Vývoj mapy knihovního fondu

Bc. Tomáš Doležal

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav počítačových a komunikačních systémů

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš Doležal**
Osobní číslo:	**A18336**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**Kombinovaná**
Téma práce:	**Vývoj mapy knihovního fondu**
Téma práce anglicky:	**The Development of a Library Collection Map**

Zásady pro vypracování

1. Proveďte literární rešerši na dané téma.
2. Definujte požadavky na systém a popište workflow.
3. Navrhněte technické řešení ve formě webové aplikace.
4. Zdůvodněte výběr jednotlivých komponentů technického řešení.
5. Realizujte a otestujte výsledné technické řešení ve spolupráci s uživatelem.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. DRASNER, Sarah. SVG animations: from common UX implementations to complex responsive animation. Boston: O'Reilly, 2017. ISBN 9781491939703.
2. BELLAMY-ROYDS, Amelia, Kurt CAGLE a Dudley STOREY. Using SVG with CSS3 and HTML5: vector graphics for web design. Sebastopol, CA: O'Reilly, [2018]. ISBN 9781491921975.
3. SVG ON THE WEB [online]. Breaking Borders, 2019 [cit. 2019-11-19]. Dostupné z: https://svgontheweb.com/
4. SKLAR, David. PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Brno: Zoner Press, 2018, 368 s. Encyklopedie Zoner Press. ISBN 9788074133633.
5. RICHARDSON, Leonard a Michael AMUNDSEN. RESTful Web APIs. Sebastopol: O'Reilly, 2013, xxviii, 373 s. ISBN 9781449358068.

Vedoucí diplomové práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Konzultant diplomové práce: **Ing. Ivan Masár**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **13. prosince 2019**
Termín odevzdání diplomové práce: **29. května 2020**

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

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/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- 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/bakalářské 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 13.8.2020

Tomáš Doležal, v. r.
podpis diplomanta

ABSTRAKT

Cílem diplomové práce je vytvořit systém pro správu mapy knihovního fondu. Vývoj systému probíhal na open source platformě WordPress. Výsledkem práce je reálný systém, který se používá na hledání knih podle signatury v budově Knihovny UTB. Výstup ze systému je ve formě SVG mapy patra, kde podle žádané signatury se vyznačí regál a patro, v kterém se kniha nachází.

Klíčová slova: UTB, WordPress, SVG, PHP, HTML, JavaScript, mapa, knihovna, plugin

ABSTRACT

The aim of the diploma thesis is to create a system for managing the library collection map. The system was developed on the open source WordPress platform. The result of the work is a real system, which is used to search for books by signature in the UTB Library building. The output from the system is in the form of an SVG map of the floor, where according to the required signature, the shelf and the floor in which the book is located are marked.

Keywords: WordPress, SVG, PHP, HTML, JavaScript, map, library, plugin

Chtěl bych poděkovat vedoucímu diplomové práce doc. Ing. Jiřímu Vojtěškovi, PhD. za pomoc s vedením práce, ochotu a trpělivost. Dále děkuji zaměstnancům Knihovny UTB, zejména Ing. Davidu Janulíkovi, za uvedení do problematiky knihovnického systému.

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

OBSAH

ÚVOD

Tato diplomová práce se zabývá vývojem mapy knihovního fondu v rámci webové aplikace. Z budoucích nákladových důvodů je dobré se především vyhnout složitým technickým řešením a využíváním externích poskytovatelů webových služeb či specializovaného personálu. Obzvláště malé a střední společnosti, jakož i soukromé osoby, chtějí sami udržovat a spravovat svůj webový obsah, v našem případě správu mapy knihovního fondu. Aby bylo možné najít kompromis mezi dostupným rozpočtem a požadovanými službami, tak mnoho vývojářů webových aplikací používá pro implementaci systém pro správu obsahu (Content Management System – CMS) s otevřeným zdrojovým kódem. Protože na trhu je velké množství takových systémů, tak každého napadne otázka, který jen ten nejlepší?

Pro můj projekt jsem si vybral redakční systém WordPress, protože s ním mám bohaté zkušenosti. Má přehledné, uživatelsky přívětivé rozhraní pro správu, ale hlavně je zde možnost jednoduše vyvíjet a přidávat pluginy, což je jeden z hlavních záměrů mé práce.

Jak jsem už zmiňoval, budu se zabývat vývojem systému pro správu mapy knihovního fondu, který bude řešen formou pluginu na redakčním systému WordPress. Mezi hlavní body práce bych zařadil vytvoření mapy jednotlivých pater knihovny ve vektorové grafice, v našem případě ve formátu SVG a následný vývoj pluginu, kde se daná mapa přidá a který bude následně na návštěvníkem viditelné části webové aplikace vybarvovat daný regál, do kterého se vybraná kniha zařadí podle signatury. Tato aplikace bude ještě napojená na knihovnický systém, kde se po zmáčknutí tlačítka u dané knihy zobrazí ve vyskakovacím okně již zmiňovaná mapa a informace o knize.

I. TEORETICKÁ ČÁST

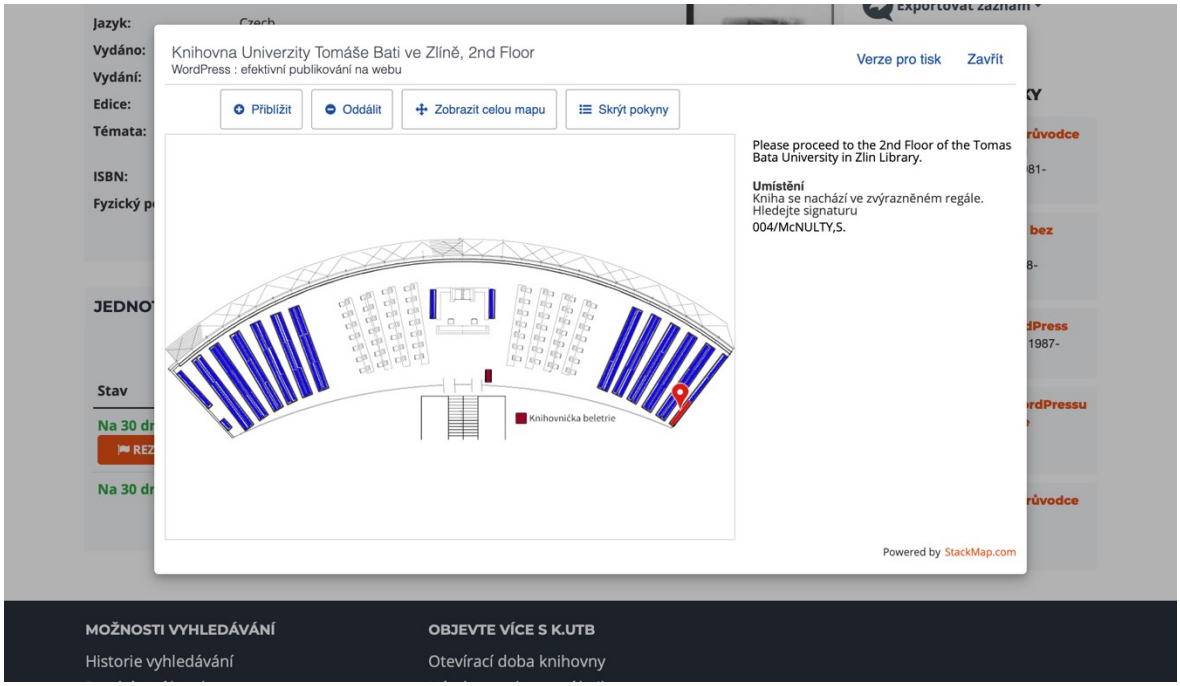
1 MAPY KNIHOVNÍHO FONDU

Rešerše mapy knihovního fondu probíhala prohledáváním katalogů knihoven s následným zobrazením dostupné knihy a zda má volbu zobrazení v regálu nebo na mapě. Jako seznam všech knihoven posloužila webová aplikace „Knihovny v České republice“ [1], která je pravidelně aktualizována a má i přímo odkazy na katalogy knihoven.

V České republice systém mapy knihovního fondu z velkých veřejných, vědeckých a univerzitních knihoven, které mají na webové aplikaci katalogu u detailu knihy možnost zobrazení mapy s umístěním knihy, používají z výsledků rešerše pouze tři knihovny. Jedná se o Knihovnu UTB, která používá hotové řešení, poté Národní technická knihovna a Krajská knihovna Františka Bartoše ve Zlíně, které používají vlastní řešení.

Knihovna UTB

Zlínská Knihovna UTB používá systém od americké firmy StackMap. StackMap se implementuje na knihovnický systém Knihovny UTB (https://vufind.katalog.k.utb.cz), kde se po stisknutí tlačítka „Ukázat v regálu“ zobrazí vyskakovací okno s názvem signatury a patra, v kterém se kniha nachází. Mapu můžeme přibližovat i oddalovat (Obrázek č. 1).



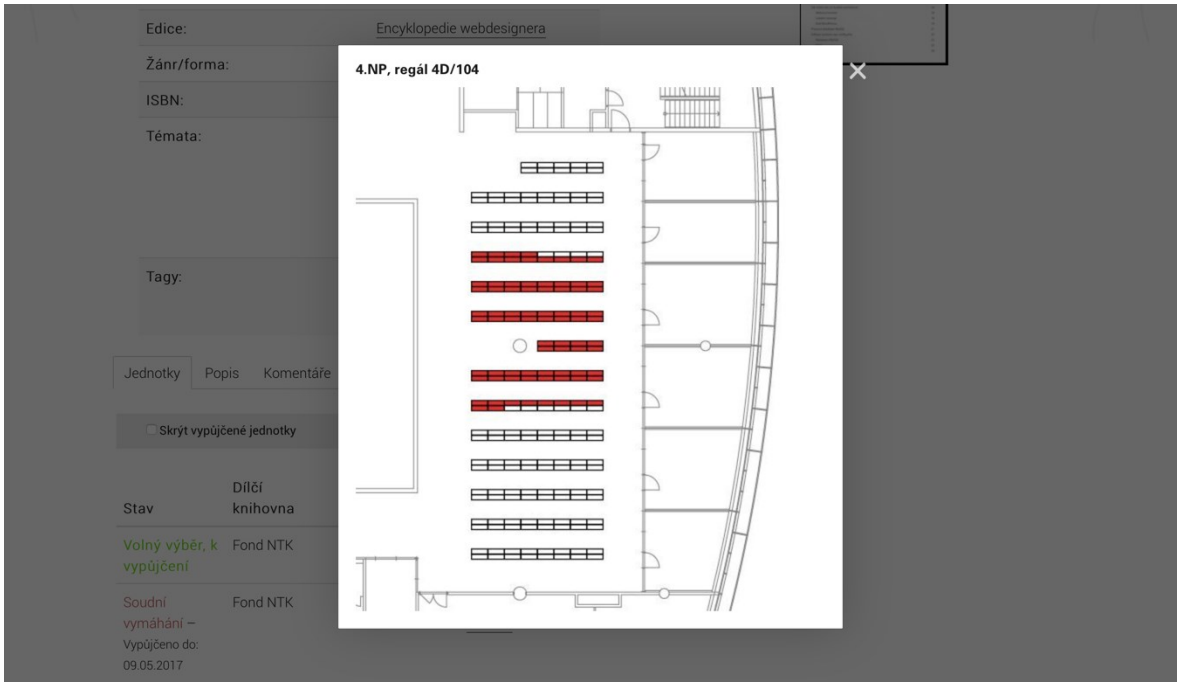
Obrázek č. 1 – Mapa knihovního fondu Knihovny UTB od firmy StackMap

Mezi hlavní výhody systému StackMap patří jednoduchá administrace mapy (administraci zajišťuje StackMap na jejich serverech) jako je editace rozsahů nebo definice nových regálů.

Nevýhody systému jsou roční poplatky v řádu desítek tisíc korun za rok, přidání mapy pouze v rastrovém typu formátu (JPG, PNG) a neodladěná responzivní verze.

Národní technická knihovna

Národní technická knihovna používá svůj vlastní systém pro správu mapy knihovního fondu. Systém na základě signatury, kterou předává do adresy obrázku, zobrazí mapu po stisknutí odkazu umístění. Systém je vlastního vývoje, proto o něm nejdou dohledat informace. Jediné, co jde z kódu vyčíst, je, že systém dynamicky generuje předem nadefinované mapy ve formátu JPG. Jaká mapa se zobrazí určuje systém podle toho, do jakého rozsahu se signatura zařadí (Obrázek č. 2).



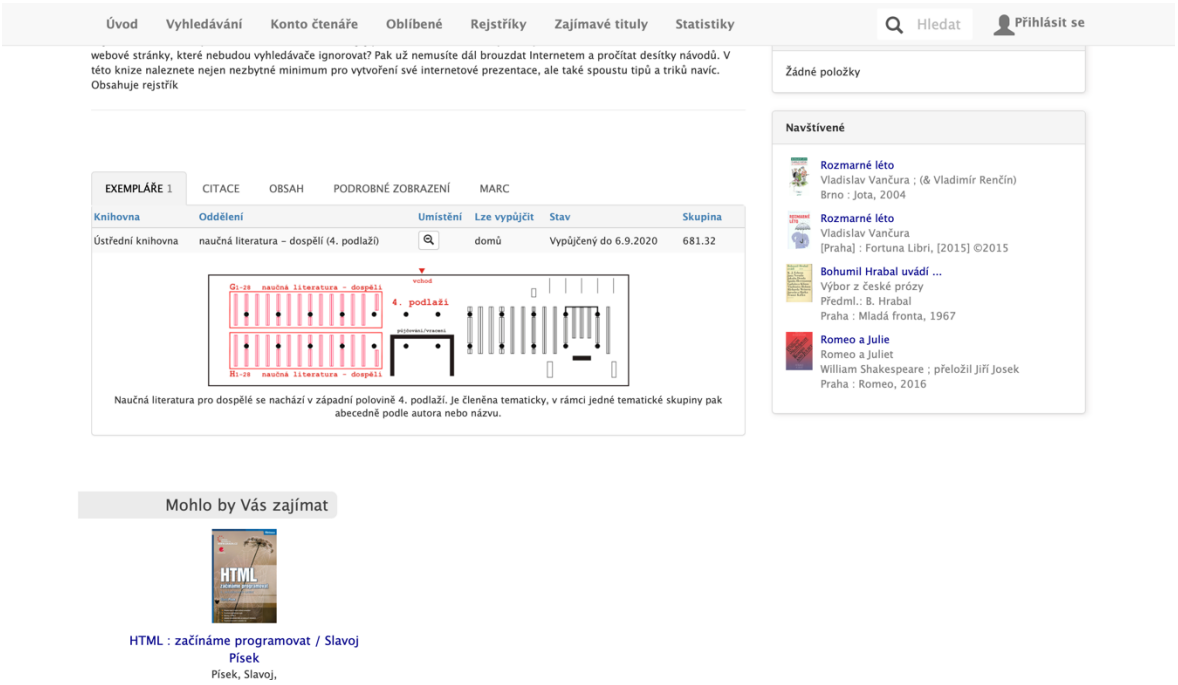
Obrázek č. 2 – Mapa knihovního fondu Národní technické knihovny

Hlavní výhoda mapy fondu Národní technické knihovny je to, že nemusí platit žádné poplatky za správu systému. U nevýhod je to podobné jako u StackMapu. Je opět použita mapa v rastrovém formátu obrázku, která je už přiblížená, což pro nové návštěvníky podstatně zhoršuje orientaci v knihovně. Mezi další nevýhodu bych zařadil, že rozsah obarvených regálů, kde se kniha nachází, je široký.

Krajská knihovna Františka Bartoše ve Zlíně

Krajská knihovna Františka Bartoše ve Zlíně má jednodušší systém mapy knihovního fondu. Systém je vlastního vývoje. Ukazuje pouze patro a zda se kniha nachází ve východní nebo

západní části. Systém zobrazuje mapu podle parametru oddělení, který je přiřazen ke knize (Obrázek č. 3).



Obrázek č. 3 – Mapa knihovního fondu Krajské knihovny ve Zlíně

Jako u Národní technické knihovny je výhoda, že nemusí platit pravidelné poplatky. Mapa je přehledně, jednoduše nakreslená (snadnější orientace) a má kvalitní popisek. Nevýhodou je, že je mapa v rastrovém formátu a obarví se celá polovina patra (ne vybraný regál).

2 ROZDÍL MEZI BLOGEM A CMS

Tato kapitola se zabývá hlavními rozdíly mezi blogy a systémy pro správu obsahu (Content Management System – CMS). Hlavním cílem je analyzovat, co odlišuje systémy a jaké zvláštnosti se v jednotlivých systémech používají. Kromě toho je věnována pozornost prostředím, ve kterých se blogy používají a od kdy má smysl používat systém pro správu webového obsahu.

2.1 Blog

Termín weblog, nebo krátce blog, je neologismus a původně pochází z angličtiny. Termín je složen ze slov web (webový) a log (zápisník) a je to druh virtuálního deníku nebo online žurnálu, ve kterém autor ukládá, spravuje a publikuje svůj obsah [2][3].

Obsah, takzvané příspěvky, obvykle uložené jako články nebo příspěvky, jsou většinou prezentovány v opačném, chronologickém pořadí. To znamená, že příspěvky přidané později budou na vyšším místě webové stránky. Většina blogů nalezených na webu také nabízí čtenářům možnost komentovat publikovaný obsah. Tato funkce umožňuje výměnu znalostí nebo názorů a další rozvoj daného tématu [2].

Blog je publikační médium, kde autoři, známí též jako blogeři, můžou přes snadno použitelný webový software publikovat a diskutovat obsah s ostatními čtenáři. V zásadě je weblog srovnatelný s osobním online deníkem nebo se zjednodušeným webovým fórem [2].

