementoru je nutné si jej doinstalovat formou pluginu z oficiální knihovny WordPress a spolu s ním i použít šablonu, která je kompatibilní. V mém případě je to šablona Hello Elementor. Pro instalaci Elementor v **Pluginy ⇢ Instalace pluginů** vyhledám klíčové slovo Elementor a kliknutím na Instalovat a následně Aktivovat jej vložím na web.



Obrázek 20: Instalace pluginu Elementor

Kromě základního Elementoru si doinstaluji PRO verzi, kterou získám jako ZIP z oficiální stránky Elementoru po přihlášení do uživatelské sekce. Tento ZIP nahraji v **Pluginy ⇢ Instalace pluginů** kliknutím na tlačítko **Nahrát plugin**, které zobrazí možnost ZIP vybrat jako soubor z počítače a pokračovat v instalaci a aktivaci. Po aktivaci PRO verze, je nutné aktivovat licenci tohoto modulu, kliknutím na tlačítko Aktivoval licenci. Po aktivaci lze PRO verzi aktualizovat přímo z administrace, podobně jako ostatní pluginy, což je velká výhoda oproti některým konkurenčním řešením.

6.3.1 Základní nastavení

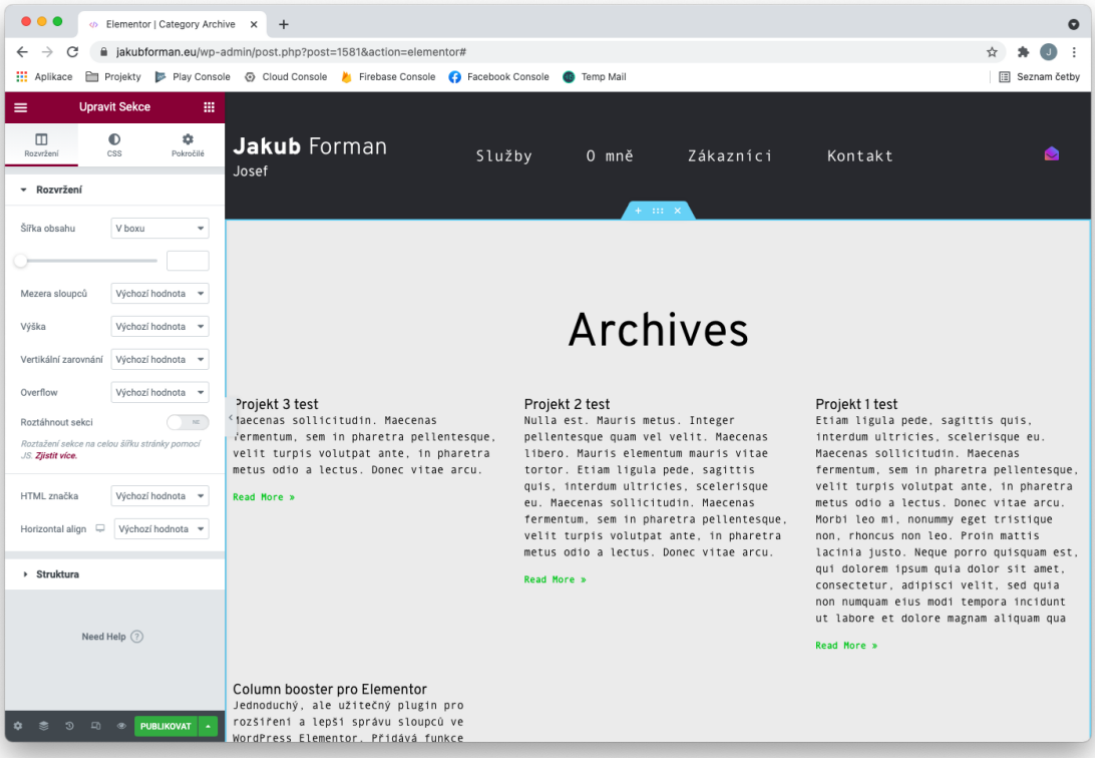
Základní nastavení není nutné, ale pro lepší funkci barev a fontů je výhodnější zakázat výchozí barvy a deaktivovat původní fonty v **Elementor ⇢ Nastavení**. Případně také v **Elementor ⇢ License** aktivovat PRO licenci.

6.4 Vytvoření designu pro stránky archivu

Jelikož vím, že příspěvky budou přibývat a bude jich víc, je asi nejlepší cesta vytvořit archiv, který bude tyto příspěvky slučovat a zobrazovat je na stránce blogu a kategorií. Ruční přidávání a editování stránky by bylo zdlouhavé a nedávalo by moc smysl, jelikož vím, že příspěvků bude v budoucnu více a bude zde i stránkovač.

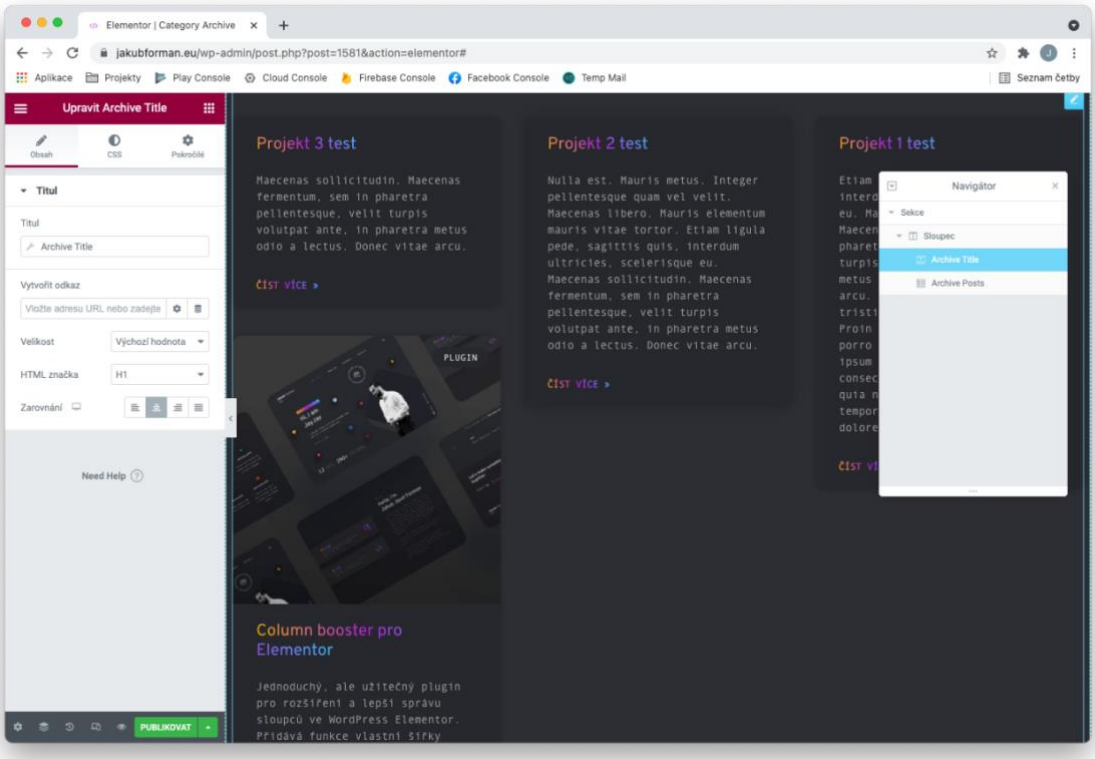
Před tím, než budu vytvářet stránku s archivem je nutné si vytvořit pár fiktivních Příspěvků, které budou pro tuto stránku zobrazovány jako testovací data. Proto si vytvořím 3 jednoduché příspěvky za pomocí *Lorem Ipsum* textu, přidám je do různých rubrik a nastavím potřebné vlastnosti. Pro zobrazení v Elementoru není nutné tyto příspěvky publikovat.