Kromě funkce komentářů jsou důležitou vlastností blogů také trvalé odkazy (permalinks). Každý příspěvek získává uživatelsky „pěkný“, jedinečný a nezávislý odkaz, pod kterým je uveden a je pod ním přístupný. Adresa zdroje na webu (Uniform Resource Locator – URL) každého příspěvku by měla být trvalá a neměla by se nikdy měnit, odtud je název trvalý odkaz. To je výhodné například pro proces optimalizace vyhledávače (Search Engine Optimization – SEO) a také nabízí možnost lepšího zapamatování krátkých adres URL [2][4].

Trvalé odkazy jsou proto často označovány jako „přátelské“ URL. Použití je také užitečné při přidávání záložek (oblíbených stránek) v prohlížeči a usnadňuje vyhledávání. V souvislosti s trvalými odkazy také hrají důležitou roli tzv. Trackbacky [2].

Trackback značí, zda bloger v jiném blogu odkazuje na tematické články nebo odkazuje na externí zdroje. To vytváří diskuse nejen v autorově blogu, ale také za hranicemi příslušného

blogu. V této souvislosti se mluví o sféře blogů, asociaci a propojení všech blogů ve virtuálním síťovém prostoru [2][5].

Takzvaný blogroll má podobnou vlastnost. Blogroll je seznam odkazů na blogu, které odkazují na jiné blogy, weby nebo média a umožňují čtenáři rychle posoudit zájmy blogera, a tak najít nové a další informace o daném tématu. Některé blogy se zabývají pouze odkazováním na jiné stránky a příspěvky [2][6].

Na webu je mnoho typů blogů s různými tématy. Zde je třeba zmínit například „soukromé blogy“, ve kterých autor osobně informuje o tom, co prožil. Tyto „soukromé blogy“ jsou obvykle ve formě webového deníku. V oblasti vývoje webu a softwaru existuje velké množství informačních odborných blogů. Tento typ blogu se často zabývá konkrétními tematickými oblastmi, přičemž autoři mají potřebné základní znalosti, aby vysvětlili čtenáři téma popsané srozumitelným způsobem.

Další velkou část blogové kultury využívají blogy médií a novinářů. Mnozí autoři jsou renomovaní novináři nebo editoři a na svých blozích píšou o aktuálních politických a ekonomických událostech. Stále oblíbenější jsou také „firemní blogy“, „fotoblogy“, „produktové blogy“, „konzultační blogy“, atd. [2].

Mnoho webových blogů také nabízí svůj obsah prostřednictvím novinek. Zprávy jsou srovnatelné s informačními kanály a generují celý nebo část obsahu weblogu v krátké nebo úplné formě. Výhodou těchto zdrojů je, že uživatelé nemusí každý den opakovaně přistupovat k weblogu, aby získali nový obsah. S pomocí programu, tzv. čtečky kanálů, si může čtenář nastavit určité zdroje a témata blogu a je tak informován o aktuálních zprávách. Kanály, jako jsou RSS nebo ATOM, jsou založeny na značkovacím jazyce XML a umožňují přenos obsahu webové stránky ve strojově čitelné formě. Informační kanály se však nepoužívají pouze pro weblogy, ale stále častěji se používají také pro zpravodajské weby [2].

2.1.1 Technologie

Weblogy jsou zpravidla založeny na malých systémech pro správu webového obsahu, které jsou podrobněji vysvětleny v následující kapitole. Existují v podstatě dva způsoby, jak spustit blog:

1. Software blogu běží na centrálním serveru, který také poskytuje nezbytný webový prostor. Tyto blogové služby, například z českých zástupců „blog.cz“ nebo světově známý „blogger.com“ od Googlu, patří mezi nejčastěji používané u blogerů. Většina z těchto služeb je

zdarma a je financována především z příjmů z reklam nebo upgradů a doplňkových služeb. Velkou výhodou takových hostitelů je, že bloger může začít publikovat ihned po registraci a krátkém představení softwaru. Nevýhodou však může být to, že zejména poskytovatelé bezplatných služeb ukončují své platformy a uživatelé pak musí přesouvat svůj obsah k jinému poskytovateli. To se stalo u služby Yahoo! 360° [2][7][8].

2. Software blogu nahraje autorizovaná osoba nebo webmaster na webový server a poté ho nakonfiguruje. Veškerá nastavení týkající se rozložení blogu, funkcí, modulů a rozšíření jsou v rukou blogera nebo administrátora. Nejznámějším zástupcem blogového softwaru v této oblasti je výkonný WordPress. Software blogu je ve většině případů založen na databázích SQL a programovacím jazyce PHP na straně serveru, který mimo jiné generuje dynamicky stránky HTML [2][9].

2.1.2 Výhody blogu

Jednou z největších výhod weblogů je přehlednost a snadnost používání systému. Mnoho nekomerčních systémů nabízí dobře strukturované uživatelské rozhraní, které se snadno používá. Bloger zvládne psát obsah bez zvláštních znalostí programování. Po správné konfiguraci a pravidelném zveřejňování nových článků, dosahuje struktura trvalých odkazů lepší hodnocení vyhledávačů. Přidružené zpětné odkazy také dosahují vyšší pozice ve vyhledávačích než běžné webové stránky [2].

2.2 CMS

Podobně jako termín weblog, pojem „content“ pochází z angličtiny. Výrazem se obecně rozumí jakákoli elektronická forma reprezentace dat. Tento termín se používá hlavně pro textový obsah, obrazové soubory, grafiku, audio a video nahrávky. V zásadě však tento termín odpovídá jakémukoli typu informací, které lze uložit do počítače. Systém pro správu obsahu (CMS) je softwarový balíček, který poskytuje určitou úroveň automatizace pro úkoly potřebné k efektivní správě obsahu [10][11].

Systémy pro správu obsahu podporují neustálou aktualizaci, archivaci a přípravu článků. Existuje celá řada typů systémů pro správy obsahu. Do této skupiny jsou zahrnuty zejména 4 typy, jako je „správa webového obsahu“, „správa podnikového obsahu“, „správa digitálních aktiv“ a „správa záznamů“. Mnoho těchto systémů pro správu obsahu jsou proto také

označovány jako redakční systémy. Pokud jsou používány na internetu, považují se za systémy správy webového obsahu (Web Content Management System – WCMS). Administrace a organizace WCMS je prováděna v rámci webového prohlížeče u klienta [10][11].

Hlavní oblasti takových systémů se obvykle skládají z několika šablon, které jsou zodpovědné za strukturování obsahu, databáze a oddělení programovacích struktur. S pomocí tohoto oddělení lze v podnikovém designu vytvořit několik šablon rozvržení, a tak může být výstupem z databáze dynamický obsah. Zejména weboví návrháři a grafičtí designéři používají šablony, aby si zachovali svobodu při navrhování webových stránek v rámci WCMS, a proto zbytečně nepřicházeli do styku se základním programovým kódem a strukturami. Designéři jsou v zásadě povinni integrovat stylistické detaily do celkového konceptu a zaměřit svou pozornost na viditelnou část stránky (frontend) [10][11].

Systémy pro správu webového obsahu umožňují uživatelsky přívětivou a jednoduchou správu webů. Především správa uživatelů a rolí u WCMS je výhodná. Přiřazení rolí není jen rozšíření bezpečnosti systému, ale také možnost další interní komunikace mezi jednotlivými aktéry [10][11].

Různé nastavení uživatelských rolí u WCMS zajišťuje, že pouze určité oprávněné osoby mohou vytvářet, uvolňovat, prohlížet a archivovat dokumenty či příspěvky. Oddělení uživatelů, rolí a skupin navíc vede ke zlepšení a flexibilitě při nakládání s důvěrnými záznamy. Autoři a editoři, kteří přidávají nebo tvoří obsah v takovém systému, nepotřebují žádné znalosti obvyklých značkovacích jazyků, jako je HTML. Obsah je zadáván většinou pomocí WYSIWYG („What you see is what you get“ – editor stránky s náhledem, jak bude stránka vypadat na frontendu webu) editoru a následně je ukládán do databáze [10][11].

2.2.1 Technologie

Systémy pro správu webového obsahu jsou stále více populární u společností i soukromých osob. Vzhledem k jejich velikosti a rostoucímu zapojení různých autorů se stále více statických stránek převádí na řešení WCMS. To může nabídnout nejen ekonomické výhody spojené s následnou správou systému, ale také časovou úsporu při tvorbě obsahu. Rozsah webů postavených na WCMS roste, to jak v komerční sféře, tak v segmentu open source [10][11].

V oblasti řešeních WCMS je mnoho různých webových technologií, a to jak v open source, tak v komerční oblasti. Patří mezi ně systémy, které jsou implementovány pomocí Java, ASP .NET, Perl, Python, Ruby On Rails, PHP nebo srovnatelnými technologiemi. Většina open

source WCMS je publikována pomocí webového programovacího jazyka PHP a databází MySQL. Mezi 3 nejznámější systémy s otevřeným zdrojovým kódem patří Wordpress, Joomla a Drupal. Důvodem může být jistě licence GPL a také otevřený zdrojový kód. Kromě toho uživatelé najdou na příslušných stránkách daného WCMS řadu různých komunit vývojářů a dokumentaci. Roste také počet odborných autorů blogů a knih, kteří se věnují tématu a nabízejí tak pomoc [10][11].

2.2.2 Požadavky WCMS

Krátký seznam níže by měl objasnit, jaké požadavky mohou být kladeny na WCMS. Určitě nebude existovat žádný systém, zejména v oblasti open source, který by obsahoval všechny uvedené požadavky. Rovněž nemůže být poskytnuta informace o tom, který je nejlepší nebo nejpříjemnější pro uživatele. Webovými návrháři a vývojáři by měly být zkoumány možnosti různých systémů a vybrat ten, který je vhodný pro daný projekt. K tomu však mohou být užitečné následující pokyny a požadavky:

- **Snadné použití.** Správa bez nutnosti pokročilých technických znalostí, jako je kódování. Snadné přidávání a aktualizace textu, přidávání obrázků a videa, vkládání odkazů a další aktualizace obsahu. CMS by měl být intuitivní a uživatelsky přívětivý jak pro technicky zdatné uživatele, tak pro technicky méně zdatné členy personálu.
- **SEO optimalizace.** Naprostá většina CMS má rozšíření, která umožňují uživateli plně spravovat SEO na webu, včetně popisů stránek, názvů. Jde hlavně o to, aby byla možnost zadávat „přátelské“ URL adresy daných stránek či příspěvků.
- **Náklady na CMS.** Některé CMS mají licenční poplatky, například jednorázový poplatek za nákup softwarové licence. Jiné jsou zpoplatněny za aktualizace, upgrade nebo přidání rozšíření.
- **Bezpečnost.** Systémy pro správu obsahu jsou jedním z největších cílů pro hackery a jiné typy počítačových zločinců. Je potřeba zjistit, co říkají ostatní uživatelé z hlediska bezpečnosti a také zjistit, jaké útoky byly provedeny v minulosti a jak rychle na to dodavatel CMS reagoval, např. formou opravné aktualizace.
- **Webová analýza.** Umožnění sledovat klíčové metriky a analýzy, jako jsou konverze, času strávených na stránkách, návštěvy stránek a další. Například možnost napojení na Google Analytics nebo zda má CMS nějaký vlastní systém na analytiku návštěvníků.

- **Technická podpora a dokumentace.** Doporučuje se vybírat ten CMS, který má podporu 24/7 nebo včas reaguje na vaše dotazy a zároveň má obsáhlou dokumentaci [12].

2.2.3 Další funkce WCMS

Mezi další funkce WCMS bych zařadil:

- WYSIWYG editor a další rozšíření s HTML editorem
- Možnosti formátování vytvořených textů
- Editor tabulek
- Funkce pro úpravu obrázku, jako je např. ořez
- Plánované publikování článků
- Generátor formulářů
- Funkce komentářů k článkům
- Ochrana proti spamu
- Archivní funkce
- Generátor souboru Sitemap (pomáhá vyhledávačům s mapováním webu)
- Vyhledávací funkce v systému
- Mnohojazyčnost
- Správa uživatelů
- Čtení novinek pomocí kanálu RSS nebo ATOM
- Funkce zálohování a aktualizace

3 WORDPRESS

Vytvoření webu je pravděpodobně jedna z hlavních věcí, kterou se vlastník firmy v 21. století zabývá nebo také normální uživatel, který chce mít osobní webovou prezentaci.

V dnešní době by měl každý zvážit vytvoření vlastního webu. V minulosti to byl celkem zastrašující krok. Jediným způsobem, jak získat webovou stránku, bylo buď najmout profesionála, který by ji pro vás vytvořil, nebo se naučit webové technologie a poté si ji vyvinout sám. V dnešní době existuje více možností vývoje webu. Pokud má být web vytvořen opravdu rychle a zároveň má být zajištěna odpovídající kvalita, tak je CMS správná a jednoduchá volba.

Nejoblíbenější volbou z hlediska procentuálního využití všech CMS je WordPress. Na něm funguje více než 37,6 % [13] všech webových stránek a očekává se, že se počet v příštích letech ještě zvýší. Více než jeden ze čtyř webů, které je navštěvován, tak pravděpodobně funguje na WordPressu. Z hlediska licence je WordPress, systém pro správu obsahu s otevřeným zdrojovým kódem, licencovaný pod GPLv2, což znamená, že kdokoli může bezplatně používat nebo upravovat samotný software. Zjednodušeně řečeno, s WordPressem může kdokoli vytvořit krásnou webovou stránku s minimálním vynaloženým úsilím a poté ji rychle zpřístupnit světu. Zde jsou ukázány některé výhody používání CMS:

- Nemusí být najat tým vývojářů a designérů.
- Uživatel nemusí být IT profesionál a učit se pokročilé znalosti PHP.
- Přesto však může dojít k vytvoření vysoce kvalitní webové stránky s téměř neomezenými možnostmi rozšíření.

V dnešní době vlastnit web má mnoho dobrých důvodů. Už to nejsou jen velké společnosti, ale i jednotlivci, rodiny, živnostníci nebo malé podniky, které mohou těžit z webové stránky. Současně však většina lidí nemá finanční zdroje na najmutí společnosti pro vývoj webových aplikací nebo nezávislého webového vývojáře, aby pro ně vytvořila webovou stránku. Zde přichází do hry WordPress. WordPress je zdarma, snadno použitelný a má pokročilé funkce. Vzhledem k tomu, že WordPress je webová aplikace, není třeba ji instalovat na domácí PC nebo Mac. Může být na serveru (druh počítače), který patří společnosti zabývající se webhostingem. Původně byl WordPress aplikace určená k provozování webové stránky blogu. Nicméně se vyvinul v plně vybavený systém pro správu obsahu (CMS) [13][14][15].

3.1 Adresářová struktura

Kořenový adresář

V kořenovém adresáři je několik základních souborů WordPressu. Pokud se vývojář nepokouší v jádru kódu WordPressu dělat úpravy nebo nehledá hooky (funkce, pomocí které lze umístit/editovat svůj vlastní kód kdekoli na webu, viz. kapitola 3.2.3 Hooky), které chce používat nebo se pouze chce podívat, jak jsou určité funkce kódovány, tak jediným kořenovým souborem WordPressu, který bude možná potřebova upravit je *wp-config.php*. Nikdy by neměly být měněny žádné jiné základní soubory WordPressu. Úprava základních souborů není vhodná, protože aktualizace na novou verzi přepíše vaše změny. Jediným adresářem, se kterým by se mělo pracovat, je *wp-content*, protože obsahuje pluginy, šablony a nahrané mediální soubory.

Kdykoli budou upravovány základní soubory WordPressu, tak je potřeba se nad tím pořádně zamyslet. Pravděpodobně existuje hook nebo filtr, kterým lze dosáhnout stejného cíle. Pokud není k dispozici hook nebo filtr pro to, co je potřeba, je nutné požádat o jeho přidání. Jádro vývojářů WordPressu reaguje velmi rychle na přidávání nových hooků a filtrů.

V kořenovém adresáři WordPressu může být potřeba ještě jeden soubor v závislosti na nastavení a způsobu, jakým se WordPress používá. Jedná se o soubor *.htaccess*. Nejde o základní soubor WordPressu, ale o soubor Apache serveru, který WordPress potřebuje ke zpracování konfigurace adresářů, trvalých odkazů a přesměrování. Tento soubor není ve výchozím nastavení k dispozici, je vytvořen automaticky při prvním definování struktury trvalých odkazů [16].

Složka wp-admin

Tento adresář obsahuje základní adresáře a soubory pro správu administračního rozhraní WordPressu. Klíčovým souborem v tomto adresáři je *admin-ajax.php*, kterým jsou zpracovávány všechny ajaxové požadavky [16].

Složka wp-includes

Adresář *wp-includes* obsahuje základní adresáře a soubory pro různé funkce WordPressu. Důrazně se doporučuje prohlédnout strukturu a kód v tomto adresáři, aby bylo lépe pochopeno samotné fungování WordPressu [16].

Složka wp-content

Tento adresář je místem, kde uživatelé a vývojáři WordPressu můžou dělat, co chtějí. Obsahuje podadresáře pro pluginy a šablony, které byly nainstalovány na web a také všechny mediální soubory, které jsou na webu nahrány. Adresář *wp-content* obsahuje několik podadresářů [16].

Složka wp-content/plugins

Do tohoto adresáře se uloží jakýkoli plugin, který bude nainstalován na webu. Ve výchozí instalaci je WordPress dodáván s pluginy Hello Dolly a Akismet.

Hello Dolly je názorná ukázka, jak funguje základní plugin WordPressu. Samotný plugin zobrazuje náhodný text z písně „Hello Dolly“ v pravém horním rohu administrátorského rozhraní. Plugin Akismet pomáhá zastavit spamové komentáře u příspěvků. Tento plugin výrazně snižuje počet spamových komentářů, které by se jinak dostávaly na web [16].

Složka wp-content/themes

V adresáři *themes* jsou umístěny všechny šablony WordPressu, které jsou nainstalovány na webu. Ve výchozím nastavení WordPress přichází s několika standardními šablonami, všechna jsou pojmenovaná podle roku, kdy byla vydána (Twenty Seventeen, Twenty Nineteen, atd.) [16].

Složka wp-content/uploads

Jakmile budou nahrány jakékoli fotografie nebo soubory do mediální knihovny, tak všechny nahrávané soubory se budou ukládat do tohoto adresáře. Některé pluginy si vytvoří v adresáři *uploads* podadresář pro různé soubory používané nebo spravované daným pluginem [16].

3.2 Pluginy

Od prvních verzí byl WordPress vždy navržen jako velmi otevřená platforma. Tato otevřenost byla podpořena nejen prostřednictvím licenčního a distribučního modelu open source, ale také open plugin architekturou, která dala vývojářům možnost poskytovat ještě bohatší možnosti svým uživatelům.

Zatímco základní instalace WordPressu poskytuje velké množství funkcí, které se neustále rozšiřují. Vydáním každé nové verze uživatelé ale často potřebují přidat ještě nějakou funkci, aby určitým způsobem vylepšila systém správy webových stránek. A to je právě možnost,

kdy je možno použít úpravy formou pluginu. Pluginem lze přidat různé rozšíření nebo manipulovat prakticky s jakýmkoli aspektem zobrazovacích a administrativních věcí na WordPressu.

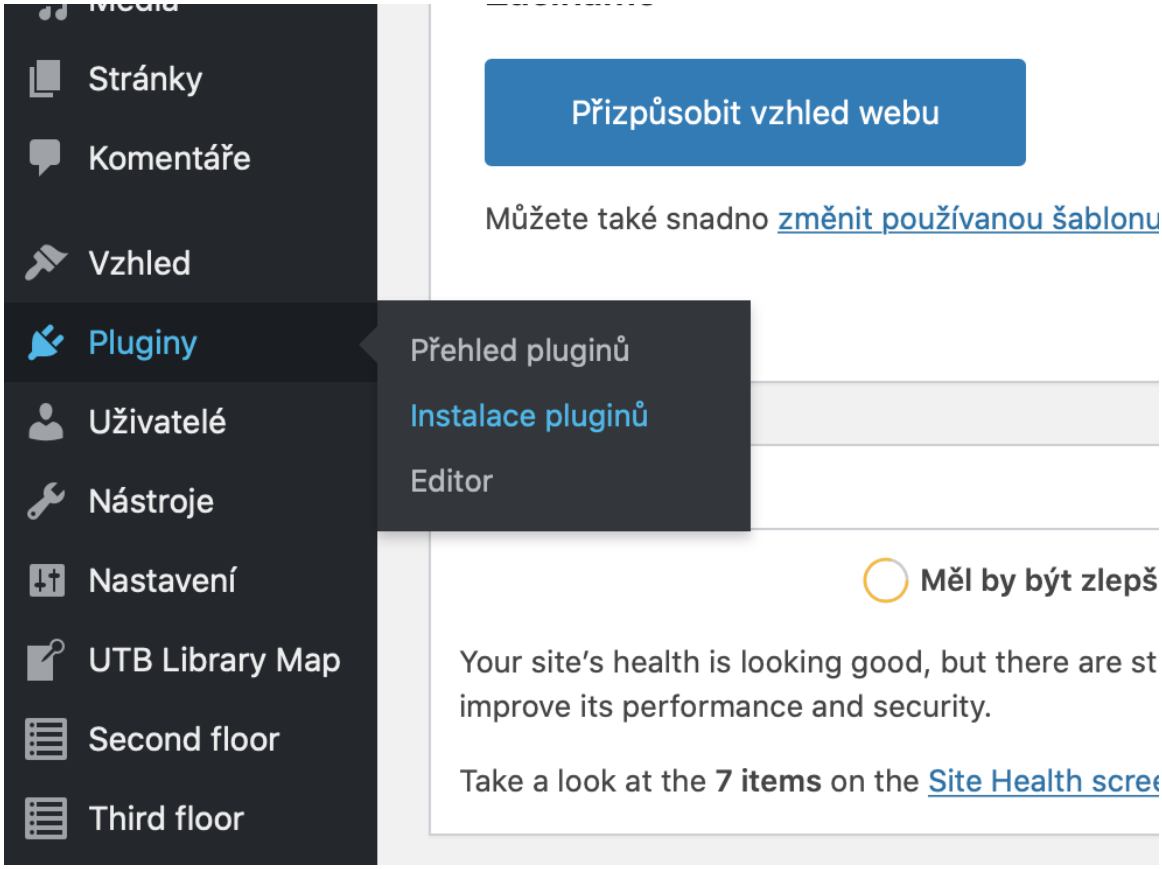
Stejně jako WordPress, tak i pluginy jsou psány v programovacím jazyce PHP, který je strukturálně podobný tradičním jazykům, jako jsou například C a C ++. Tento kód je uložen v obyčejných textových souborech, které se čtou a provádějí na webovém serveru, když se zavolá požadavek zobrazení stránek. Mechanismus zpětného volání (callback) umožňuje pomocí hooků přidat funkce pluginu do zdrojového kódu WordPressu. Tyto hooky máme ve dvou verzích, které se nazývají action a filter. Umožňují pluginům přidávat obsah na web a upravovat data před jejich zobrazením. Ať už se jedná o úpravu frontendu webu nebo administračního rozhraní.

Kromě schopnosti rozšířit funkčnost WordPressu je vedlejší výhodou pluginů to, že většina funkcí, které na web přidávají, je nezávislá na aktivní šabloně. Proto se uživatelé, kteří rádi mění své šablony, nemusí starat o opětovné ruční přidání vlastních úprav do svých nových šablon, když ji přepnou na novou [17].

Prostřednictvím oficiálního úložiště WordPressu máme přístup k více než 55 000 pluginům (srpen 2020) [16]. Vzhledem k tomu, že zde nejsou obsaženy všechny pluginy, tak je možno najít další na internetu v různých fórech a repozitářích. Mnoho tvůrců pluginů je prezentuje prostřednictvím osobních nebo firemních webů a mnozí si za ně účtují poplatky. Tyto pluginy pak nazýváme též jako prémiové. Podobně jako u mobilních aplikací, tak i u pluginů může být zdarma k dispozici lite verze (zjednodušená verze, která většinou neobsahuje všechny nabízené funkce) prémiového pluginu, kdy je následně za poplatek zpřístupněna plná verze. Toto se většinou využívá, když uživatel chce vyzkoušet daný plugin. Většina prémiových pluginů také nabízí licence pro vývojáře, což umožňuje vývojářům vytvářejícím více webů zaplatit jednou za plugin, který pak mohou instalovat na více instalací WordPressu [16].

Při instalaci pluginu se jednoduše přihlásíme do administrace svého WordPressu, známého také jako backend. Klikneme v sekci „Pluginy“ na „Instalace pluginů“, jak je znázorněno na obrázku č. 4. Poté máme možnost prohledat oficiální úložiště pluginů WordPressu nebo je možno nahrát plugin, který byl stažen nebo koupen z jiného zdroje. Po dokončení vyhledávání a nalezení potřebného pluginu je provedena instalace. Nainstalovaný plugin se následně

může aktivovat. Pokud plugin není aktivován, zůstane deaktivován na záložce „Pluginy“. Je nutno mít na paměti, že po aktivaci je většinou potřeba plugin nakonfigurovat [16].



Obrázek č. 4 - Instalace pluginu

Pokud byl stažen plugin z jiného zdroje, než je oficiální úložiště pluginů, soubor by měl být ve formátu ZIP. Má-li být plugin nahrán na web, je nutno kliknout v sekci „Instalace pluginů“ na možnost „Nahrát plugin“ a poté vybrat ZIP soubor obsahující plugin. Plugin je možno rovnou aktivovat nebo ho případně v sekci „Pluginy“ aktivovat později [16].

3.2.1 Vlastní plugin

Při tvorbě pluginu se nejprve musí v adresáři *wp-content/plugins* vytvořit nová složku nazvaná například *muj-plugin* a v této složce vytvořit PHP soubor s názvem *muj-plugin.php*. Soubor *muj-plugin.php* bude následně otevřen v textovém editoru, například editor s názvem Brackets, a pak bude do něho vepsán následující text:

```
1 <?php
2 /**
3  * Plugin Name: Můj plugin
4  * Plugin URI: https://dolezal.cloud/muj-plugin/
```

```
 5  * Description: Stručný popis pluginu.
 6  * Author: Tomáš Doležal
 7  * Version: 1.0
 8  * Author URI: https://dolezal.cloud
 9  * License: GPLv2
10  * License URI: http://www.gnu.org/licenses/gpl-2.0.txt
11  */
12 ?>
```

Soubor se uloží a tím je vytvořen plugin, který sice zatím nic nedělá, ale měl by být viděn v sekci „Pluginy“, kde se zároveň i aktivuje.

Když je plugin aktivovaný, je potřeba zkusit přidat kód pro vypsání textu do patičky webu. Následující kód je nutno zkopírovat a uložit za informaci o pluginu:

```
1 <?php
2 function muj_plugin_wp_footer() {
3     echo 'Vypisujeme testovací text v patičce webu.';
4 }
5 add_action( 'wp_footer', 'my_plugin_wp_footer' );
6 ?>
```

Na frontendu webu by měl být viděn v patičce vypsaný testovací text. Toto je velmi základní příklad, jak začít. Dále budou ukázány další základní prvky pluginu [16].

3.2.2 Struktura pluginu

Některé pluginy vykonávají pouze jednu nebo dvě věci a pro takové stačí jeden PHP soubor. Ale existují také pluginy, které jsou mnohem komplikovanější. Skládají se z mnoha souborů (CSS, obrázky a šablony), včetně knihoven, souborů tříd a tisíce řádků kódu, které je vhodné zorganizovat do více než jednoho souboru.

Následující struktura souborů pluginu byla vygenerována pomocí webové aplikace WordPress Plugin Boilerplate Generator, která slouží k vygenerování základní struktury, včetně všech potřebných složek a souborů obsahujících základní třídy, pluginu pro WordPress. Ne všechny tyto složky a soubory mohou být nezbytné a podle potřeby budou přidány do pluginu [16][18].

Základní struktura pluginu:

Složka admin

Zde jsou soubory, které mají být pomocí pluginu zobrazeny v administraci WordPressu, jde

například o položky v menu nebo poté o samostatné stránky, na kterých je plugin konfigurován. Hlavně jde o class soubor a display soubor.

Složka public

Adresář *public* obsahuje soubory, které jsou zobrazovány na frontendu webu. Jedná se opět o class soubor, kde je funkční stránka věci a poté o display soubor, pomocí kterého je vypsán obsah na frontend.

Složka includes

Do adresáře *includes* patří všechny systémové PHP soubory, které plugin potřebuje, například k instalaci. Jediný PHP soubor v kořenové složce pluginu by měl být hlavní soubor pluginu. Všechny ostatní PHP soubory by měly jít do jedné z ostatních složek.

Standardním postupem je přidání souborů *function.php*, *common.php* nebo *helpers.php* tak, aby obsahoval pomocný PHP kód používaný pluginem. Tento soubor by měl obsahovat jakékoli malé skripty, které nemají ústřední roli v logice nebo funkčnosti pluginu, ale jsou nutné k jeho fungování. Například oříznutí textu, funkce pro generování náhodných řetězců nebo jiné podobné funkce, které nejsou k dispozici prostřednictvím základních funkcí WordPressu.

Složka css

Do této složky patří CSS soubory, které používá výhradně plugin. Složka *css* je v pluginu na dvou místech. Jedná se o adresář *admin* a *public*. Zde se nahrávají vlastní CSS soubory v závislosti na tom, kde je dané CSS potřeba, jde o administraci a frontend.

Složka js

Ve složce *js* jsou vloženy všechny soubory JavaScriptu potřebné pro plugin. Složka je, jak v případě CSS, na dvou místech. A to ve složce *admin*, tak i ve složce *public*. Do těchto složek se můžou také přidat knihovny JavaScriptu třetích stran, které budou používány.

Stejně jako u stylů může být obtížné určit, zda se má soubor JavaScriptu uložit do adresáře pluginu nebo do adresáře šablony. Obecně platí, že soubory JavaScriptu, které vyžaduje šablona (např. efekty slideru nebo animace menu), by měly jít do adresáře šablony a soubory JavaScriptu, které používá plugin (např. AJAX kód) se řadí do složky pluginu [16].

Hook *admin_enqueue_scripts* je použit k zařazení skriptů a stylů určených pro administraci WordPressu, ale hook *wp_enqueue_scripts* se používá k zařazení skriptů a stylů určených pro frontend.

3.2.3 Hooky

Hooky jsou „alfou a omegou“ WordPressu. Jsou to základní funkce, které jsou potřeba k vývoji pluginu. Hooky umožňují pluginům připojit se k WordPressu a vykonávat potřebné funkce. I když je možné vytvořit plugin, který nepotřebuje funkci hooku, ale takový plugin by byl raritou. Většina pluginů použije alespoň jeden hook. Jakmile se vývojář začne učit, jak hooky fungují, tak pochopí, proč je WordPress tak výkonná platforma.

V aplikaci WordPress jsou hooky hlavní funkcí Plugin API. Umožňují vývojářům pluginů upravovat WordPress, aniž by změnily základní kód. Pokud není upraveno jádro WordPressu, tak mohou uživatelé upgradovat na novější verze, aniž by ztratili jakékoli změny, které provedly jejich pluginy. A to je právě jedna z hlavních výhod hooků.

WordPress má dva typy hooků: action a filter hook. Action hook umožní provést funkci (akci) v určitém bodě provádění kódu WordPressu. Filter hook umožňuje manipulovat (filtrovat) výstup procházející hookem [14][19].

Action hook

Action hooky umožňují spustit kód v určitém okamžiku načítání WordPressu. Jsou považovány za „události“, ke kterým dochází v určitých fázích načítání webové aplikace. Připojením funkce k action hooku náš plugin řekne WordPressu, že chce při této události něco udělat.

Funkce *do_action()* ve WordPressu vytvoří a spustí action hook.

```
1 <?php
2 do_action($tag, $arg = '');
3 ?>
```

Funkce přijímá dva parametry:

- *$tag:* Název nebo identifikátor pro action hook.
- *$arg*: Hodnota předaná registrovaným akcím. Tento parametr lze rozdělit na libovolný počet parametrů.

Následuje příklad, jak by vypadal action hook s více parametry:

```php
<?php
do_action($tag, $arg_a, $arg_b, $arg_c);
?>
```

Jedním z nejčastějších hooků ve WordPressu je action hook *wp_head*. Je aplikován v rámci <*head*> značek ve struktuře webu. Je to užitečný hook pro SEO pluginy a pro přidávání Open Graph meta značek.

```php
<?php
do_action('wp_head');
?>
```

Jak lze vidět, tento konkrétní action hook nepředává žádné parametry. Je tomu často u action hooků. Když se tento kód spustí ve WordPressu, tak hledá každou action zaregistrovanou na hooku *wp_head*. Poté provede každý action hook v pořadí podle priority. Proces načítání stránky pokračuje i po dokončení jeho provedení.

Nyní se podíváme na action hook, který předává další argumenty všem akcím, které jsou k němu připojeny.

```php
<?php
do_action('save_post', int $post_ID, WP_Post $post, bool $update);
?>
```

Tento hook se spustí, když je uložen jakýkoli příspěvek ve WordPressu. Předá ID příspěvku, obsah příspěvku a logickou hodnotu, zda je příspěvek nový nebo je upravený. To mohou být užitečné informace, které máme k dispozici při připojování vlastní akce k *save_post* [19].

Filter hook

Filter hooky jsou druhým typem hooků ve WordPressu. V některých ohledech jsou podobné action hookům. Jsou však velmi odlišné ve své nejdůležitější funkci, která umožní manipulovat s výstupem kódu. Filter hooky fungují tak, že vždy procházejí data hookem a jakýkoli filtr na hooku může data jakýmkoli způsobem upravovat.

Aby bylo pochopeno, jak fungují filter hooky, je nutno se nejprve naučit, jak funguje funkce WordPressu *apply_filters()*. Je to základ systému filter hooků.

```php
<?php
apply_filters(string $tag, mixed $value);
?>
```

Funkce využívá dva a více parametrů.

- *$tag*: Jedinečný název filter hooku.
- *$value*: Hodnota, která je předána všem filtrům přidaným do hooku za účelem úprav.
- *$args:* Stejně jako funkce *do_action()* pro action hooky, tak i filter hook akceptuje libovolný počet dalších argumentů, které se předají filtrům připojeným k hooku.

Funkce vždy vrátí proměnnou *$value* poté, co byla filtrována. Je také nezbytné, aby všechny filtry přidané do hooku také vrátily hodnotu.

Následuje příklad filter hooku v jádru WordPressu, kde lze vidět, jak filter hook vypadá.

```
1 <?php
2 $template = apply_filters('template_include', $template);
3 ?>
```

V tomto příkladu je *template_include* název filter hooku. *$template* je řetězec představující cestu k souboru šablony, která se má načíst. Plugin by například mohl tuto cestu k souboru filtrovat, aby WordPress věděl, že by měl načíst šablonu z jiného umístění [19].