Pro vytvoření šablony stránky, která se bude zobrazovat pouze v případě, že si návštěvník rozklikne kategorii nebo blog, vytvořím šablonu v **Šablony ⇢ Vytvořit šablonu** typu **Archive** a výstižně si ji pojmenuji. Po založení šablony se mohou zobrazit nabízené bloky rozložení, takže si zvolím základní rozložení abych si ulehčil práci.



Obrázek 21: Vytvoření stránky zobrazující archívy – šablona

Po vložení základní šablony (mohl bych ručně vytvářet z jednotlivých Widgetů) začnu s její editací. Pozadí, které je vloženo na sekci, je barevně špatně, takže jej odeberu. Aby se použilo to výchozí ze stránky. Následuje úprava Widgetu *Archive Posts*, který je uvnitř sloupce obsaženého v editované sekci. A pokračuji v editace prvků na stránce, dokud se mi nepodaří dosáhnout požadovaného vzhledu. Elementor automaticky ukládá v průběhu práce rozdělanou práci, ale během práce si vše ukládám také pomocí CTRL + S.



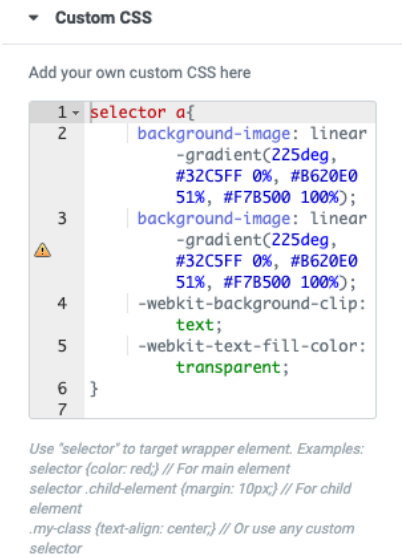
Obrázek 22: Vytvoření stránky zobrazující archívy – finální náhled

Na stránce používám barevné přechody, které jsou řešeny vlastním CSS, jelikož Elementor nabízí pouze základní přechody dvou barev. Tyto barevné přechody jsou vloženy jako dodatečné CSS Class v přizpůsobení zobrazovaného vzhledu Hello Elementor a tato Class je nastavena ve Widgetu *Archvie Posts* jako CSS Class, která ovlivňuje veškeré HTML tagy <a> uvnitř této třídy.

```
.text-gradient-child-links a{
    background-image: linear-gradient(225deg, #32C5FF 0%, #B620E0 51%, #F7B500 100%);
    background-image: linear-gradient(225deg, #32C5FF 0%, #B620E0 51%, #F7B500 100%);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}
```

Obrázek 23: Vložení vlastní CSS třídy k Widgetu v Elementor

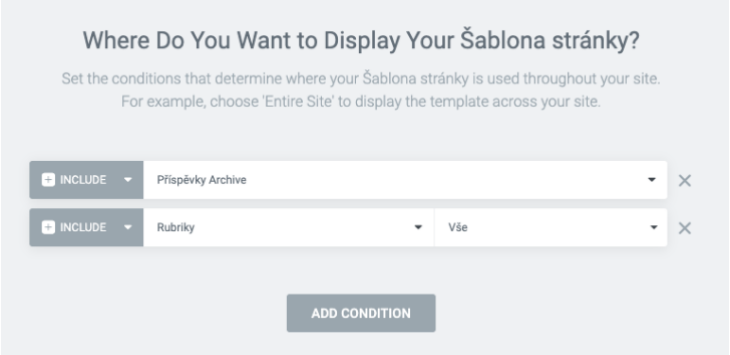
Druhou možností, jak použít vlastní CSS je vložit vlastní CSS kód přímo k Widgetu a použít Elementor ***selector***, který označuje aktuální Widget, ke kterému se přidává vlastní CSS. Použití této metody je v ojedinělých případech správnou volbou, jelikož používám barevný přechod na více stránkách, více Widgetech je výhodnější definovat základní CSS třídu a tu jen vložit k potřebnému Widgetu.



Obrázek 24: Vložení vlastního CSS k Widgetu v Elementor

Publikování archivu pro blog a kategorie lze nastavit po kliknutí na tlačítko publikovat. Zobrazí se stránka s podmínkami, kde lze zvolit požadované nastavení. Pokud by se tato

možnost nezobrazovala, lze ji zobrazit dodatečně rozšířenými možnostmi u tlačítka publikovat / aktualizovat.



Obrázek 25: Publikace designu archívů

Stejným stylem lze vytvořit jednotlivé designy stránek, příspěvků anebo speciálních kategorií, které se mají zobrazovat jinak než zbytek archivů. Jediné omezení ve vytváření designu jsou fantazie autora, který design vytváří a limitace WordPressu. Po publikování a nastavení podmínek, se stránka zobrazuje na místě blogu a kategorií tam, kde by jindy byla stránka z aktuálního vzhledu.

7 ROZŠIŘUJÍCÍ PLUGIN

Elementor pro WordPress je skvělý nástroj webdesignéra, ale ne vše v něm funguje podle představ samotných designérů. Tu a tam se občas stane, že se musí použít vlastní CSS a proklikávat se inspektorem HTML, hledat zanořené prvky, jejich CSS Class, které se ručně upravují, aby se dosáhlo potřebných výsledků. Protože mě při tvorbě šablon v Elementoru chyběla funkce vlastní šířky sloupců s možností zadat ji v pevné šířce nebo možnost dopočítání pomocí calc(), tak jsem tuto možnost přidal. Navíc jsem funkci rozšířil o možnost vložení všech CSS width validních hodnot, což se může hodit při použití vlastních proměnných a nebo již zmiňovaného calc(). Rozšiřující plugin je implementací CSS Flex, jehož nastavení je v Elementoru značně omezené a designér by si ho musel u každého sloupce a sekce psát vlastním CSS, a to značně prodlužuje čas realizace designu.

Tato kapitola slouží zároveň jako možná šablona při vytváření vlastních pluginů pro Elementor a v příloze nebo na GITu lze nalézt i kompletní zdrojový kód.