3.3 Šablony

Šablony WordPressu řídí frontendové rozhraní webové aplikace a pohledem jsou podobné MVC frameworku. Analogie není dokonalá, ale šablony a view jsou podobné v tom, že ovládají, jak bude aplikace vypadat a jsou místem, kde návrháři tráví většinu času.

Theme Developer Handbook sestavená komunitou WordPressu je dobrým zdrojem pro naučení, jak vytvářet šablony pro WordPress standardním způsobem. Tento zdroj by měli používat všichni vývojáři šablon.

Kdyby měla být porovnávána šablona s pluginem, tak na určité úrovni jsou všechny zdrojové soubory v šablonách a pluginech pouze soubory formátu PHP načtené v různých časech pomocí WordPressu. Teoreticky se může celý kód webové aplikace nacházet v jedné šabloně nebo jednom pluginu. V praxi je ale lepší vyhradit kód pro šablonu související s frontendem (views) webu a pluginy pro backend aplikace (model a controller).

Kde se vloží nějaký kód, bude záležet na tom, zda se bude primárně vytvářet kompletní webová aplikace či samostatný plugin nebo šablona [16].

Při vytváření aplikací je doporučeno použít následující pokyny:

- K uložení základního kódu aplikace je potřeba jeden hlavní plugin a ke správě kódu frontendu jednu šablonu.

- Jakákoli modulární funkce, která by mohla být užitečná v jiných projektech nebo případně nahrazena jiným pluginem, by měla být kódována jako samostatný plugin.
- Nikdy neupravovat systémové soubory WordPressu.

Pluginy se přirovnávají k modelu a controlleru, kde všechny datové struktury definující kód (logika a např. AJAX) by měly jít do hlavního pluginu. Věci jako taxonomie, zpracování formulářů, třídy Post a User by také měly jít do pluginu.

Šablony jsou zase přirovnávané k view. Všechny frontendové kódy a samotná logika frontendu by měly patřit do šablony. Hlavní rámec webu, záhlaví, zápatí, menu a postranní panely by měly být kódovány opět v šabloně, jako i jednoduché logické podmínky, např. zobrazení menu jen pro přihlášeného uživatele.

Je-li vytvářen plugin, který bude použit na jiných webech, tak je rozumné ponechat kód v pluginu. V těchto případech je možno do pluginu ukládat i soubory šablon, aby bylo možné poté na jiném webu zachovat UI komponenty.

3.3.1 Struktura šablony

Když uživatel navštíví web a přejde na jakoukoliv stránku, použije WordPress systém, který se nazývá „template hierarchy“ (určuje, který soubor se má použít k vykreslení potřebné stránky). Například pokud uživatel přejde na stránku příspěvky, WordPress vyhledá soubor *single-post.php*. Pokud to nenajde, bude hledat *single.php* a pokud ani ten nenajde, tak bude hledat *index.php*.

Soubor *index.php* je záložním souborem pro všechna načtení stránky a se souborem *style.css* je jediným požadovaným souborem pro fungování šablony. Obvykle je v šabloně následující seznam souborů:

- *404.php*
- *author.php*
- *archive.php*
- *attachment.php*
- *category.php*
- *comments.php*
- *date.php*
- *footer.php*
- *front-page.php*

- *functions.php*
- *header.php*
- *home.php*
- *image.php*
- *index.php*
- *page.php*
- *search.php*
- *sidebar.php*
- *single.php*
- *single-(post-type).php*
- *style.css*
- *tag.php*
- *taxonomy.php*

Některé soubory v tomto seznamu se načtou, když se zavolá konkrétní funkce *get*. Například *get_header()* načte *header.php*, *get_footer()* načte *footer.php* a *get_sidebar()* načte *sidebar.php*. Předáním parametru těmto funkcím se přidá parametr do názvu souboru. To znamená, že například *get_header('alternate')* načte soubor *header-alternate.php* ze složky šablony [16].

3.3.2 Šablona stránky

Jedním z nejjednodušších způsobů, jak spustit libovolný PHP kód na webu WordPressu, je vytvořit šablonu stránky, přidat ji do šablony a poté ji použít na jedné ze svých stránek.

WordPress prohledá všechny PHP soubory ve složce a podsložkách aktivní šablony, zda neobsahuje soubory šablony stránky. Všechny soubory nalezené se zakomentovaným řádkem, který obsahuje frázi „Template Name:“ se zobrazí ve výběru v pravém sloupci editace příspěvku, kde se daná šablona stránky může aktivovat.

```
1 <?php
2 /*
3 Template Name: Vlastní šablona stránky
4 */
5 ?>
```

Výše je ukázka kódu, který představuje vytvoření vlastní šablony stránky [16].

3.4 Custom Post Type

Při vývoji některých webů je možno se setkat s vlastními typy příspěvků (Custom Post Type). Nejběžnější typy obsahu ve WordPressu jsou příspěvky a stránky. Při domněnce, že si to projekt vyžaduje, lze vytvořit libovolný počet nových typů příspěvků využitím výhod vlastní funkce typu příspěvku a jeho širokého nastavení (pozice v menu, možnost hledání, zobrazení pro určitého uživatele, atd.).

Potřeba vlastních typů příspěvků se objevuje v mnoha scénářích. Například, když fotograf vytváří web osobního portfolia, možná bude potřebovat vlastní typ příspěvku s referencemi, aby prezentoval své focení atraktivním způsobem, namísto pouhého použití standardních příspěvků, kde by se mu to míchalo s ostatními typy příspěvků, jakou jsou například blogové články.

U vlastních příspěvků je možnost použít vlastní meta boxy a taxonomie. Meta boxy jsou určené k tomu, aby mohl vývojář u příspěvku přidat další uživatelské pole. Když se vezme v potaz výše zmíněný příklad fotografa, tak u příspěvku reference se například přidají meta boxy data vyfocení, místa focení nebo typ fotoaparátu. Zato taxonomie je něco jako kategorie, kde je možnost příspěvky zařazovat. V tomto případě třeba svatební focení, sportovní fotografie anebo studiové focení.

Kód vlastního typu příspěvku lze také přidat k pluginu nebo widgetu, pokud si vývojář nepřeje, aby byl vázán na konkrétní téma [15][20].

3.4.1 Vlastní typ příspěvků

Vlastní typ příspěvku je možno přidat k šabloně nebo pluginu, případně widgetu, jak je zmíněno výše. V případě šablony stačí přidat kód do souboru *functions.php* vaší šablony. Je dobrým zvykem spojit vytvoření nového typu s funkcí *init*, aby byl spuštěn ve správný okamžik v zaváděcím procesu. Následuje úvodní kód typu *post_type*:

```
1 <?php
2 function reference_init() {
3   register_post_type('reference');
4 }
5 add_action('init', 'reference_init');
6 ?>
```

Funkce *register_post_type()* bere svůj druhý parametr jako pole a v tomto poli je určeno, zda je objekt veřejný nebo má být zobrazena úprava URL, zobrazení nadpisu nebo editoru, atd. Následující kód ukazuje nastavení pole argumentů a následné předání funkci:

```
<?php
function reference_init() {
  $args = array(
    'description' => 'Vlastní typ příspěvku mých referencí',
    'public' => true,
    'rewrite' => array('slug' => 'reference'),
    'has_archive' => true,
    'supports' => array('title', 'editor', 'author', 'excerpt',
    'custom-fields', 'thumbnail'),
    'show_in_rest' => true
  );
  register_post_type('reference', $args);
  flush_rewrite_rules();
}
add_action('init', 'reference_init');
?>
```

Dále zde je pole s popisky (*$labels*), kde jsou pojmenovaná akční tlačítka v daném typu příspěvku, jako je například „Přidat nový příspěvek“, v našem případě to je „Přidat novou referenci“ [15][20].

```
$labels = array(
  'name' => 'Reference',
  'singular_name' => 'Reference',
  'add_new' => 'Přidat novou',
  'add_new_item' => 'Přidat novou referenci',
  'edit_item' => 'Editovat referenci',
  'new_item' => 'Nová reference',
  'view_item' => 'Zobrazit referenci'
);
```

3.4.2 Vlastní taxonomie

Na vytvoření vlastní taxonomie bude použit předchozí příklad. Fotograf pravděpodobně nebude chtít kombinovat reference svateb a sportovních událostí, takže si vytvoří vlastní taxonomii s názvem „Kategorie referencí“. Následující kód bude přidán do souboru *functions.php*:

```php
<?php
function reference_taxonomies() {
    $args = array();
    register_taxonomy('reference_category', 'reference', $args);
}
add_action ('init', 'reference_taxonomies', 0);
?>
```

Stejně jako u vlastního typu příspěvků, tak i zde může být nastavena celá škála parametrů:

```php
<?php
function reference_taxonomies() {
  $args = array(
    'hierarchical' => true,
    'public' => true,
    'show_in_rest' => true,
    'label' => 'Kategorie referencí',
    'query_var' => true,
    'rewrite' => array('slug' => 'reference-kategorie')
  )
  register_taxonomy('reference_category', 'reference', $args);
}
add_action('init', 'reference_taxonomies', 0);
?>
```

Je to velmi podobné vlastnímu typu příspěvku. Jak lze vidět, tak je opět možnost přidat popisek nebo třeba povolit hierarchii [15][20].

3.4.3 Vlastní meta boxy

Jako vývojář pluginů je důležité si uvědomit, že dlouhodobá budoucnost meta boxů je trochu nejistá. Se zavedením editoru bloků (Gutenberg) se vývojářům doporučuje používat systém blokových editorů pro zpracování post metadat. Neexistuje žádný pevný termín pro to, zda a nebo kdy budou meta boxy zcela zastaralé a odstraněné z WordPressu. Toto je důležité mít na paměti při rozhodování o tom, jak uživatelům představit možnosti metadat.

Meta boxy jsou „boxy“ na obrazovce pro úpravy příspěvků, které umožňují uživatelům upravovat metadata, jako je např. box pro hlavní obsah příspěvku, autora nebo změnu URL. Při vývoji pluginu nebo šablony je možnost vlastního vytvoření těchto meta boxů. Technicky jde dát do meta boxu cokoli. Meta boxy jsou dobrou volbou k rozšíření uživatelského rozhraní úpravy příspěvku.

Pro přidání vlastního meta boxu na obrazovku úpravy příspěvku se použije funkce *add_meta_box()*.

Přidání meta boxu:

```
<?php
function misto_foceni() {
  add_meta_box(
    'misto_foceni',
    __( 'Místo focení reference', 'textdomain' ),
    'misto_foceni_obsah',
    'reference',
    'normal',
    'default'
  );
}
add_action( 'add_meta_boxes', 'misto_foceni' );
?>
```

Funkce *add_meta_box()* přijímá parametry, které obsahují ID meta boxu, nadpis, callback funkci nebo třeba pozici meta boxu [15][20].

4 POUŽITÉ TECHNOLOGIE

Tato kapitola se zabývá základními technologiemi, které se využívají v CMS WordPress a grafickým vektorovým formátem SVG, ve kterém se bude vytvářet mapa pater knihovny.

4.1 HTML

HTML (Hypertext Markup Language) je kód, který se používá ke strukturování webové stránky a jejího obsahu. Například obsah může být strukturován v sadě odstavců, seznamu odrážkových bodů nebo pomocí obrázků a tabulek.

HTML není programovací jazyk, ale je to jazyk značkovací, který definuje strukturu vašeho obsahu. HTML se skládá z řady prvků, které se používají k „uzavření“ nebo „zabalení“ různých částí obsahu. Uzavírací značky mohou vytvořit systém nadpisů, hypertextový odkaz na slovo nebo obrázek, který po kliknutí uživatele odkáže na cíl odkazu, mohou textu dát kurzívu nebo ho dát tučným písmem, atd. Jako příklad poslouží následující text:

```
1 Mapa knihovního fondu
```

Pokud má být text na samostatném řádku jako odstavec, tak je přidán do značek odstavce:

```
1 <p>Mapa knihovního fondu</p>
```

Hlavní části ukázkového prvku jsou následující:

- Úvodní značka: Skládá se z názvu značky (v tomto případě *p*), zabaleného do úhlových závorek. Zde se uvádí, kde prvek začíná – v tomto případě, kde začíná odstavec.
- Závěrečná značka: Zde je to stejné jako při úvodní značce, kromě toho, že před název prvku přidáme lomítko. Závěrečnou značkou uvádíme, kde prvek končí. Nepřidání závěrečné značky je jednou ze standardních chyb a může se tím rozhodit struktura webové stránky.
- Obsah: Obsah prvku je většinou text, který se nachází mezi úvodní a koncovou značkou.
- Prvek: Úvodní značka, závěrečná značka a obsah společně tvoří prvek [21].

4.2 PHP

PHP (rekurzivní akronym pro PHP: Hypertext Preprocessor) je široce používaný skriptovací programovací jazyk, který je vhodný zejména pro vývoj webových stránek.

PHP lze použít dvěma primárními způsoby:

Skriptování na straně serveru

PHP bylo původně navrženo tak, aby vytvořilo dynamický webový obsah, a je pro tento úkol stále nejvhodnější. K vygenerování HTML je potřeba PHP analyzátor a webový server, přes který budou zasílány soubory kódovaných dokumentů. PHP se také stalo populárním pro generování dynamického obsahu prostřednictvím databázových připojení, XML dokumentů, grafiky, souborů PDF a mnohem více.

PHP běží na všech hlavních operačních systémech, od unixových variant (včetně Linuxu, FreeBSD, Ubuntu, Debian a Solaris) až po Windows a MacOS. Může být použit se všemi předními webovými servery, včetně serverů Apache, Nginx a OpenBSD. Na vzestupu jsou i cloudová prostředí jako Azure a Amazon.

Samotný jazyk je velmi flexibilní. Například nemá omezení pouze na výstup v HTML nebo jiném textovém souboru, ale lze vygenerovat jakýkoli formát dokumentu. PHP má vestavěnou podporu pro generování souborů PDF, obrázků GIF, JPG a PNG.

Jednou z nejvýznamnějších funkcí PHP je rozsáhlá podpora databází. PHP podporuje všechny hlavní databáze (např. MySQL, PostgreSQL, Oracle, Sybase, MS-SQL nebo DB2). Podporovány jsou i novější databáze NoSQL, jako je CouchDB a MongoDB. S PHP je vytváření webových stránek s dynamickým obsahem z databáze poměrně jednoduché.

Skriptování z příkazového řádku

PHP může spouštět skripty z příkazového řádku, podobně jako Perl, awk nebo Unix shell. Skripty příkazového řádku se můžou použít pro úkoly správy systému, například pro zálohování nebo třeba analýzu logů. Dokonce i některé skripty typu CRON mohou být provedeny tímto způsobem [22].

4.3 CSS

Cascading Style Sheets (CSS) je webová technologie, která umožňuje aplikovat rozvržení, téma a styl na dokumentu. Ve většině případů je dotyčným dokumentem soubor HTML a vykreslování se provádí pomocí webového prohlížeče.

CSS je často považován za konstrukční nástroj, protože umožňuje autorovi nebo designérovi webové stránky určit vizuální vzhled této stránky. Díky své kontrole nad konečným vzhledem webové stránky má CSS přímý dopad na použitelnost i dostupnost. Vzhledem k těmto

faktorům se vytváření stylů a psaní CSS někdy považuje za designérské úkoly a za správu stylů může být zodpovědný designér softwarového týmu.

CSS definují barvu, velikost, polohu textu a dalších HTML značek, zatímco soubory HTML definují obsah a způsob jeho uspořádání. Jejich oddělením je možno změnit barevné schéma, aniž by bylo nutno přepisovat celý web.

Kaskádování znamená, že styl aplikovaný na nadřazený prvek se bude vztahovat také na všechny podřízené prvky v nadřazeném prvku. Například nastavení barvy textu těla dokumentu bude znamenat, že všechny nadpisy a odstavce v těle budou mít stejnou barvu.

Tři hlavní způsoby možnosti implementace CSS na webovou stránku:

- Inline: Do úvodní HTML značky se vloží atribut *style*, kde se definují potřebné styly.
- Značka style: Styly se píšou do HTML prvku *<style></style>*.
- Externí: Podobné druhému typu (značka style), jen s tím rozdílem, že se styly píšou do externího CSS souboru [23][24].

4.4 JavaScript

JavaScript je plnohodnotný dynamický programovací jazyk, který může na web přidat interaktivitu. Byl vynalezen Brendanem Eichem (spoluzakladatelem projektu Mozilla), Mozilla Foundation a Mozilla Corporation.

JavaScript je univerzální a přátelský pro začátečníky. S většími zkušenostmi je možnost vytvářet hry, animovanou 2D a 3D grafiku nebo třeba komplexní aplikace založené na databázi.

Samotný JavaScript je relativně kompaktní, přesto velmi flexibilní. Vývojáři napsali nejrůznější nástroje nad základní jazyk JavaScriptu a odemkli obrovské množství funkcí s minimálním úsilím.

Tyto funkce zahrnují např.:

- Rozhraní pro programování aplikací (Application Programming Interface – API) třetích stran, která umožňují vývojářům začlenit funkce do webů od jiných poskytovatelů obsahu.
- Frameworky a knihovny třetích stran, které můžete použít pro urychlení práce při vytváření webů a aplikací [25].

4.5 SVG

Scalable Vector Graphics (SVG) je jednou z nejvýkonnějších součástí moderního vývoje webu. Při správném použití může vyřešit běžné problémy související s návrhem, vývojem a dodáváním obrazových snímků a UI.