7.1 Připravení vývojového prostředí

Před začátkem vývoje samotného pluginu si programátor musí připravit vývojové prostředí. V mém případě je to Docker, který mi nahradí veškeré servery (PHP, MySQL, Apache) odstíněně od operačního systému. To však neznamená, že nelze využít jiné prostředí, například využít aplikaci MAMP (LAMP, WAMP, …) pro spuštění serverů nebo i ruční instalaci všech serverů a ovládání přes příkazový řádek nebo terminál.

7.1.1 Docker

Docker je Ope-Source projekt pro automatizaci nasazení aplikací jako přenosných a vlastních kontejnerů, které mohou běžet v cloudu nebo místně. Docker využívá tzv. kontejnerů, jedná se o jednotlivé aplikace nebo služby (i s návaznostmi) zabaleny do jednoho balíku (image). Tento balík lze následně testovat samostatně a následně jej i nasadit do operačního systému. [34] Pro tuto práci jsem zvolil tyto Docker images:

- `wordpress:latest` – Poslední verze WordPressu.
- `wordpress:cli` – CLI příkazy pro WordPress z image `wordpress:latest`.
- `phpmyadmin/phpmyadmin` – Administrace databáze phpMyAdmin.
- `mariadb:latest` – Databáze MySQL spustitelná na ARM Apple M1 CPU.

7.1.2 Napojení na existující instalaci WordPressu

Jako druhý krok je potřeba do vývojového prostředí nainstalovat funkční kopii redakčního systému WordPress (instalováno v Dockeru). Tuto instalaci si v použitém IDE naimportuji jako PHP knihovnu, pomocí kterých bude IDE znát nutné závislosti a bude fungovat našeptávač.

7.2 Vývoj

Při vývoji je dobré dodržet určitou strukturu kódu tak, jak je doporučeno v oficiální dokumentaci. IDE PhpStorm, které při psaní kódu využívám má možnost validace projektu jako WordPress Plugin, za pomocí čehož upravuje kód tak, aby vyhovoval standardům WordPressu. Složkovou strukturu pluginu je nutné si vytvořit vlastní, nejlépe vycházet z některého již existujícího pluginu.

7.2.1 Struktura pluginu

Během implementace bylo dbáno na použití doporučených postupů a moderního zápisu kódu. Podle toho byla vytvořena adresářová struktura.

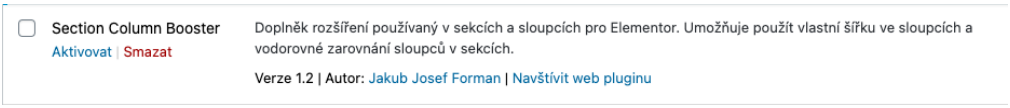
- V kořenovém adresáři je umístěn jediný PHP soubor, představující hlavní soubor pluginu. Spolu s ním jsou v kořenovém adresáři soubory typu *.txt*, *.md*, *.gitignore* a *composer.json*.
- Další soubory jsou rozděleny logicky do adresářů podle jejich významu a typu souboru. Jelikož tento plugin nevyužívá žádné CSS ani JS soubory, jsou zde pouze 4 adresáře.
    - Adresář includes – obsahuje soubory pro načítání dalších závislostí, jako jsou jazykové soubory a lokalizace, validaci částí WordPressu a serveru a obecnou třídu usnadňující komunikaci s nativním PHP API WordPress.
    - Adresář languages – obsahuje zdrojové soubory lokalizací ve formátu .pot, který lze vygenerovat příkazem skrze WP CLI (*wp i18n make-pot . languages/section-column-booster.pot --domain=section-column-booster*) a následně jej upravit pro rozdílné lokalizace.
    - Adresář src – je hlavním zdrojovým adresářem, ve kterém se nachází veškerý kód, který souvisí přímo s pluginem.

- Adresář lib – adresář externích knihoven, které se v kódu využívají.

Hlavní soubor nazvaný *section-column-booster* zavádí plugin do systému WP. Začátek souboru obsahuje hlavičku, která určuje základní náležitosti pluginu podle specifikací WP. V hlavičce je specifikovaný název, popis, verze, informace o autorovi, licence a informace k lokalizacím.

Composer

V kořenové složce se nachází soubor composer.json ve kterém jsou nastaveny základní knihovny a další nastavení pro správu balíčků třetích stran. Hlavní a jediná knihovna pro tento projekt je vlastní – *jayjay666/wp-requirements-checker* určena pro kontrolu a validaci pluginů, serveru a WordPressu.



Obrázek 26: Náhled zobrazení pluginu v administraci WP

7.2.2 Pomocná třída Loader

Tato třída zjednodušuje práci s WordPress Hooky a umožňuje jejich objektový zápis. Jde o rozšíření implementace práce s Hooky, kdy jsou všechny Hooky předem známy a zavolány až v jednom konkrétním bodě metodou `run()`. V kódu se rozlišují Hooky na dva typy stejně jako je tomu u WordPressu. Jde o Action a Filters, které je možné vložit místo klasické metody `add_action()` nebo `add_filter()` pomocí statických metod `Loader::addAction()` nebo `Loader::addFilter()`. Tento způsob zápisu zpřehledňuje kód a je více objektově orientovaný než původní zápis WordPressu.

V kódu se nejprve vytvoří požadavky na tyto Hooky výše zmíněnými metodami a metodou `run()` se zaregistrují do WP. Kód Hooků se zavádí na jednom místě což vede k lepší stabilitě a přehlednosti kódu.

7.2.3 Zavedení pluginu

V hlavním souboru je inicializován autoloader pro automatické načítání závislostí mezi zdrojovými soubory a zavedení pluginu, který se má spustit přes inicializační funkci.

```
if ( is_readable( __DIR__ . '/lib/autoload.php' ) ) {
        require __DIR__ . '/lib/autoload.php';
}

function runSectionColumnBooster() {

    $plugin = new SectionColumnBooster();
    $plugin->run();

}
runSectionColumnBooster();
```

7.2.4 Třída SectionColumnBooster

Soubor a stejnojmenná třída je hlavním zdrojovým kódem, který je jako ostatní soubory načítán v *autoload.php*. Při volání `new SectionColumnBooster()` je prvním bodem inicializace konstruktor, který provede validaci jestli je plugin možné aktivovat na aktuálním serveru a má vše potřebné pro svůj běh. Pokud při validaci údajů nebyl proces přerušen výjimkou, volají se metody pro lokalizace a rozšíření pro plugin Elementor.

```
public function __construct()
{
    RequirementValidator::requirementsValidate();
    $this->i18n();
    $this->loadElementorExtends();
}
```

Metoda `i18n()`, která je zavolána v konstruktoru registruje Hook skrze vlastní Loader metodu `LoadPlugiTextdomain` z objektu `i18n` při volání `plugins_loaded` do systému WP. Tento Hook se nevyvolává do doby, než je volán skrze `Loader::run()`.

```
public function i18n()
```

```
{
    $c = new i18n();
    Loader::addAction('plugins_loaded', $c, 'LoadPlugiTextdomain');
}
```

Druhou metodou zavolanou v konstruktoru je `loadElementorExtends()`, která inicializuje objekt pracující s rozšířením Elementoru. Důvod existence objektu Elementor je v organizaci a budoucím upravování kódu. Pokud by plugin nejenom rozšiřoval stávající Elementor Widgety, ale přidával vlastní, lze vytvořit v tomto objektu metody pro načtení nových Widgetů a udržet si kód čistý a logicky dělený.

```
public function loadElementorExtends()
{
    $elementor = new Elementor();
    $elementor->loadElementorExtends();
}
```

Metoda `run()` spouští `Loader::run()` pro zavedení všech Hooků. Bez této metody by nebylo možné spustit aktivní část pluginu, aby se projevil kód a rozšířil funkce Elementoru. Jde tedy o poslední metodu, která musí být zavolána před dokončením kódu.

Celou třídu SectionColumnBooster lze označit za jakýsi rozcestník, který organizuje strukturu kódu a dělí ho na logické části. Takto rozdělená hlavní třída se vyplatí zejména u větších pluginů, kdy se načítá velké množství Hooků a ty nejsou navzájem provázané s jedním konkrétním bodem a zasahují do více částí WordPressu nebo dílčích pluginů.

7.2.5 Třída Elementor

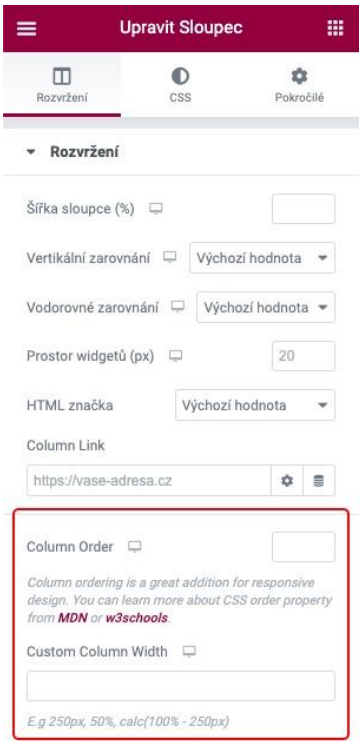
Tato třída má za úkol dělit kód na část používanou výhradně s Elementorem. V jejím kódu lze nalézt jednu metodu pro registraci Hooků, které se vážou na již existující Widgety z pluginu Elementor. Metoda podobně jako `i18n` využívá třídu `Loader` a přidává do ní dvě akce.

```
public function loadElementorExtends()
```

```
{
    // Přidá rozšířené možnosti sloupcům
    $hook = 'elementor/element/column/layout/before_section_end';
    Loader::addAction($hook, new Column(), 'addColumnControls');

    // Přidá rozšířené možnosti sekcím
    $hook = 'elementor/element/section/section_layout/before_section_end';
    Loader::addAction($hook, new Section(), 'addSectionControls');
}
```

První akce přidává rozšířené funkce z vlastního objektu Column, které jsou obsaženy v její metodě `addColumnControls()` a registruje je k Hooku `elementor/…/column/layout/before_section_end`, který označuje konec sekce rozvržení v záložce layoutu (Rozvržení) při editaci Widgetu sloupce.



Obrázek 27: Vizuální náhled na pozici podle Hooku

Druhá akce přidává rozšířené funkce z vlastního objektu Section, které jsou obsaženy v její metodě `addSectionControls()` a registruje je k Hooku `elementor/…/section/section_layout/before_section_end`, který označuje stejně jako u první akce konec sekce rozvržení v záložce layoutu (Rozvržení) při editacy Widgetu sekce.

7.3 Rozšíření existujících Widgetů

Třídy upravující Widgety se nacházejí v adresáři src/Extend/Elements a obsahují jednu základní metodu pro modifikaci již existujících Widgetů. Každá třída upravuje Widget pro ni specifický a není tak jednotná třída upravující více Widgetů naráz.

Metody `addColumnControls()` a `addSectionControls()` jsou vstupními metodami, které se volají při registraci Hooků a předává se do těchto metod jako parametr třída `Element_Column` v případě sloupců a `Element_Section`, které jsou potomky třídy `Element_Base`. Tato třída umožňuje registrovat další funkcionalitu a vlastnosti Widgetů nebo již existující funkce změnit. Pro přidání dalších ovládacích prvků se využívá metoda `add_control()` obsažená v třídě `Element_Base`. Tato metoda nenabízí responzivní zobrazení prvků, které by bylo možno nastavit pro různé šířky zařízení. Pro tyto případy existuje `add_responsive_control()`, která provede potřebné akce pro zaznamenání dat v responzivních hodnotách.

7.3.1 Upravení sloupců

U Widgetu sloupců je implementována možnost řazení sloupců skrze otočení pořadí při responzivním designu. Tato cesta však není mnohdy dostačující, a proto zde implementuji vlastní rozšíření, které integruje nový ovládací prvek, je plně responzivní a umožní nastavit pořadí sloupce podle jeho čísla stejně jako v CSS Flex.

V těle metody `addColumnControls()` je volána `add_responsive_control()` s parametry ID a Argumenty jako pole typu key:value. Jako ID jsem zvolil prefix + výstižný název – `_scb_column_order` a některé argumenty:

- label – popisný štítek nového prvku,
- separator – oddělovač a jeho umístění,
- typ – typ inputu, který má být vytvořen,
- selectors – definované CSS proměnné, které se mění v závislosti na hodnotě inputu,
- options – definování hodnot pole SELECT,

- description – popis prvku s vysvětlivkami.

Významy klíčů uvnitř pole argumentů:

- `label` je titulek pro input, který je zobrazen v panelu editace. Jeho hodnota je nastavena na `__('Column Order', 'section-column-booster')`, tento zápis umožňuje překlad za pomocí funkce `__()`, která obsahuje základní parametr text a Text Domain, který je jednotný v celém vytvářeném pluginu a všech jeho souborech.
- `separator` označuje pozici oddělovače a má přípustné hodnoty `before`, `after`, `none`, a je nepovinný.
- `type` určuje typ prvku, který má být zobrazen. Pro nastavení hodnoty se používá třída `Controls_Manager`, která obsahuje desítky typů polí, které lze využít při vytváření ovládacího prvku. U prvku, který se má zabývat pouze číselnými hodnotami jsem zvolil `Controls_Manager::NUMBER`, který definuje typ pole jako číselné hodnoty a zároveň i jeho validaci.
- `description` je dodatečný popis tohoto pole, který uživatelům může předat nápovědu, která jim řekne, co ovládací prvek dělá a jak funguje. V kódu je uveden překlad s vložením dvou odkazů, které se nepřekládají.
- `selectors` předává CSS hodnoty z tohoto prvku do CSS a zapisuje se jako pole typu key:value. Pro zápis do konkrétního prvku spojeného s aktuálním Widgetem se používá hodnota `{{WRAPPER}}`. Tato hodnota označuje právě základní HTML element Widgetu. Pro přesné specifikování a přepsání původních hodnot lze přidat i CSS třídy k tomuto prvku náležící. Výsledný klíč `{{WRAPPER}}.elementor-column` označuje konkrétní prvek a kompilátorem CSS kódu bude nahrazen za validní CSS `.elementor-element.elementor-element-aa40730.elementor-column`. Hodnota tohoto klíče je poté string ve tvaru CSS přípustných hodnot, v případě pozice `-ms-flex-order:{{VALUE}}; order: {{VALUE}};` Hodnota `{{VALUE}}` je získaná hodnota z ovládacího prvku.