SVG je značkovací jazyk založený na XML používaný k definování obrázků, který může obsahovat dvourozměrné vektory. Vektory mohou být jednoduché cesty, přímky nebo jakýkoli tvar, který lze popsat pomocí křivek. Je velmi flexibilní a může být implementován jako samostatný obrázek pomocí atributu *src* obrázku nebo jako obrázek na pozadí v CSS, jako je PNG, GIF nebo JPG. Může být také vložen přímo do stránky HTML a manipulovat s ním pomocí CSS nebo JavaScriptu za účelem vytváření animací, vizualizací nebo třeba interaktivních grafů.

SVG bylo vydáno v roce 1999 (je starší než XHTML), avšak kvůli nedostatečné podpoře v tehdejších dominantních prohlížečích Internet Explorer se na deset let odmlčelo. Tato technologie začala získávat přízeň před několika lety s knihovnami JavaScriptu, jako je například Raphaël, který přidal programovou podporu záložních verzí pro starší verze IE a trend od té doby zesílil. Všechny moderní verze Internet Explorer a Edge mají podporu pro SVG a existuje silná podpora pro technologii od všech ostatních výrobců prohlížečů, včetně Chrome, Firefox a Safari [26][27][28].

II. PRAKTICKÁ ČÁST

5 POŽADAVKY NA SYSTÉM

Hlavním úkolem této diplomové práce je vyvinout systém pro správu mapy knihovního fondu Knihovny UTB. Po stisknutí tlačítka „Ukázat v regálu“ v knihovnickém systému se zavolá náš systém pro správu a zobrazí vyskakovací okno s názorným zobrazením druhého nebo třetího patra Knihovny UTB (podle typu sbírky), které obsahuje regály s knihami. U regálů se poté vybarví ten, kde se daná kniha nachází. Dále bude ještě ve vyskakovacím okně napsáno, ve kterém patře se kniha nachází a název signatury, podle které budeme poté knihu hledat v regálu. Momentálně má Knihovna UTB systém StackMap, který ale nevyhovuje jejím požadavkům. Hlavní důvod je, že mají jen rastrovou formu mapy a samozřejmě musí za systém platit poplatky v řádu desítek tisíc korun za rok.

Jako systém pro správu mapy byl vybrán open source systém pro správu obsahu WordPress, kde bude správa mapy realizována pomocí pluginu. Mezi hlavní důvody, proč byl WordPress vybrán, je to, že s ním mám bohaté pracovní zkušenosti a má mnoho užitečných funkcí, které budou potřeba k vývoji systému. Zároveň s ním mají i zkušenosti správci Knihovny UTB, protože na WordPressu funguje další řada univerzitních systémů a webů.

5.1 Funkční požadavky

Mezi hlavní funkční požadavky patří:

- Řešení formou open source webové aplikace
- Mapy pater s regály vytvořit pomocí vektorového formátu SVG
- Responzivita webové aplikace a možnost přiblížení mapy u mobilní verze
- Možnost backendové editace rozsahu signatur u regálů
- Následné zařazení signatury do regálu a zobrazení mapy
- Jazykové verze aplikace
- Dodržení/sladění vizuálu s knihovním vyhledávacím rozhraním

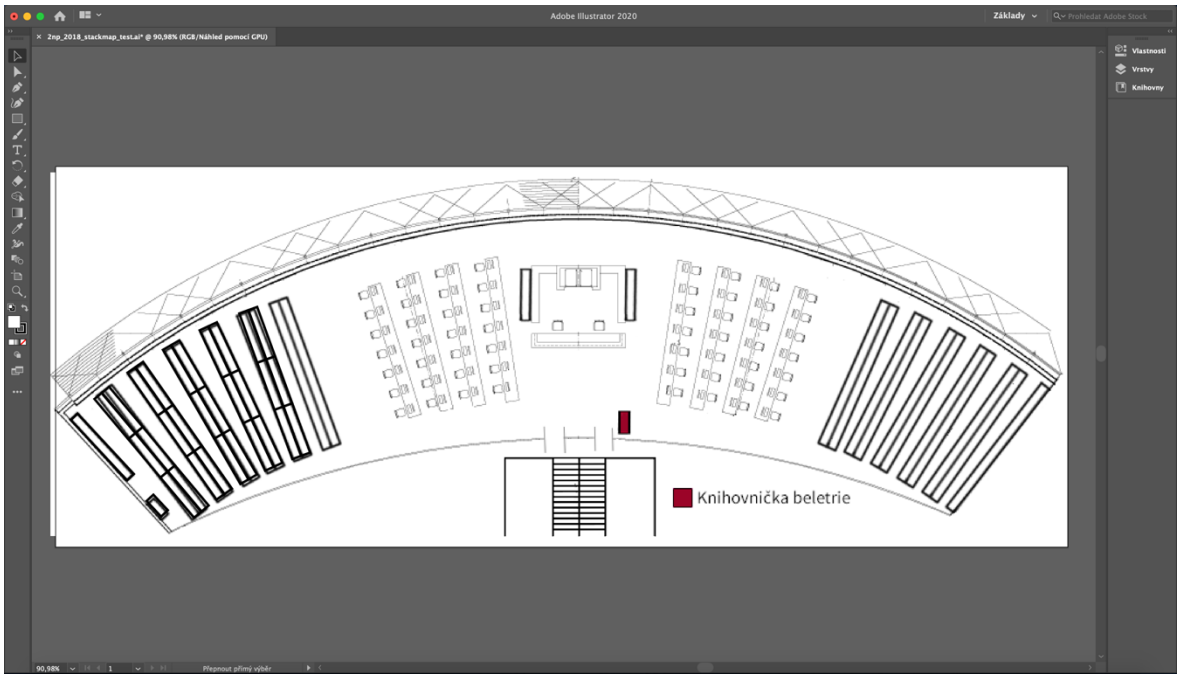
5.2 Nefunkční požadavky

Mezi doplňky funkčních požadavků, které popisují další nezbytné potřeby, náleží:

- Vývoj pomocí pluginu WordPressu
- Ukládání rozsahů signatur do vlastního typu příspěvků s meta boxy
- Zařazení do patra pomocí taxonomie sbírky
- AJAXové uložení a zobrazení rozsahu signatur

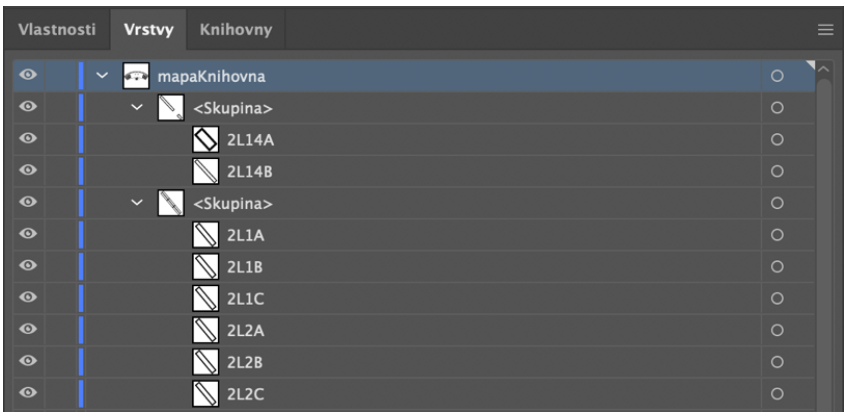
6 VYTVOŘENÍ SVG MAPY

Vývoj systému bude započat vytvořením pater Knihovny UTB ve formátu SVG. K překreslení map poslouží jakýkoli vektorový program, mezi které patří například Adobe Illustrator, Sketch nebo CorelDraw [29]. Do programu je nahrána předloha mapy v rastrovém formátu JPG a pomocí nástroje obdélník je mapa překreslena do vektorové formy (Obrázek č. 5).



Obrázek č. 5 - Překreslování mapy v Adobe Illustrator

Po překreslení všech regálů byly zařazeny sousedící regály do skupin, aby v tom byl přehled a každému regálu bylo přiřazeno jedinečné ID, např.: 2L14A nebo 2L14B (Obrázek č. 6). Názvy ID byly převzaty z minulého systému StackMap. Tento krok je při vytváření mapy nejdůležitější, protože při ukládání rozsahu do systému se zároveň uloží i ID regálu a následně na frontendové straně aplikace při vyvolání rozsahu se daný regál obarví.



Obrázek č. 6 - Spojení regálů do skupin a přiřazení ID

7 PLUGIN UTB LIBRARY MAP

Správa mapy je udělána pomocí pluginu a tím bude následný kód oddělený od jádra WordPressu, který se následně může aktualizovat bez ztráty našich funkcí. Struktura pluginu bude vytvořena pomocí webové aplikace WordPress Plugin Boilerplate Generator. Jak je zmíněno v teoretické části, tak generátor nám vytvoří základní složky a soubory, které plugin vyžaduje k instalaci. Tím nám ušetří spoustu práce při vytváření daných souborů. Na webové prezentaci https://wppb.me se vyplní základní informace o našem pluginu (jméno pluginu, informace o autorovi, URL a slug pluginu). Kliknutím na tlačítko „Build plugin“ se vygeneruje a následně stáhne ZIP soubor, který obsahuje onu zmiňovanou základní strukturu pluginu. V následujících sekcích této kapitoly je popsán postup vývoje pluginu (workflow) [18].

7.1 Vývoj pluginu

Plugin bude vyvíjen za použití základních funkcí WordPressu, jako je např. vlastní typ příspěvku, meta boxy, taxonomie nebo WordPress AJAX. V první řadě bude vyvíjena administrátorská část, kde bude samotné zadávání a přehled rozsahů signatur. Poté bude vyvíjena frontendová část webu, kde bude už jen samotný výpis ve formě vybarveného regálu.

7.1.1 Vlastní typ příspěvku

Vlastní typ příspěvku neboli Custom Post Type je jedna z nejdůležitějších funkcí WordPressu. Bude registrován typ příspěvku druhého a třetího patra, kde následně budou ukládány data o rozsazích signatur pomocí meta boxů, ale o tom později. V kořenu pluginu se nachází soubor *utb-library-map.php*, kde bude registrován vlastní typ příspěvku, ale i meta boxy a taxonomie. Níže je přidán kód registrace vlastního typu příspěvku druhého patra, třetí patro bude mít totožné nastavení, jen bude mít jiné popisky a post type klíč.

```
1 function register_second_floor() {
2    register_post_type( 'second_floor',
3         array(
4             'labels' => [
5                 'name' => __( 'Second floor', 'utb-library' ),
6                 'all_items' => __( 'All ranges', 'utb-library' ),
7                 'edit_item' => __( 'Edit range', 'utb-library' ),
8             ],
9             'menu_icon' => 'dashicons-list-view',
```

```
10              'menu_position' => 81,
11              'public' => false,
12              'publicly_queryable' => true,
13              'show_ui' => true,
14              'has_archive' => false,
15              'rewrite' => false,
16              'supports' => array('title'),
17              'capabilities' => array(
18                  'create_posts' => 'do_not_allow'
19                ),
20              'map_meta_cap' => true
21          )
22      );
23  }
24  add_action( 'init', 'register_second_floor' );
```

Popis nastavení typu příspěvku:

- labels – popisy tlačítek v administraci, budou využity 2 popisky, kde se jedná o samotný název (Second floor) a zobrazení všech rozsahů (All ranges)
- menu_icon – výběr ikony z předem definovaných ikon na adrese https://developer.wordpress.org/resource/dashicons/
- menu_position – zde je nastaveno vysoké číslo, aby byl typ příspěvek úplně na spodu v levém sloupci administrace
- public – nastavuje se true nebo false a znamená to, jestli mají být příspěvky vypisovány na frontendové části webu (šabloně)
- publicly_queryable – znamená to povolení výpisu příspěvků, např. v pluginu, tady musí být true, jinak by se data nevypsala a nemohly by se porovnávat rozsahy
- show_ui – zobrazení typu příspěvku v menu
- has_archive – archiv příspěvků na frontendu
- rewrite – možnost vlastní URL, v našem případě false, protože naše data nepotřebují vlastní URL
- supports – zde se nastavuje, jaké boxy a nastavení mají být na samotné stránce administrace příspěvku
- capabilities – u capabilities je zakázáno vytvoření nového rozsahu z přehledu rozsahů, nový rozsah půjde přidávat pouze přes interaktivní SVG mapu patra knihovny
- map_meta_cap – možnost editace i ze seznamu rozsahů a nejen z interaktivní mapy

7.1.2 Taxonomie sbírky

Aby bylo jasné, do jakého patra má být zařazena signatura, tak je k tomu potřeba funkci taxonomií, což jsou v podstatě kategorie. V našem případě to budou názvy sbírek a podle nich se pozná, jestli signatura patří do druhého nebo třetího patra. Níže je ukázána registrace taxonomie, bude zase jak pro druhé, tak pro třetí patro. Lišit se budou jen v klíči a v názvu objektu, ke kterému budou připojeny.

```
function floor_custom_taxonomy_init() {
    register_taxonomy(
        'secondfloor',
        'second_floor',
        array(
            'label' => __( 'Floor names' ),
            'rewrite' => false,
            'hierarchical' => false,
            'show_in_menu' => true
        )
    );

    register_taxonomy(
        'thirdfloor',
        'third_floor',
        array(
            'label' => __( 'Floor names' ),
            'rewrite' => false,
            'hierarchical' => false,
            'show_in_menu' => true
        )
    );
}
add_action( 'init', 'floor_custom_taxonomy_init' );
```

Popis nastavení taxonomie:

- secondfloor/thirdfloor – klíč taxonomie
- second_floor/third_floor – klíč typu příspěvku, ke kterému je taxonomie přiřazena
- label – název taxonomie v menu administrace
- hierarchical – možnost vytvoření vnořených kategorií, v našem případě není potřeba
- show_in_menu – zobrazení celé taxonomie v menu

7.1.3 Meta boxy

Poslední věcí, která se bude registrovat v souboru *utb-library-map.php* v kořenovém adresáři jsou meta boxy. Meta boxy jsou uživatelská pole, kde je možnost ukládat další informace k příspěvku (rozsahu). Do meta boxů bude ukládáno ID rozsahu z SVG mapy, začáteční a koncový rozsah daného regálu. Na následujícím kódu je ukázáno, jak se meta box registruje. Jelikož jsou potřeba dva typy příspěvků (druhé a třetí patro), tak je využit cyklus *foreach* (opakuje blok kódu pro každý prvek v poli), aby se nemusely vypisovat meta boxy pro každý typ příspěvku.

```
1  public static function add() {
2      $metas = ['second_floor','third_floor'];
3      foreach ($metas as $meta) {
4          add_meta_box(
5              'range_id',
6              'Range ID',
7              [self::class, 'rangeID'],
8              $meta
9          );
10
11         add_meta_box(
12             'range_from',
13             'Range from',
14             [self::class, 'rangeFrom'],
15             $meta
16         );
17
18         add_meta_box(
19             'range_to',
20             'Range to',
21             [self::class, 'rangeTo'],
22             $meta
23         );
24     }
25 }
```

Jak je zmíněno, jsou tu 3 meta boxy, kde první s klíčem *range_id* přidává uživatelské pole, kde se bude ukládat předávané ID z SVG mapy. Dále tu je počáteční rozsah s klíčem *range_from* a koncový rozsah *range_to*. Na druhém místě ve funkci *add_meta_box* jsou popisky meta boxů. O řádek níže je callback funkce, která zavolá upravený input pro zadávání

rozsahu. Na posledním řádku funkce *add_meta_box* je název typu příspěvku, ke kterému je box přiřazen.

7.1.4 Definice rozsahů přes SVG mapu

Všechen kód bude v souborech v adresáři *admin*, který se nachází v kořenu pluginu. Nejprve budou definovány v souboru *class-utb-library-map-admin.php* položky v menu, kde na následných stránkách, které budou vytvořeny ve složce *partials* v adresáři *admin*, bude přidána mapa patra a rozsahy signatur. Každé patro bude mít svoji stránku na zadávání rozsahů, třetí soubor se bude týkat testování zařazení signatur. Na ukázku je zobrazen kód pouze pro přidání hlavní položky menu. Další submenu položky mají stejnou logiku, jen se zobrazí jiné strany (třetí patro nebo testování rozsahu).

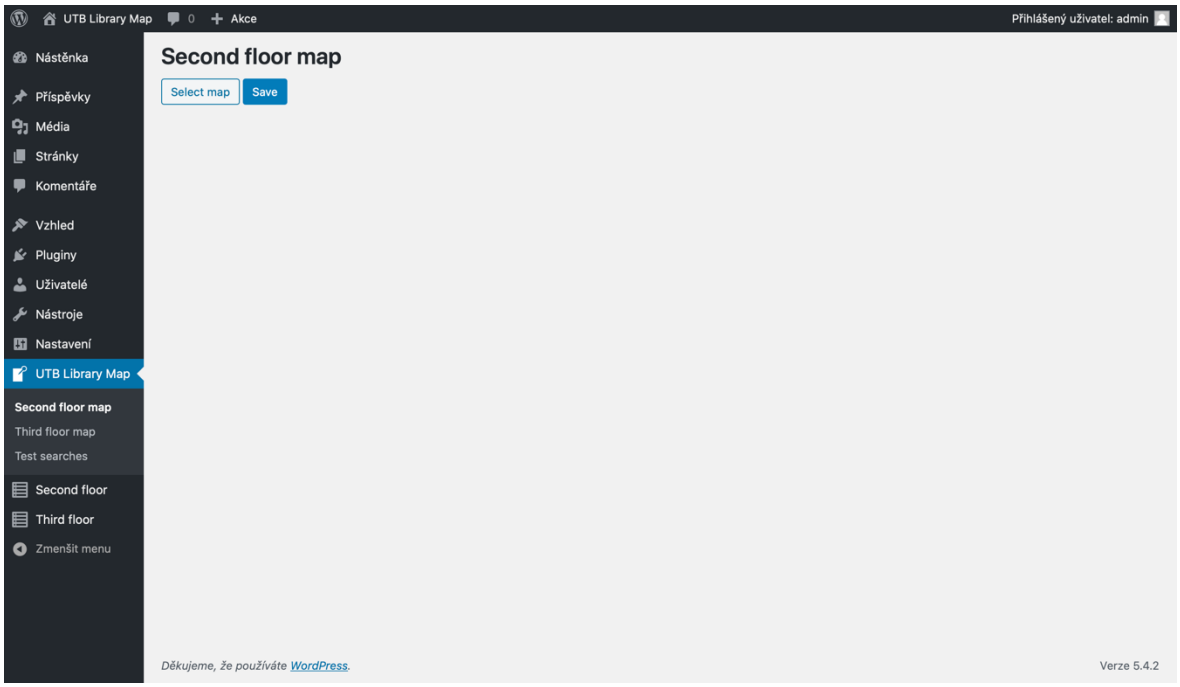
```
1  function custom_admin_menu(){
2      add_menu_page(
3          'UTB Library Map',
4          'UTB Library Map',
5          'administrator',
6          'second-floor-map',
7          array( $this, 'utb_library_map_second' ),
8          'dashicons-pressthis',
9          80
10     );
11 }
12
13 public function utb_library_map_second(){
14     include (plugin_dir_path( __FILE__ ).'partials/utb-library-map-
   admin-display-second.php');
15 }
```

Ve funkci *custom_admin_menu* je vnořená funkce *add_menu_page*, která obsahuje zase některá nastavení. Jedná se například o název položky v menu, název samotné stránky, nastavení práv uživatele, kdo může položku menu vidět, URL název a poté zavolání funkce *utb_library_map_second*, která dá položce menu odkaz na určený PHP soubor ve volané funkci. Poslední dvě položky funkce *add_menu_page* jsou nastavení ikony menu a pozice v levém sloupci administrace.

Tím je splněno vytvoření stránek v menu, kde bude vyvíjeno přidání SVG mapy a následného zadávání rozsahů přes mapu. WordPress nemá ve výchozím nastavení podporu zobrazení SVG souborů a toto je vyřešeno nainstalováním pluginu SVG Support.

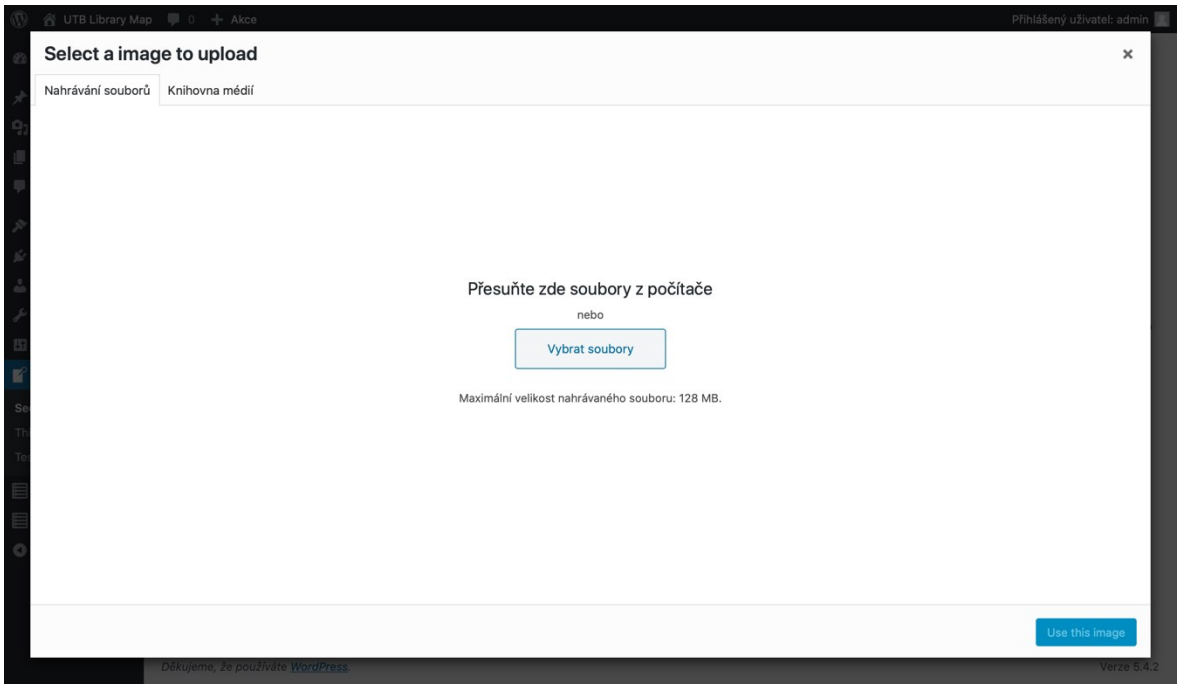
Přidávání mapy bude řešeno pomocí JavaScriptu, který uložíme do souboru *utb-library-map-admin.js*. Na samotné straně (soubor *utb-library-map-admin-display-second.php*), kterou byla v předchozím kroku vytvořena, bude přidán formulář, který vyvolá vyskakovací okno, kde bude možnost nahrát potřebnou SVG mapu.

```
<form method='post' class="fillRect">
    <input id="upload_map_button" type="button" class="button" value="<?php _e( 'Select map' ); ?>" />
    <input type='hidden' name='second_map_id' id='second_map_id' value='<?php echo get_option( 'second_map_selector_id' ); ?>'>
    <input type="submit" name="submit_map_selector" value="Save" class="button-primary">
    <div class='image-preview-wrapper'>
        <?php
        if (!empty(get_option( 'second_map_selector_id' ))) {
            $svg_file = file_get_contents(wp_get_attachment_url( get_option( 'second_map_selector_id' ) ));
            echo "<div class='svg-file'>" . $svg_file . "</div>";
        }
        ?>
    </div>
</form>
```



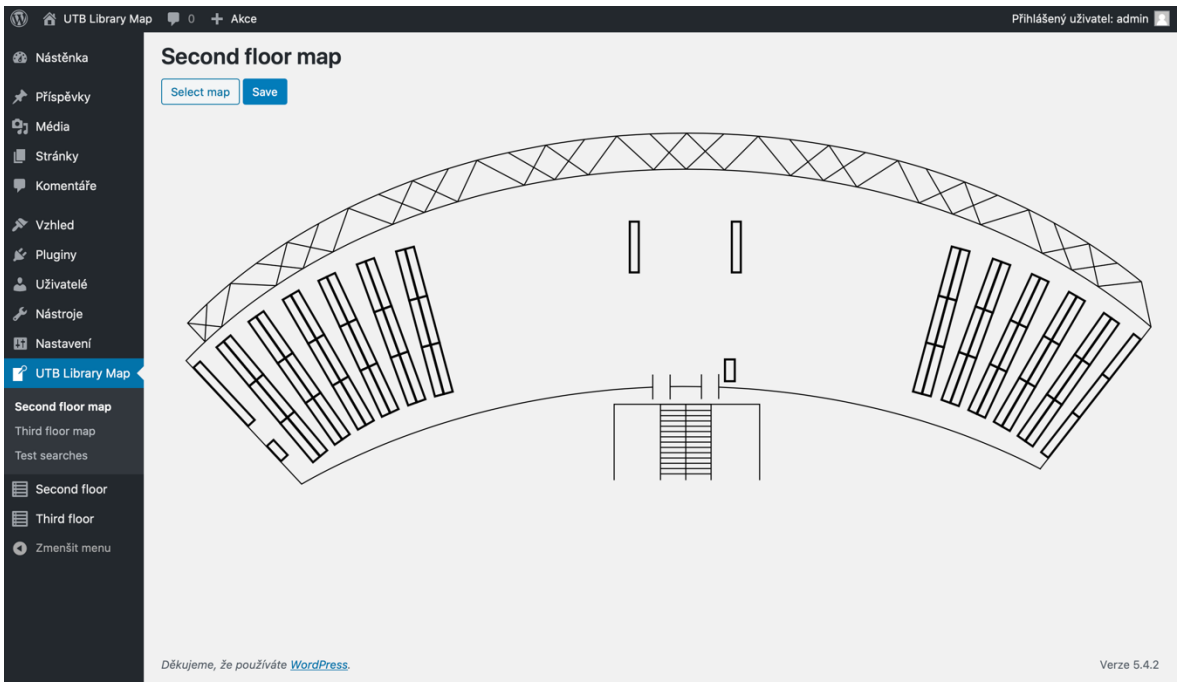
Obrázek č. 7 - Přidání SVG mapy

Tímto je hotovo nahrání mapy do systému, kde na obrázku č. 7 lze vidět, jakým způsobem se mapa nahrává. Po stisknutí tlačítka „Select map“ se zobrazí vyskakovací okno (Obrázek č. 8). Následně tlačítkem „Vybrat soubory“ je nahrána z počítače potřebná mapa. Tlačítkem „Use this image“ se mapa vybere a zobrazí se náhled mapy (Obrázek č. 9).



Obrázek č. 8 - Výběr souboru s mapou

Po výběru mapy je důležité ji potvrdit tlačítkem „Save“ (Obrázek č. 9).



Obrázek č. 9 - SVG mapa 2. patra

Nyní je popsáno samotné přidávání a editací rozsahů. To bude probíhat tak, že po kliknutí na regál, kde má být zadán rozsah, se zobrazí vyskakovací okno s možností zadat název rozsahu, počáteční a koncový rozsah. Dále se kromě zadaných hodnot uloží i ID regálu, které se vezme z SVG mapy kliknutého regálu a bude to jedinečný identifikátor rozsahu. Přidání a editaci bude řešena formou AJAXu, aby se stránka po každém uložení rozsahu nemusela znovu načítat. K načtení a uložení rozsahu poslouží dvě funkce, které jsou popsány níže.

Nejprve bude přidám do souboru *utb-library-map-admin-display-second.php* kód, který definuje vyskakovací okno, které se zobrazí po kliknutí na regál.

```
<div id="myModal" class="modal" data-key="secondFloor">
    <div class="modal-content">
        <div class="closeDiv">
            <h2>Update Call Number Range</h2>
            <span class="close">&times;</span>
        </div>
        <div class="boxInput">
            <input type="hidden" id="rangeID" name="rangeID">
            <div class="inputRange ml-0">
                <label for="rangeName">Range name</label>
                <input type="text" id="rangeName" name="rangeName" placeholder="Range name">
            </div>
            <div class="inputRange">
                <label for="inputRange">Range from</label>
                <input type="text" id="rangeFrom" name="rangeFrom" placeholder="Range from">
            </div>
            <div class="inputRange mr-0">
                <label for="inputRange">Range to</label>
                <input type="text" id="rangeTo" name="rangeTo" placeholder="Range to">
            </div>
            <button type="button" class="btnAdd button-primary" id="btnClick">Update</button>
        </div>
    </div>
</div>
```

Funkce *load_range_second_floor()* vyhledává rozsah z vlastního typu příspěvku s názvem *second_floor* podle meta boxu ID rozsahu (*_range_id*). Tato funkce bude volána AJAXovým načítáním po kliknutí na regál v SVG mapě, kde následně funkce pošle ve formě objektu potřebná data na zobrazení.

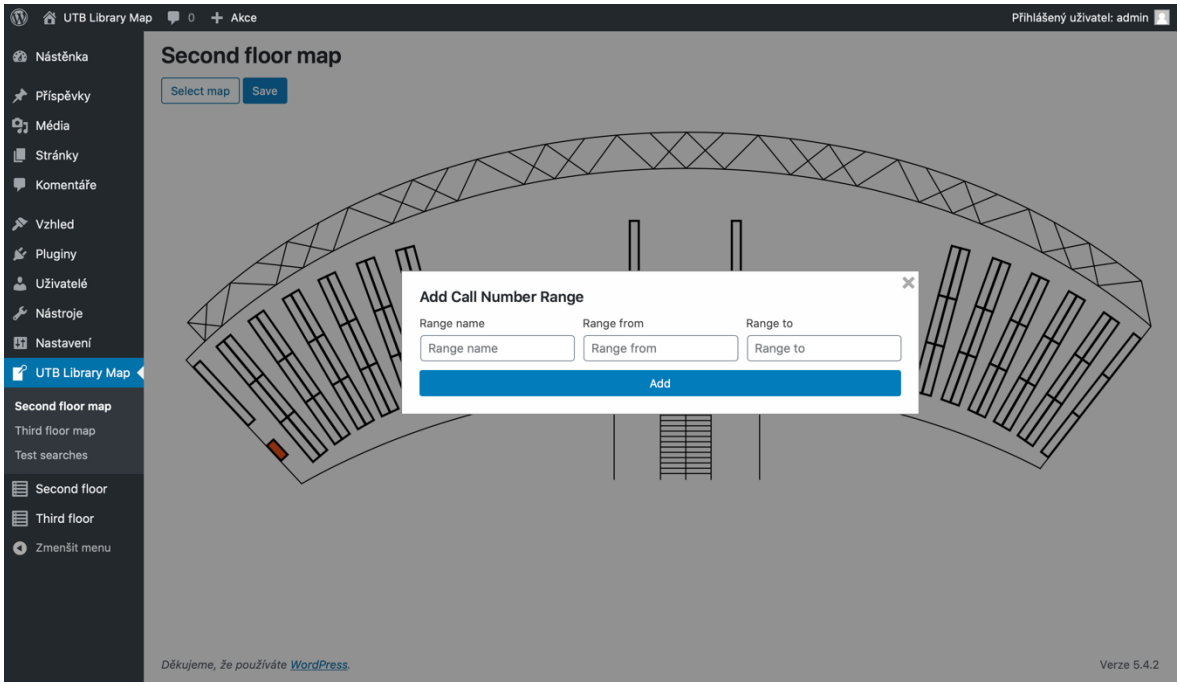
```
function load_range_second_floor() {
    $object = new stdClass();
    $rangeID = $_POST['rangeID'];

    $args = array(
        'post_type'   => 'second_floor',
        'meta_key'    => '_range_id',
        'meta_value'  => $rangeID,
        'fields'      => 'ids'
    );

    $query = new WP_Query($args);

    if ($query->have_posts()) {
       $ajaxposts = get_posts($args);
       $id = $ajaxposts[0];
       $object->rangename = get_the_title($id);
       $object->rangefrom = get_post_meta($id, '_range_from', true);
       $object->rangeto = get_post_meta($id, '_range_to', true);
       echo json_encode($object);
    } else {
       echo json_encode("notexist");
    }
    exit;
}
```

K zavolání funkce bude potřebova metoda AJAX, kterou lze vidět na následujícím kódu.

```
$(".fillRect rect[id^='2P'],.fillRect
rect[id^='2L']").click(function (event) {
        $("#rangeName").val('');
        $("#rangeFrom").val('');
        $("#rangeTo").val('');

        $(this).addClass('fillColor');
        $("#rangeID").val(event.target.id);
```

```
8           var id = $("#rangeID").val();
9
10          $.ajax({
11              type: 'post',
12              url: ajaxurl,
13              dataType: 'json',
14              data: {
15                  action: 'secondfloorload',
16                  rangeID: id
17              },
18              success: function (response) {
19                  if (response != "notexist") {
20                      $("#rangeName").val(response.rangename);
21                      $("#rangeFrom").val(response.rangefrom);
22                      $("#rangeTo").val(response.rangeto);
23                  } else {
24                      $(".closeDiv h2").html('Add Call Number Range');
25                      $("#btnClick").html('Add');
26                  }
27                  $('#myModal').addClass('d-flex');
28              }
29          });
30  });
```

Zároveň byla omezena možnost přidání rozsahu do systému jen na ty regály, u kterých byly při tvorbě mapy zadány ID a které začínají na 2P nebo 2L. Tím je omezena možnost přidat rozsah na regály, které nemají být v rozsazích. Na kódu výše je vidět, že po kliknutí na potřebný regál se zavolá pomocí AJAXu akce *secondfloorload*, která je definována v souboru *class-utb-library-map.php* ve složce *includes*. Akci je přiřazena funkce *load_range_second_floor(),* která předá do vyskakovacího okna název rozsahu, počáteční a koncový rozsah. Jelikož zatím nejsou přidány žádné rozsahy, zobrazí se prázdné hodnoty a jen se změní název vyskakovacího okna na „Add Call Number Range“ a to samé u tlačítka, kde se změní „Update“ na „Add“ (Obrázek č. 10).