Druhý nedostatek Elementoru je v nastavování šířky sloupce, která lze nastavit pouze v % hodnotách 0.1–99.9 %. Toto omezení je dostačující u většiny designů, u designů které se ale zaměřují na tzv. Pixel Perfect je téměř nemožné dosáhnout přesné šířky v PX za pomocí %. Pomocí druhé funkce rozšiřuje sloupce, lze nastavovat CSS width libovolné hodnoty a dosáhnout tím potřebných specifikací šířky sloupce.

Parametry funkce `add_responsive_control()` jsou částečně stejné jako v případě přidávání CSS order, jako ID je použito `_scb_column_width` a tyto klíče jsou upravené:

- `type` – jde upravené na hodnotu `Controls_Manager::TEXT`, aby bylo možné zadat veškeré textové výrazy a bylo tak možné zadat hodnoty jako jsou PX, EM, %, calc() nebo fit-content.
- `selectors` – kde jako klíč je využíván stejný zápis jako u CSS order, ale hodnota je nastavena na `width: {{VALUE}};`.

Tyto vlastní ovládací prvky jsou ukázkou, jak jednoduše lze upravovat CSS hodnoty u již existujících Widgetů v Elementoru. Mohou posloužit jako základní šablona pro další úpravy Widgetů v budoucnu a ukázka studentům informačních technologií ke zlepšování jejich dovednosti a zkušeností v budoucí praxi.

7.3.2 Upravení sekcí

Sekce jsou podobně jako sloupce dostatečné, avšak při návrhu některých designů se designér setká s drobnými nedostatky. Nejzásadnějším nedostatkem je zarovnání obsahu (sloupců) sekce, který v základní ani PRO verzi Elementoru implementován není. Sekce, které jsou uživateli k dispozici mají základní implementaci CSS Flex. Jednou z možných hodnot CSS Flex je `justify-content`. To definuje zarovnání podél hlavní osy. Pomáhá distribuovat zbylé volné místo, když jsou buď všechny položky flexu na řádku nepružné, nebo jsou flexibilní, ale dosáhly své maximální velikosti. Také nabízí určitou kontrolu nad zarovnáním prvků, když přetékají řádek. Možné hodnoty jsou:

- `flex-start (výchozí)` – prvky jsou seřazeny na začátku směru flexu.
- `flex-end` – prvky jsou seřazeny ke konci směru flexu.
- `Center` – prvky jsou vycentrovány podél osy.
- `space-between` – prvky jsou rovnoměrně rozloženy v řadě s rovnoměrnými mezerami mezi prvky. První prvek je na počátečním řádku, poslední prvek na koncovém řádku.
- `space-around` – prvky jsou rovnoměrně rozloženy v řadě se stejným prostorem kolem nich. U tohoto nastavení si lze všimnout, že vizuálně nejsou mezery stejné, protože všechny položky mají stejný prostor na obou stranách. První položka bude mít jednu jednotku prostoru proti okraji kontejneru (sekce), ale dvě jednotky prostoru

mezi další položkou, protože následující položka má své vlastní mezery, které platí spolu s dalšími mezerami.

- `space-evenly` – prvky jsou rozloženy tak, aby mezery mezi libovolnými dvěmi položkami (a mezerou po okrajích) byly stejné.

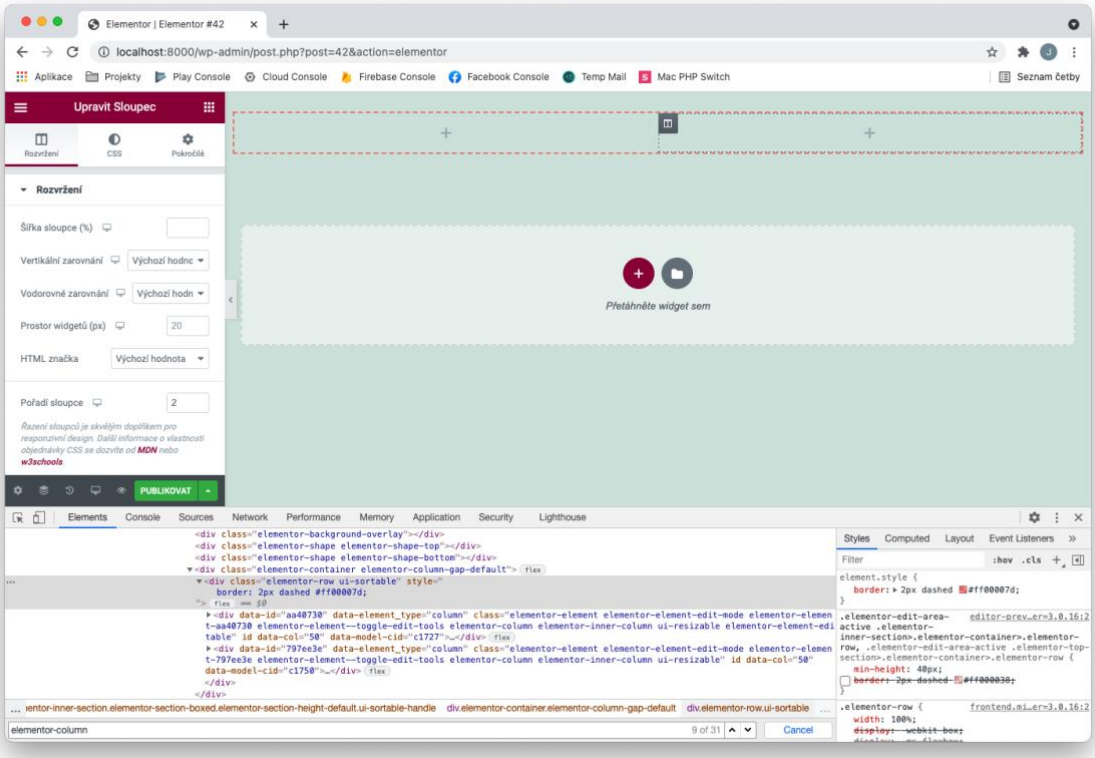
Vlastností je více, ale další jsou vázané na další nastavení CSS Flex, které v pluginu neimplementuji. Implementace těchto hodnot je dostačující a nabízí velkou flexibilitu.

Pro úpravu sekcí je vytvořená vlastní třída `Section`, která v sobě implementuje jednu metodu, ve které jsou upravovány funkce již existujících sekcí. Vložené argumenty do metody `add_responsive_control()` jsou podobné jako v případě upravování sloupců. Jako ID je použito `_scb_section_horizontal_align`, které specifikuje ovládací prvek a pole atributů. Pole atributů je nastaveno mírně odlišným způsobem, v závislosti na použitém typu ovládacího prvku.