Obrázek č. 10 - Definice rozsahu k vybranému regálu

Načítaní rozsahů do vyskakovacího okna je vyřešeno, ale je potřeba ještě vymyslet přidání rozsahu do systému. Přidání rozsahu bude opět řešeno přes metodu AJAX, kde po kliknutí na tlačítko „Add“ se zavolá funkce *add_range_second_floor()*.

```
1  public function add_range_second_floor(){
2          $postTitle = $_POST['rangeName'];
3          $rangeID = $_POST['rangeID'];
4          $rangeFrom = $_POST['rangeFrom'];
5          $rangeTo = $_POST['rangeTo'];
6          $metaToStore = [];
7          $metaToStore['_range_id'] = $rangeID;
8          $metaToStore['_range_from'] = $rangeFrom;
9          $metaToStore['_range_to'] = $rangeTo;
10         $args = array(
11             'post_type' => 'second_floor',
12             'meta_query' => array(
13                 array(
14                     'key' => '_range_id',
15                     'value' => $rangeID,
16                     'compare' => '='
17                 )
18             )
19         );
```

```
20          $query = new WP_Query($args);
21          $post_id = null;
22
23          if ($query->have_posts()) {
24              $post_id = $query->post->ID;
25              $query->post->post_title = $postTitle;
26              wp_update_post($query->post);
27              foreach ($metaToStore as $metaKey=>$value) {
28                  update_post_meta($post_id, $metaKey, $value);
29              }
30              echo json_encode("exists");
31          } else {
32              $new_post = array(
33                  'post_title' => $postTitle,
34                  'post_status' => 'publish',
35                  'post_type' => 'second_floor'
36              );
37              $post_id = wp_insert_post($new_post);
38              foreach ($metaToStore as $metaKey=>$value) {
39                  add_post_meta($post_id, $metaKey, $value);
40              }
41          }
42          $terms = get_terms([
43              'taxonomy' => 'secondfloor',
44              'hide_empty' => false,
45          ]);
46          if ( $terms && ! is_wp_error( $terms ) ) :
47              $selectedTerm = array();
48              foreach ( $terms as $term ) {
49                  $selectedTerm[] = $term->term_id;
50              }
51          wp_set_post_terms( $post_id, $selectedTerm, 'secondfloor');
52          endif;
53          exit;
54      }
```

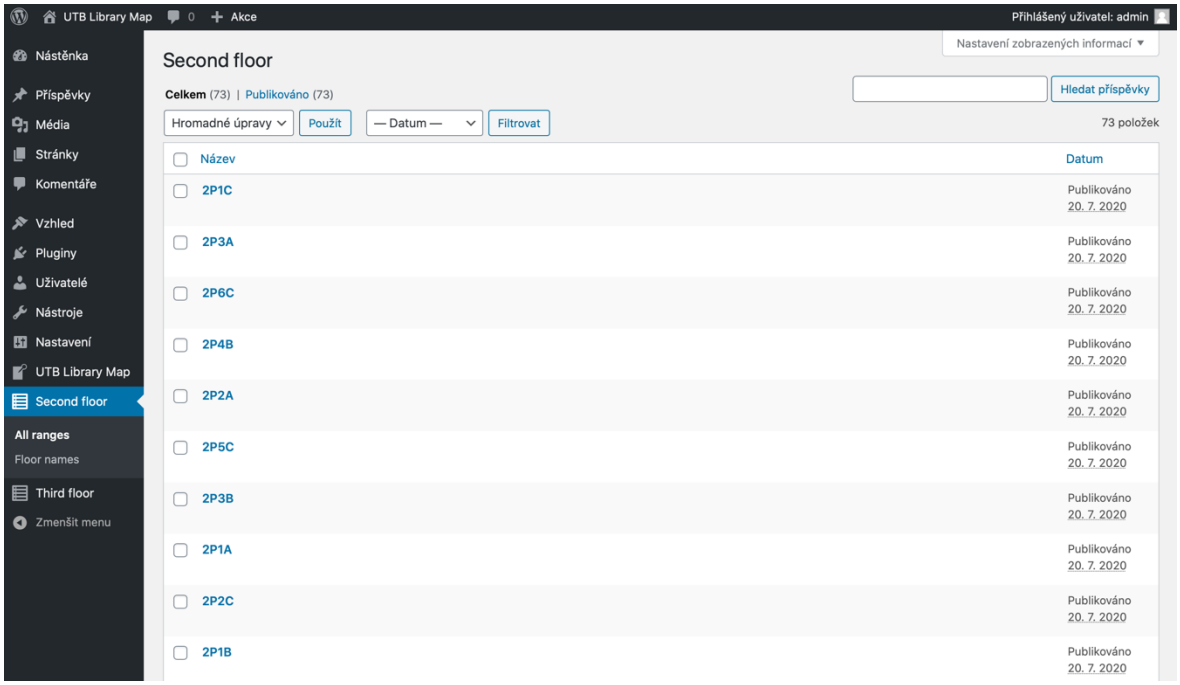
Tato funkce je jedna z nejdelších v systému, protože musí obsahovat více funkcí, jako je přidání nového rozsahu, případně jen aktualizace, když bude ID regálu existovat. Dále také automatické vybrání všech taxonomií, v našem případě sbírek. Toto vše se musí nacházet v jedné funkci.

Ještě stojí za zmínku AJAXová metoda, která zavolá funkci *add_range_second_floor()*. Po stisknutí tlačítka se předají potřebné parametry, jakou je název rozsahu, ID, začáteční a koncový rozsah. Po úspěšném provedení funkce se skryje vyskakovací okno.

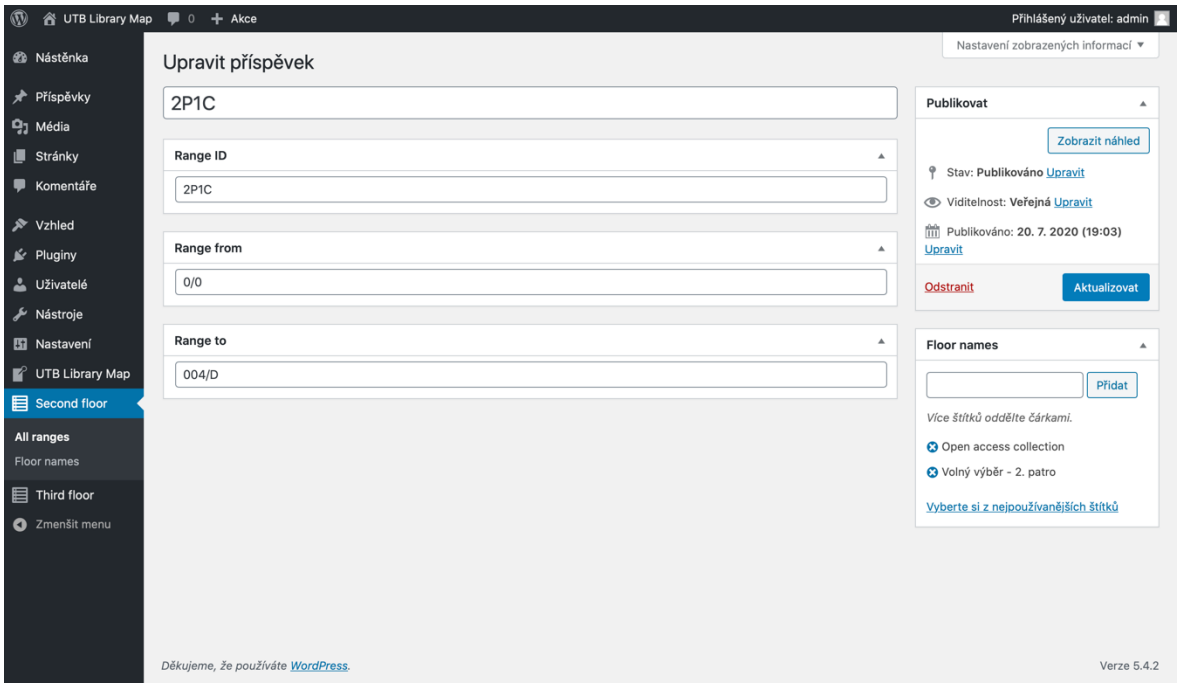
```
$("#btnClick").on("click", function () {
    var name = $("#rangeName").val();
    var id = $("#rangeID").val();
    var from = $("#rangeFrom").val();
    var to = $("#rangeTo").val();

    $.ajax({
        url: ajaxurl,
        type: 'post',
        dataType: 'json',
        data: {
            action: 'secondflooradd',
            rangeName: name,
            rangeID: id,
            rangeFrom: from,
            rangeTo: to
        },
        success: function (data) {
             removeModal();
        }
    });
});
```

Momentálně je možnost v administraci přidávat, editovat nebo mazat rozsahy. Tímto je vyřešena administrační část pluginu. Na obrázku č. 11 a 12 je ukázán seznam všech rozsahů a detail rozsahu, kde se rozsah, ale taky i taxonomie edituje. Taxonomie se nachází v pravém sloupci (Floor names).

Nyní bude řešena frontendová část pluginu, kde se bude požadovaná signatura zařazovat do regálu a při správném zařazení bude regál obarvan.



Obrázek č. 11 - Seznam všech rozsahů



Obrázek č. 12 - Detail rozsahu

7.1.5 Frontendová strana kódu

Veškerý kód, který se týká frontendové části pluginu bude ukládán do adresáře public v kořenu pluginu a následně do souboru *class-utb-library-map-public.php*. Kód pro frontendovou část by šel dát do jedné funkce, ale pro větší přehlednost je kód rozdělen do pěti funkcí s názvem *get_parameters()*, *lang_versions()*, *query_signatures()*, *compare_by_alphabet()* a *render_map()*.

Funkce get_parameters()

Celý systém bude fungovat na předávání parametrů z URL adresy. URL adresa bude ve formátu https://jmeno_domeny.cz/?floor=nazev_patra&signature=nazev_signatury&lang=jazyk. Ve funkci *get_parameters()* se budou brát parametry z URL adresy a následně podle toho bude vybrána mapa druhého nebo třetího patra.

```
1 if (isset($_POST['run_test'])){
2     $this->params['signature'] = $_POST['signature_name'];
3     $this->params['floor'] = $_POST['floor_names'];
4     $this->test_search = true;
5 } else {
6     $url = 'https://'.$_SERVER['HTTP_HOST'].$_SERVER['REQUEST_URI'];
7     $url_components = parse_url($url);
8     parse_str($url_components['query'], $this->params);
9 }
```

Do funkce *get_parameters()* bylo přidáno i získání parametrů z testovacího hledání, které se nachází v administraci WordPressu. Na kódu je vidět, že když se zmáčkne tlačítko v administraci se jménem „run_test“, tak se uloží parametry signatury a sbírky z *inputu* a *selectu*.

Když se ale nepoužije testovací hledání z administrace, tak na frontendové straně budou získány parametry z URL adresy pomocí PHP funkce *parse_url*, která rozloží URL adresu. Naše parametry budou v řetězci, který se nazývá *query*, a ten je následně ještě rozdělen funkcí *parse_str*.

Dále jsou do proměnných *$second_floor* a *$third_floor* uloženy všechny taxonomie, které patří k druhému, případně třetímu patru.

```
10 $second_floor = get_terms( array(
11     'taxonomy' => 'secondfloor',
12     'hide_empty' => false
13 ));
```

```
14 $third_floor = get_terms( array(
15     'taxonomy' => 'thirdfloor',
16     'hide_empty' => false
17 ));
```

V následujících cyklech budou porovnávány taxonomie (v našem případě sbírky) s parametrem *floor*, který byl obdržen z URL adresy. Při shodě taxonomie s parametrem se cyklus ukončí a určí se, zda se jedná o druhé nebo třetí patro. Tato informace o patře se poté uloží do proměnné *$floor*.

```
18 if (!is_wp_error($second_floor)) {
19     foreach ( $second_floor as $term ) {
20         if ($term->name == $this->params['floor']){
21             $floor = 'second';
22             break;
23         }
24     }
25 }
26 if (!is_wp_error($third_floor)) {
27     foreach ( $third_floor as $term ) {
28         if ($term->name == $this->params['floor']){
29             $floor = 'third';
30             break;
31         }
32     }
33 }
```

Poslední částí funkce je vybrání mapy daného patra a uložení typu příspěvku do proměnné *$args*. Které patro bude vybráno poznáme podle proměnné *$floor*, do které byla v předchozí ukázce kódu uložena informace, zda se jedná o druhé nebo třetí patro.

```
34 if($this->floor == 'second'){
35     $this->svg_file = file_get_contents(wp_get_attachment_url(
   get_option( 'second_map_selector_id' ) ));
36     $this->args = array(
37         'post_type'   => 'second_floor',
38         'fields'      => 'ids',
39         'posts_per_page' => -1
40      );
41 } elseif ($this->floor == 'third') {
42     $this->svg_file = file_get_contents(wp_get_attachment_url(
   get_option( 'third_map_selector_id' ) ));
```

```
43      $this->args = array(
44          'post_type'   => 'third_floor',
45          'fields'      => 'ids',
46          'posts_per_page' => -1
47      );
48  }
```

Funkce query_signatures()

Ve funkci *query_signatures()* je obdržena z předchozí funkce proměnná *$args*, kde je uloženo nastavení pro výpis signatur pomocí funkce *get_posts()*.

```
1  $posts = get_posts($this->args);
2  if ($posts) {
3      $this->show_button = true;
4      foreach ( $posts as $id ) {
5          $first = $testArray[0] = trim(preg_replace('/\s*\([^)]*\)/',
   '', $this->params['signature']));
6          $from = $testArray[1] = get_post_meta($id, '_range_from',
   true);
7          $to = $testArray[2] = get_post_meta($id, '_range_to', true)
   . 'žžžŽŽŽ999';
8
9          usort($testArray, array($this,'compare_function'));
10
11         if ((array_values($testArray)[1] == $first)){
12             echo '<style>rect[id^="'. get_post_meta($id,
   '_range_id', true).'"]{fill:#e65014 !important;}</style>';
13             $finalFrom = get_post_meta($id, '_range_from', true);
14             $finalTo = get_post_meta($id, '_range_to', true);
15         } else {
16             $testArray = NULL;
17         }
18         wp_reset_postdata();
19     }
20 }
```

Pomocí cyklu *foreach* se projdou všechny rozsahy signatur, kde při každém průchodu se do testovacího pole bude ukládat název hledané signatury, začáteční a koncový rozsah. Toto pole je poté seřazeno pomocí PHP funkce *usort*. Funkce *usort* seřadí testovací pole pomocí předem definované funkce seřazení s názvem *compare_function()*. Pomocí podmínky *if* se

pak ověří, zda je na pozici 1 v testovacím poli hledaná signatura. V případě, že bude signatura na pozici 1, tzn. že na pozici 0 bude začáteční rozsah a na pozici 2 bude koncový rozsah, tak se daný regál obarví pomocí ID regálu za použití CSS vlastnosti *fill*.

Ve funkci je ještě jedna podmínka pro testovací hledání, kdy na testovací stránce v administraci se po provedení hledání kromě mapy a signatury zobrazí ještě začáteční a koncový rozsah. Ve funkci se ještě nachází samotné vypsaní mapy pomocí metody *echo*.

7.2 Implementace na knihovnický systém

V době psaní diplomové práce (srpen 2020) byl WordPress se systémem pro správu mapy knihovního fondu nainstalován na koupeném serveru s vlastní doménou a zaměstnanci Knihovny UTB poskytli pro testovací napojení kopii knihovnického systému na doméně https://vufind-test.katalog.k.utb.cz. V reálném provozu bude WordPress s pluginem spuštěn na serverech UTB a knihovnický systém bude na doméně https://vufind.katalog.k.utb.cz.

Ještě, než se začne s napojováním na knihovnický systém, tak je potřeba vytvořit úplně základní šablonu WordPressu, která bude mít jen dva základní soubory nutné pro instalaci šablony. Jedná se o soubor *style.css*, kde jsou informace o šabloně a poté *index.php*. Do souboru *index.php* je přidán následující řádek:

```
1  do_action('show_map');
```

Tím se spustí action hook, který vykoná frontendovou stranu pluginu a výše zmíněných funkcí v kapitole 7.1.5 Frontendová strana kódu.

První krok, který byl požadován po správci knihovnického systému bylo odstranit stávající napojení na minulý systém StackMap a následně přidat atributy signatury, sbírky a jazyka do tlačítka „Ukázat v regále“ (Obrázek č. 13).

Obrázek č. 13 - Přehled jednotek

Jakmile budou parametry přidané k tlačítku, tak se musí přidat do knihovnického systému dva soubory. Jedná se o CSS, který bude mít za úkol nastylovat vyskakovací okno, které se zobrazí pomocí JavaScriptu po kliknutí na tlačítko „Ukázat v regále“.

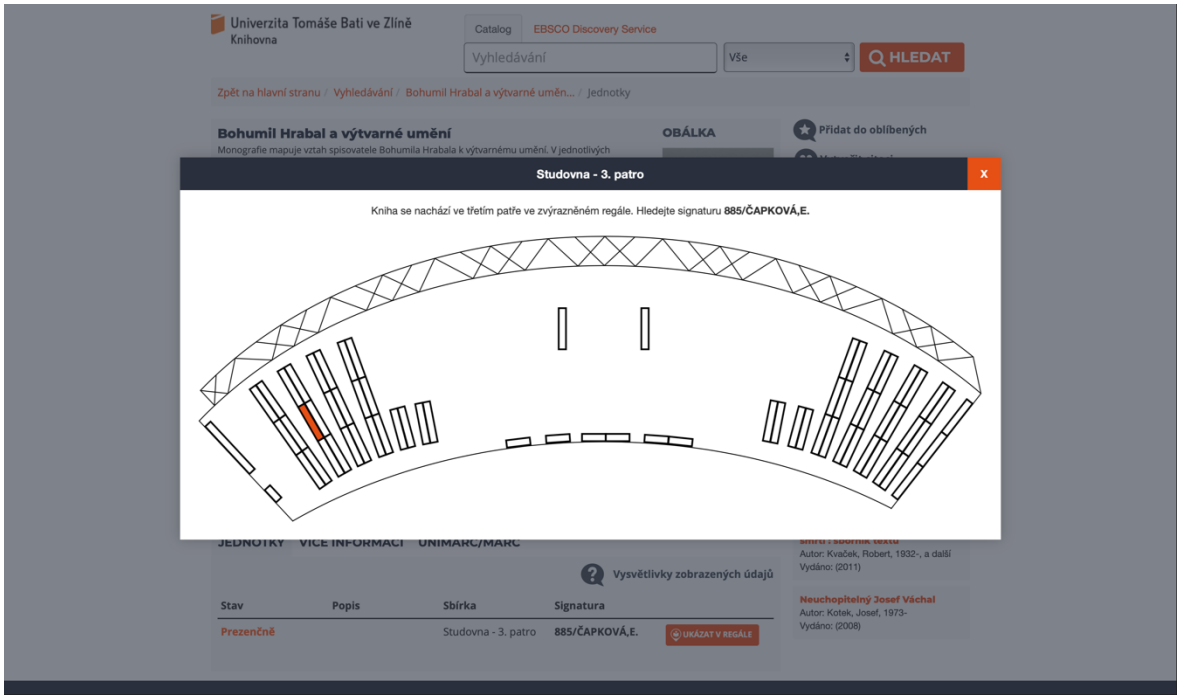
Soubor s JavaScriptem obsahuje mimo kód implementace ještě knihovnu svg-pan-zoom, která se bude zabývat zvětšením, oddálením a defaultním zobrazení SVG mapy. Následující kód se provede po stisknutí tlačítka „Ukázat v regále“. Po stisknutí se uloží do proměnných *signature*, *location* a *lang* právě ty atributy, které jsou potřeba ke správnému zařazení. Jedná se o název signatury, podle které se kniha zařadí do správného regálu a název sbírky, díky které se kniha zase zařadí do správného patra. Ještě je zde atribut obsahující kód jazyku, podle kterého bude zobrazen buď český nebo anglický popisek.

```
1 var signature = $(this).attr('data-callno');
2 var location = $(this).attr('data-location');
3 var lang = $(this).attr('data-lang');
4
5 $.get("https://dolezal.cloud/?floor=" + location + "&signature=" +
  signature + "&lang=" + lang, {}, function (results) {
6     $(results).insertAfter($("#modal")).fadeIn(100);
7 });
```

Žádané proměnné jsou poté přidány do URL adresy našeho systému. Funkcí *get* je získán obsah frontendové části systému pro správu knihovního fondu mapy, který bude vložen do knihovnického systému. Následně se zobrazí vyskakovací okno s mapou patra knihovny, kde se hledaná kniha nachází (Obrázek č. 14).