Významy klíčů uvnitř pole argumentů pro nový responzivní ovládací prvek jsou následující:

- `label` je nastaven podobným způsobem jako v případě ovládacích prvků u sloupců `__('Horizontal align', 'section-column-booster')`.
- `type` je nastaven na `Controls_Manager::SELECT`, který definuje typ pole jako HTML input select.
- `selectors` předává CSS hodnoty z pole `option` (klíče) prvku do CSS. Pro aplikování CSS je nutné použít zanořený prvek, který obsahuje řádek (rodiče) elementů sloupců. Ten lze získat pomocí `{{WRAPPER}} .elementor-row`. Tento zápis označuje jakýkoliv prvek uvnitř hlavního prvku sekce. Pro zjištění tohoto prvku jsem využil inspektor HTML elementů, který lze nalézt v každém prohlížeči mezi vývojovými nástroji. Pro aplikování vlastního CSS je využita hodnota `justify-content: {{VALUE}};`.

Výsledným označením prvku pomocí `{{WRAPPER}} .elementor-row` je získán zvýraznění HTML element zobrazený na obrázku 24. Tento element je ručně zvýrazněný červeně-přerušovaným okrajem se šířkou 2px.



Obrázek 28: Získání CSS Class z inspektoru HTML elementů

8 TESTOVÁNÍ A POUŽITÍ VLASTNÍHO PLUGINU

Před tím, než se plugin zveřejní do knihovny WordPress nebo zpřístupní jeho zdrojový kód na GIT ke stažení, měl by projít kontrolou na možné chyby. Tato kapitola se nezabývá automatizovaným testováním, které je v tomto případě velmi komplexní, nýbrž se zabývá testováním uživatelským a kontrolou bezpečnosti.

8.1 Bezpečnost

V teoretické části zmiňuji základy bezpečnosti pluginů, která platí pro všechny vyvíjené pluginy. Jelikož vše, co je v ukázkovém pluginu vytvořeno je napojeno přímo na Elementor, který řeší velkou část oprávnění za vývojáře a není použit žádný vlastní kód přístupu do databáze ani JavaScriptové volání PHP skriptu, není potřeba v tomto specifickém případě tuto část řešit.

Pokud by měl mít plugin omezení pouze pro uživatele s určitým oprávněním, musí vývojář dbát na to, aby byly zachovány jeho funkce. Pokud by bylo omezení řešeno stylem, že se nespustí Hooky, které registrují funkce pro rozšíření stávajících prvků do Elementoru, nastane problém, kdy Elementor již dříve existující data z tohoto rozšíření nebude vkládat do CSS a HTML kódu. Uživatel s nižším oprávněním, který by neměl k tomuto rozšíření přístup, neuvidí dříve upravený kód pomocí vlastního pluginu, a následně tento kód přepíše, čímž bude dříve upravený kód Elementorem „zahozen“. Tento problém by se ve většině případů řešil formou nastavení atributu `disable` nebo `readonly` k HTML elementu, který nesmí být editovatelný, ale čitelný. Avšak Elementor tuto možnost nenabízí a omezit funkcionalitu jednotlivých inputů pro uživatele s rozdílným oprávněním momentálně nelze. Tým Elementoru podle Issues v síti GitHub na možnosti vypínat a zapínat prvky pomocí atributů pracuje a v budoucích aktualizacích by měla tato funkce být dostupná ve všech Widgetech.

8.2 Testování a použití

V rámci testování byly prováděny pouze uživatelské testy a podpora prohlížečů s WebKit a Chromium, jelikož jsou v současné době nejpoužívanějšími typy prohlížečových renderovacích jader. Ze získaných výsledků testů jsou poté vyhodnoceny klady a zápory implementovaného pluginu.

8.2.1 Manuální testování

Implementovaný plugin je možné otestovat ručně tím, že budou vyzkoušeny jeho funkce. V průběhu vytváření se tyto funkce testují taktéž, ale dohromady s ostatními částmi Elementoru, až při dopsání kódu. Ruční testování spočívá ve vyzkoušení všech přípustných hodnot z testovací skupiny a jejich vyhodnocení.

Testování sloupců

U sloupců bylo přidáno rozšíření pro nastavení vlastní šířky sloupce v libovolné hodnotě typu TEXT. Při implementaci pluginu bylo dbáno na zachování nastavení šířky v % původní cestou. Pokud uživatel zadá šířku sloupce do původního pole, bylo zjištěno, že vlastní šířka sloupce nemá na upravení originální % hodnoty vliv a stává se tedy nepoužitelnou.

Pro zadání testovacích hodnot jsou uvedeny tabulky reprezentující zadané hodnoty a fyzické vykreslení z těchto hodnot se dvěma sloupci nebo jedním sloupcem. **Tabulka 1** se zaměřuje na testování hodnot se dvěma sloupci, kdy jsou hodnoty nastavené pouze v jednom ze sloupců, výsledky zjištění jsou zaznamenány do tabulky spolu se zadanou hodnotou. **Tabulka 2** je tabulka zobrazující reálné hodnoty při existenci pouze jednoho sloupce uvnitř sekce. **Tabulka 3** zobrazuje hodnoty zadané do pořadí sloupců v případě existence 3 sloupců. Tyto hodnoty byly také testovány na 2 sloupcích a chování bylo totožné s chováním ve více sloupcích.

Tabulka 1: Testovací hodnoty pro vlastní šířku sloupce – dva sloupce

Zadaná hodnota	**Výsledné vykreslení**
0	Vykreslí sloupec velikosti 25 x 50 PX v případě zobrazení v editoru, na stránce je sloupec nevykreslen i v případě, že sloupec obsahuje jiný prvek.
auto	Při prázdném sloupci se chová stejně jako hodnota 0, pokud je ve sloupci libovolný Widget, šířka sloupce se přizpůsobí šířce hlavního prvku – pokud vnitřní Widget není obalen jiným prvkem, který nemá přímo upravitelnou šířku.
0%	Stejné chování jako v případě hodnoty 0.
20%	Nastaví šířku sloupce na 20 % šířky rodičovského prvku.
100%	Nastaví šířku sloupce na maximální možnou šířku do 100 %.
200%	Nastaví šířku sloupce na maximální možnou šířku do 200 %. Zobrazení druhého nenastaveného sloupce je menší jako v případě 100 %.
initial	Stejné jako v případě hodnoty auto.
inherit	Podobné jako v případě initial s podobným rozdílem jako u rozdílů hodnot 100 % a 200 %.
fit-content	Stejné jako u inherit.
content-box	Žádná změna – jde o nepřípustnou hodnotu CSS Width.
100px	Nastaví šířku na 100 PX.
3000px	Nastaví šířku na maximální přípustnou hodnotu do 100 % z původních 3000 PX.
20em	Nastaví šířku na 400 PX.
calc(100% - 100px)	Nastaví šířku na 100 % - 100 PX (pokud je přípustná) jinak na nejbližší přípustnou.
asdf	Žádná změna – jde o nepřípustnou hodnotu CSS Width.

Tabulka 2: Testovací hodnoty pro vlastní šířku sloupce – jeden sloupec

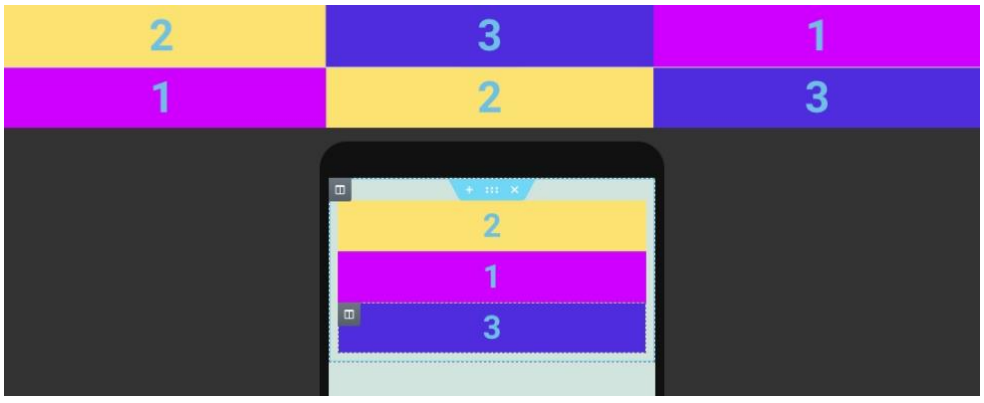
Zadaná hodnota	**Výsledné vykreslení**
0	Vykreslí sloupec velikosti 25 x 50 PX v případě zobrazení v editoru, na stránce je sloupec nevykreslen i v případě, že sloupec obsahuje jiný prvek.
auto	Při prázdném sloupci se chová stejně jako hodnota 0, pokud je ve sloupci libovolný Widget, šířka sloupce se přizpůsobí šířce hlavního prvku – pokud vnitřní Widget není obalen jiným prvkem, který nemá přímo upravitelnou šířku.
0%	Stejné chování jako v případě hodnoty 0, pokud obsahuje vnitřní Widget, je zobrazena základní šířka 25 x 50 PX i v případě zobrazení na stránce.
20%, 100%, 200%	Stejná reakce jako v případě více sloupců.
Initial, inherit, fit-content	Stejná reakce jako v případě více sloupců.
content-box	Žádná změna – jde o nepřípustnou hodnotu CSS Width.
100px	Nastaví šířku na 100 PX.
3000px	Nastaví šířku na maximální přípustnou hodnotu do 100 %.
20em	Nastaví šířku na 400 PX.
calc(100% - 100px)	Nastaví šířku na 100 % - 100 PX.
asdf	Žádná změna – jde o nepřípustnou hodnotu CSS Width.

Změny uváděné v tabulkách jsou testované skrze ruční zadávání a může se objevit chyba měření podle použitého monitoru a jeho rozlišení. Hodnoty jsou však platné pro FullHD rozlišení a jde je tedy fyzicky zaznamenat lidským zrakem a fyzicky zjistit z inspektoru HTML elementů. Tyto vlastnosti fungují i v responzivním designu.

Tabulka 3: Testovací hodnoty pro pořadí sloupce – 3 sloupce

Zadaná hodnota	**Výsledné vykreslení**
1, 2 a 3 sloupec **Všechny sloupce mají nastavené hodnoty svého pořadí (4, 2, 3)**	Sloupec č. 1 má hodnotu 4, sloupec č. 2 má hodnotu 2, sloupec č. 3 má hodnotu 3. Sloupec č. 1 je zobrazen až po sloupci č. 3 který se zobrazuje na druhém místě skrze platnost 4>3.
1 sloupec – 0 **Ostatní sloupce jsou bez hodnoty**	Pokud je sloupci nastavena hodnota jiná než prázdná, automaticky se zařadí na konec. Pro to, aby byl zařazen jako druhý, je třeba ostatním sloupcům přidat také hodnoty
1, 2 a 3 sloupec – mobil **Všechny sloupce mají nastavené hodnoty svého pořadí (2, 1, 3)**	Sloupce v responzivním designu reagují stejně jako sloupce v klasickém návrhu bez responzivního designu. **Sloupec 1** bude tedy na místě, protože platí 2>1. **Sloupec 2** bude na prvním místě, protože je platné 1<2. **Sloupec 3** bude na třetím místě, protože platí 3>2 kde hodnota 2 je ze **sloupce 1** sloupce.

Pro tabulku 3 přikládám ilustrační příklad vykreslení sloupců uvnitř sekce, který znázorňuje pořadí sloupců za pomocí barvy a nadpisu s číslem označující původní pozici sloupce při nenastavených hodnotách.



Obrázek 29: Testovací hodnoty pro pořadí sloupce – náhled

8.2.2 Podpora prohlížečů

Doba, kdy bylo nutné podporovat více než 3 hlavní prohlížeče už je minulostí, avšak zavedení určitých CSS atributů bývá stále u některých prohlížečů implementována pozdě, nebo nepřesně. Některé prohlížeče využívají vlastní implementaci a je nutné tuto implementaci zaznamenat do CSS kódu. Testovány byly tři prohlížeče Google Chrome, Apple Safari a Firefox. Pro testování byly vybrány tyto prohlížeče skrze jejich masivní rozšíření a míru implementace CSS atributů. Prohlížeč Google Chrome využívá jádro Chromium, které pohání další řadu prohlížečů jako je třeba Microsoft Edge, Opera, Vivaldi nebo Brave. Apple Safari má vlastní jádro (WebKit), který je dostupné na Apple platformě. [31] Firefox využívá vykreslovací jádro Gecko. Všechny tři prohlížeče byly testovány v posledních vydaných verzích v době psaní této diplomové práce:

- Google Chrome – 90.0.4430.85 (arm64)
- Safari – 14.0.3 (16610.4.3.1.4) (arm64)
- Firefox – 83 (emulace: Rosseta2 – Intel Base x86)

Plugin, který byl implementován nemá v těchto prohlížečích žádný problém se zobrazením, nebo vykreslením hodnot a je tedy plně podporován.

8.2.3 Zhodnocení výsledků

Z výsledků uživatelského testování vyplývá, že implementovaný plugin úspěšně splňuje specifikaci CSS Flex `justify-content` a `order` tak, jak bylo účelem pluginu zamýšleno. Instalací toho pluginu do libovolné webové prezentace postavené na WordPress a Elementor získá designér možnost přizpůsobovat flexibilně obsah sekcí, sloupců a jejich formátování, které v základní verzi Elementor neimplementuje.

Mezi další klady tohoto pluginu lze zařadit kompatibilitu s budoucími aktualizacemi WordPress i Elementor a Elementor Pro pluginů. Toho bylo možné dosáhnout využitím převážně funkcí, které byly nabízeny systémem WordPress a pluginem Elementor v době psaní této diplomové práce. Dalšími klady je jednoduchá instalace, kdy plugin stačí pouze nainstalovat a aktivovat pro jeho funkčnost, nepřímé spojení s databází a využití zabezpečení vestavěného v pluginu Elementor.

ZÁVĚR

Tato práce si kladla za cíl seznámit čtenáře s redakčním systémem WordPress, jeho základními funkcemi, možnou customizací skrze plugin Elementor a vytvoření rozšíření pro tento plugin a možný nástin problematiky ve výuce webových stránek na středních školách s návazností na praktické využití.

V úvodní části je rozebrán systém WordPress a jeho požadavky, šablony a vzhledy, rozšíření, základní nastavení tohoto systému. Rozšíření je již probíráno podrobněji a jsou objasněné základní pojmy Hook, bezpečnost, struktura pluginu, DocBlock a doporučení, jak vytvářet vlastní pluginy. Dalším tématem je představení pluginu Elementor, kde je vysvětleno jeho nastavení, funkce a možnost použití a nasazení na webu. Spolu s použitím jsou vysvětleny a rozebrány funkce jako jsou šablony, jejich typy a využití, globální nastavení designu, barev a rozložení a orientace v Elementoru. V kapitole Elementor se práce zabývá zabezpečením designu od nechtěných úprav neoprávněnými uživateli a jejich minimalizací. Toto zabezpečení je porovnáváno se standartním zabezpečením WordPress (rolí) a vysvětluje základní rozdíly jednotlivých pohledů obou systémů.

Kromě obecného představení a seznámení s Elementorem jsou rozebrány základy Widgetů, jejich editace a manipulace s nimi, knihovna již existujících šablon pro rychlý začátek nebo vlastní šablony. V práci nechybí ani základní srovnání systémů Elementor a Elementor Pro a k čemu lze každý ze systémů použít.

Praktická část práce se zabývá pohledem na možnost zlepšení výuky pro studenty středních škol více navazující formou, která by mohla vyvolat zájem studovat dále informatické obory. Důvodem proč, vyvíjet vlastní plugin pro WordPress nebo Elementor. Návrh webu ve WordPress, nutné kroky a prozkoumání částí systému. Základní návrh designu stránky pro archivy pomocí Elementoru a tvorbu rozšiřujícího pluginu pro Elementor.

SEZNAM POUŽITÉ LITERATURY

[1] CO JE TO WORDPRESS A JAKÉ JSOU JEHO VÝHODY A NEVÝHODY? [online]. 2016 [cit. 2021-04-05]. Dostupné z: https://www.wedesin.cz/webove-stranky/co-je-to-wordpress-a-jake-jsou-jeho-vyhody-a-nevyhody

[2] WordPress Powers 39.5% of All Websites [online]. 2021 [cit. 2021-04-05]. Dostupné z: https://www.searchenginejournal.com/wordpress-powers-39-5-of-all-websites/391647/

[3] Requirements: To run WordPress we recommend your host supports [online]. 2021 [cit. 2021-04-05]. Dostupné z: https://wordpress.org/about/requirements/

[4] Theme Development [online]. Wordpress.org [cit. 2021-04-20]. Dostupné z: https://codex.wordpress.org/Theme_Development

[5] HEDENGREN, T. D. Smashing WordPress: Beyond the Blog. John Wiley & Sons, 2014. ISBN 978-1-118-60075-7.

[6] Template Hierarchy [online]. Wordpress.org [cit. 2021-04-20]. Dostupné z: https://developer.wordpress.org/themes/basics/template-hierarchy/

[7] BONDARI, B a E GRIFFITHS. WordPress 3 Plugin Development Essentials. Packt Publishing, 2011. ISBN 978-1-849513-52-4.

[8] What is Event-Driven Architecture? [online]. Solace.com [cit. 2021-04-20]. Dostupné z: https://solace.com/what-is-event-driven-architecture/

[9] How to Easily Add Custom CSS to Your WordPress Site [online]. Wpbeginner.com [cit. 2021-04-20]. Dostupné z: https://www.wpbeginner.com/plugins/how-to-easily-add-custom-css-to-your-wordpress-site/

[10] WILLIAMS, Brad, David DAMSTRA a Hal STERN. Professional WordPress: design and development. Second edition. Indianapolis, IN: John Wiley & Sons, [2013]. Wrox professional guides. ISBN 978-1-118-44227-2.

[11] WILLIAMS, Brad. *Professional WordPress Plugin Development*. Indianapolis, IN: Wiley Publishing, [2011]. ISBN 978-0470916223.

[12] *WordPress Nonces* [online]. Wordpress.org [cit. 2021-04-20]. Dostupné z: https://codex.wordpress.org/WordPress_Nonces

[13] Best Practices [online]. Wordpress.org [cit. 2021-04-20]. Dostupné z: https://developer.wordpress.org/plugins/plugin-basics/best-practices/

[14] Theme Style – Global Settings [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/theme-style-global-settings/

[15] *Full Comparison - Pro VS Free* [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/pro-vs-free/

[16] The #1 Free WordPress Page Builder [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/features/page-builder/

[17] Settings [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/settings/

[18] Tools [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/tools/

[19] Hello [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/hello-theme/

[20] How to Save, Import, & Export Templates from the Elementor Template Library [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/template-library/

[21] Elementor's Theme Builder [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/elementors-theme-builder/

[22] Site Settings [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/site-settings/

[23] Navigator [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/navigator/

[24] Document Settings [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/document-settings/

[25] Revision History, Undo and Redo [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/revision-history-undo-and-redo/

[26] Mobile Responsive Editing [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/mobile-editing/

[27] Widgets [online]. Elementor.com [cit. 2021-04-20]. Dostupné z: https://elementor.com/help/widgets/

[28] HTML Standard [online]. Whatwg.org [cit. 2021-04-20]. Dostupné z: https://html.spec.whatwg.org/multipage/introduction.html

[29] PHP /základy/ [online]. Whatwg.org [cit. 2021-04-20]. Dostupné z: https://www.tvorba-webu.cz/php/

[30] What Is Composer for PHP and How to Install It [online]. Tutsplus.com [cit. 2021-04-20]. Dostupné z: https://code.tutsplus.com/tutorials/what-is-composer-for-php-and-how-to-install-it--cms-35160

[31] Browser & Platform Market Share: March 2021 [online]. W3Counter.com [cit. 2021-4-20]. Dostupné z: https://www.w3counter.com/globalstats.php

[32] BEDROŠOVÁ, Marie, Renata HLAVOVÁ, Hana MACHÁČKOVÁ, Lenka DĚDKOVÁ a David ŠMAHEL. EU KIDS ONLINE IV v České republice: Zpráva z výzkumu na základních a středních školách [online]. Masarykova univerzita, Brno, 2018 [cit. 2021-5-1]. Dostupné z: https://irtis.muni.cz/media/3115505/eu_kids_online_report.pdf. Výzkum. MUNI.

[33] WORDPRESS. File:WordPress blue logo.svg. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-5-1]. Dostupné z: https://commons.wikimedia.org/wiki/File:WordPress_blue_logo.svg

[34] *Úvod do kontejnerů a Docker* [online]. 2020 [cit. 2021-5-2]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/architecture/containerized-lifecycle/introduction-to-containers-and-docker

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

WP	WordPress.
CMS	Content Management System – Systém pro správu obsahu.
PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P 1: OFICIÁLNÍ GITHUB REPOZITÁŘ OBSAHUJÍCÍ ZDROJOVÉ KÓDY

Tento repozitář je umístěn na platformě GitHub a je jej možné rozšiřovat a dále čerpat.
https://github.com/JayJay666/section-column-booster