Jedním z funkčních požadavků na systém správy mapy bylo udělat responzivní verzi systému s možností přiblížení, oddálení a výchozího zobrazení mapy. Reponzivita byla řešena pomocí CSS stylů a metody Media Queries (umožňuje vykreslení stylu podle potřebných podmínek, jako je např. rozlišení obrazovky).

Na obrázku č. 15 je mobilní verze systému s přiblíženou mapou, kde jsou k vidění i potřebná tlačítka na příblížení, oddálení a výchozí zobrazení.

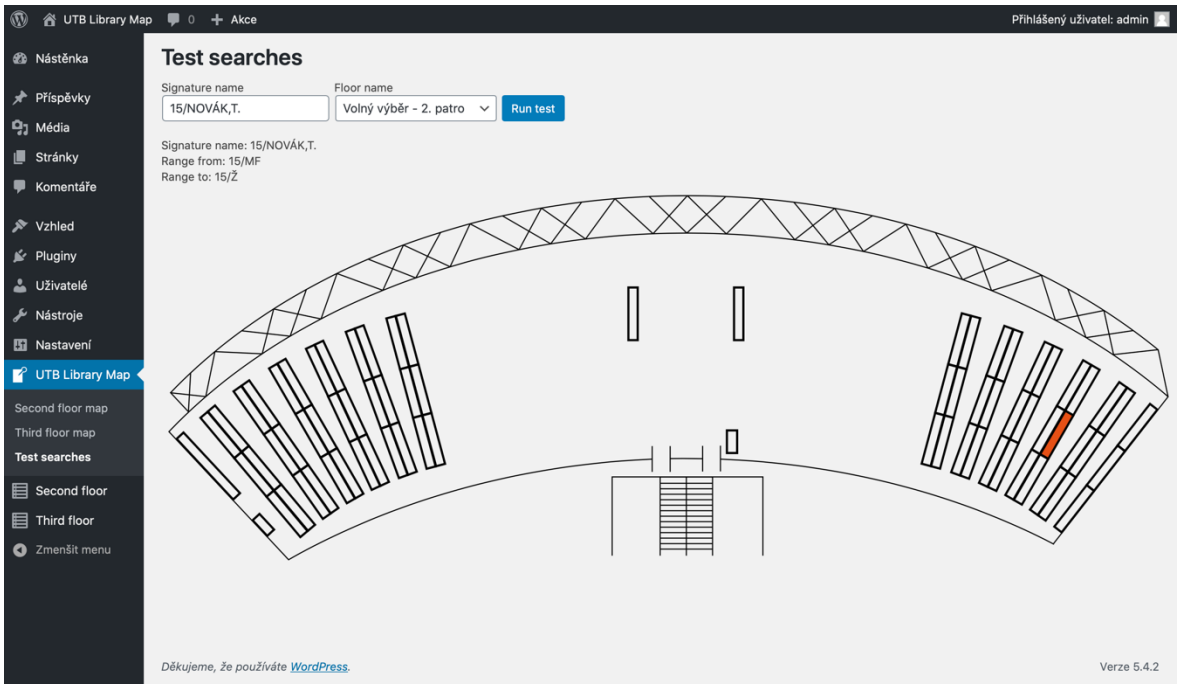


Obrázek č. 14 - Vyskakovacího okno s označeným regálem v SVG mapě

Obrázek č. 15 - Mobilní verze systému

8 TESTOVÁNÍ

Na testování zařazení signatury do rozsahu byla v administraci WordPressu vytvořena samostatná strana (používá funkce z frontendové strany kódu), kde je *input*, do něhož se zadává název signatury a poté ještě *select* s výběrem patra, ve kterém se má signatura hledat. Po zadání signatury, výběru patra a následném kliknutí na tlačítko „Run test“ se zobrazí mapa hledaného patra s vybarveným regálem, dále hledaná signatura a rozsahy, do kterých spadá (Obrázek č. 16).



Obrázek č. 16 - Testovací hledání signatur

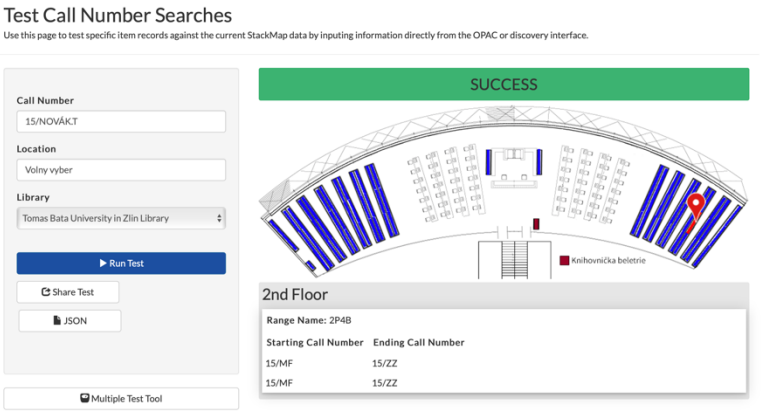
Samotné testování probíhalo ve spolupráci se zaměstnanci Knihovny UTB, kde byly vybrány desítky náhodných signatur a pomocí testovacího hledání byly srovnávány se současným systémem StackMap, který má v administraci podobnou funkci (Obrázek č 17).

Nevýhodou stávajícího testování na StackMapu bylo, že se sbírka musela zadávat ručním vepsáním, kdežto v systému na WordPressu je možnost výběru z předem definovaných sbírek, což při testování ušetřilo čas.

Nová mapa knihovního fondu se rychleji načítá, protože se stahuje menší objem dat, než ze systému StackMap, kde knihovnický systém musel zobrazovat obrázek mapy v rastrovém typu formátu.

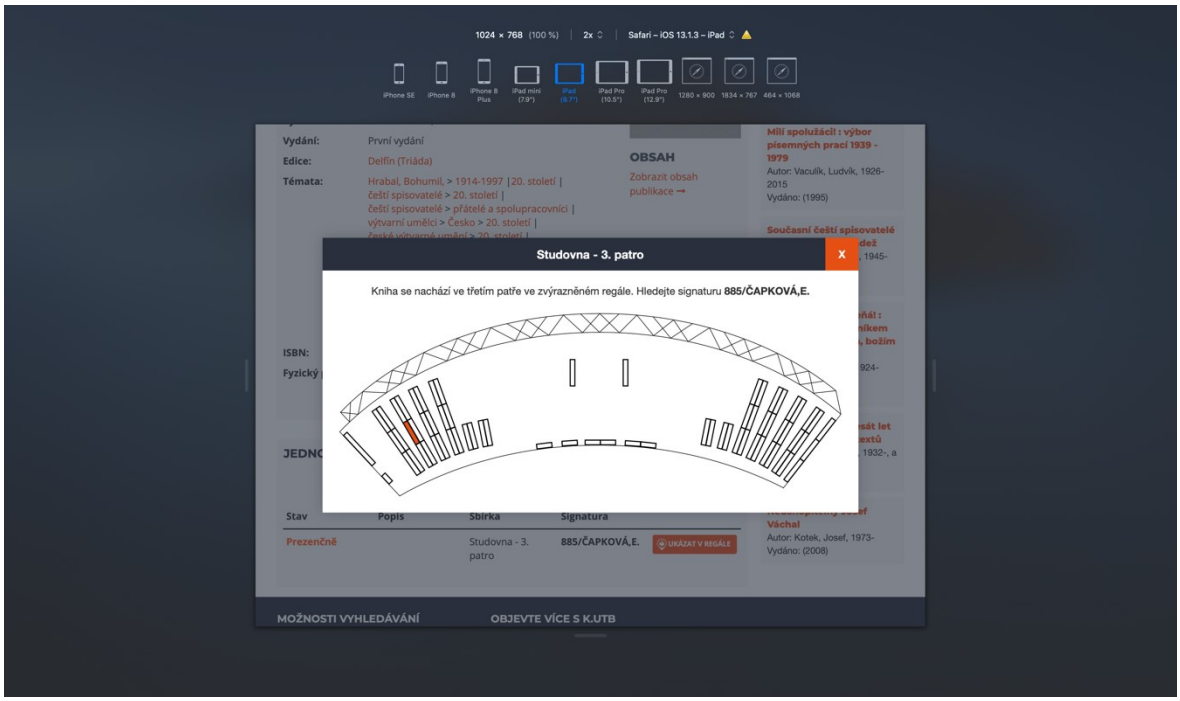
Po desítkách otestovaných signatur, kde se všechny výsledky hledání rozsahů shodovaly se současným systémem testovacího hledání v administraci StackMapu, se může systém mapy

knihovního fondu na WordPressu a jeho funkce pro porovnávání signatur považovat za spolehlivou.



Obrázek č. 17 – Testovací hledání v systému StackMap

Na frontendu aplikace probíhalo testování responzivní verze systému pomocí webového prohlížeče Safari a jeho funkce reagujícího návrhového režimu, kde jsou předdefinované zařízení s různou velikostí rozlišení (Obrázek č. 18).



Obrázek č. 18 - Zobrazení systému pro Apple iPad

ZÁVĚR

Cílem diplomové práce bylo vyvinout systém pro správu mapy knihovního fondu. Systém byl vyvíjena pro reálné použití s napojením na knihovnický systém Knihovny UTB. Za úkol bylo vytvořit systém, který zařadí signaturu do předem definovaných rozsahů s následným označením vybraného regálu na mapě.

V teoretické části byla nejprve provedena rešerše na současné systémy, které mají mapu knihovního fondu. Existují hotová řešení (současný systém Knihovny UTB) a poté vlastní řešení mapy knihovního fondu (Národní technická knihovna a Krajská knihovna Františka Bartoše ve Zlíně).

Po úvodu do blogu a CMS byl vybrán redakční systém WordPress. Dále jsou vypsány základní informace o systému s popisem adresářové struktury systému a jeho funkcí, které byly potřeba při následném vývoji pluginu. U pluginů je zmíněna hlavně struktura a vytvoření vlastního pluginu, ale také jedna z hlavních funkcí WordPressu, což jsou hooky, které se často používají při vývoji pluginu. Šablony jsou další kapitolou, kde je popsáno fungování šablony a její struktura, jejíž znalost je potřeba k vytvoření vlastní šablony. Poslední sekcí kapitoly WordPress jsou vlastní typy příspěvku. Ty jsou zcela zásadní u vývoje pluginu, protože si pomocí nich můžeme vytvořit samostatný typ příspěvku, které tímto způsobem rozdělíme do potřebných sekcí s různým nastavením. U vlastního typu příspěvku je možnost nastavení vlastních meta boxů a taxonomií. Poslední kapitola teoretické části je o technologiích, které využívá systém WordPress a o formátu SVG, který je potřeba k vytvoření mapy.

Praktická část byla hlavně o vytvoření SVG mapy pater knihovny a vývoji pluginu UTB Library Map. Celý systém byl definován požadavky, mezi které hlavně patřilo použití open source systému, vektorového formátu mapy, responzivní verzí a možností editace z backendové strany systému. Celý plugin byl vytvořen pomocí hlavních funkcí WordPressu, které jsou popsány v teoretické části diplomové práce. Po vytvoření pluginu a jeho instalaci proběhlo zadání rozsahů do systému pro každý regál. Posledním bodem vývoje pluginu bylo napojení na reálný knihovnický systém Knihovny UTB. Po úspěšném napojení proběhlo testování, zda se vybraná signatura zařadí do správného regálu, to bylo ověřováno společně se zaměstnanci Knihovny UTB porovnáním se současným systémem StackMap.

Testování systému pro správu mapy knihovního fondu proběhlo úspěšně, kde označené regály hledaných signatur byly totožné s testovacím hledáním z původního systému StackMap.

SEZNAM POUŽITÉ LITERATURY

[1] *Knihovny v České republice* [online]. [cit. 2020-08-09]. Dostupné z: http://www.knihovny.net

[2] BYRON, D. L. a Steve BROBACK. *Blogy: publikuj a prosperuj: blogování pro váš business*. Praha: Grada, 2008. ISBN 9788024720647.

[3] Blog. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-06-01]. Dostupné z: https://cs.wikipedia.org/wiki/Blog

[4] TOONEN, Edwin. What is a permalink? *SEO for beginners • Yoast* [online]. [cit. 2020-06-14]. Dostupné z: https://yoast.com/what-is-a-permalink/

[5] CIRKL, Tomáš. Co je to pingback, trackback a jaký je mezi nimi rozdíl. *WordPress návody, tipy a rady* [online]. [cit. 2020-06-14]. Dostupné z: https://www.wplama.cz/co-je-to-pingback-trackback-a-jaky-je-mezi-nimi-rozdil/

[6] MIŽIKOVÁ, Alena. Čo je Blogroll? *BlogIT - TECHNICKÉ ROZMOTÁVANIE* [online]. [cit. 2020-06-15]. Dostupné z: https://blogit.sk/co-je-blogroll/

[7] BLOGGER.COM VS. BLOG.CZ - SROVNÁNÍ BLOGOVACÍCH SYSTÉMŮ. *Ze skříně* [online]. [cit. 2020-06-15]. Dostupné z: https://zeskrine.blogspot.com/2013/06/bloggercom-vs-blogcz-srovnani.html

[8] Yahoo! 360°. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-08-05]. Dostupné z: https://en.wikipedia.org/wiki/Yahoo!_360°

[9] OPEN SOURCE - SHOPY A REDAKČNÍ SYSTÉMY CMS. *Hosting, Registrace domény, Virtuální servery - VPS - BEST-HOSTING.cz* [online]. [cit. 2020-06-15]. Dostupné z: https://best-hosting.cz/cs/napoveda/joomla-drupal-wordpess-quickcart-shopy-cms

[10] BOIKO, Bob. *Content management bible*. 2nd ed. Indianapolis, IN: Wiley Pub., 2005. ISBN 978-0-764-57371-2.

[11] BARKER, Deane. *Web content management: systems, features, and best practices*. Boston: O'Reilly, 2016. ISBN 978-1-491-90812-9.

[12] JOHNSTON, Mike. How to Choose a CMS in 7 Easy Steps. *CMS Critic* [online]. [cit. 2020-07-15]. Dostupné z: https://www.cmscritic.com/how-to-choose-a-cms-in-7-easy-steps/

[13] *What Is WordPress? Explained for Beginners* [online]. [cit. 2020-07-16]. Dostupné z: https://kinsta.com/knowledgebase/what-is-wordpress/

[14] DOLEŽAL, Tomáš. *Možnosti pokročilých úprav pomocí Wordpress API*. Zlín, 2018. Bakalářská práce. Univerzita Tomáše Bati, Fakulta aplikované informatiky.

[15] KRÓL, Karol. *WordPress 5 Complete: Build beautiful and feature-rich websites from scratch*. Seventh edition. Birmingham: Packt Publishing, 2019. ISBN 978-1-78953-201-2.

[16] MESSENLEHNER, Brian a Jason COLEMAN. *Building Web Apps with WordPress, 2nd Edition*. O'Reilly Media, 2019. ISBN 9781491990087.

[17] *WordPress Plugin Development Cookbook*. Second edition. Birmingham: Packt Publishing, 2017. ISBN 978-1-78829-118-7.

[18] *WordPress Plugin Boilerplate Generator* [online]. [cit. 2020-08-01]. Dostupné z: https://wppb.me

[19] WILLIAMS, Brad, Justin TADLOCK a John James JACOBY. *Professional wordpress plugin development*. 2. Indianapolis: John Wiley, 2020. ISBN 9781119666974.

[20] CIRKL, Tomáš. *Jak u WordPress na vlastní typ příspěvků (Custom Post Types)* [online]. [cit. 2020-07-20]. Dostupné z: https://www.wplama.cz/wordpress-vlastni-typ-prispevku/

[21] *HTML basics - Learn web development | MDN* [online]. [cit. 2020-08-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[22] *Programming PHP: Creating Dynamic Web Pages*. 4. Sebastopol: O'Reilly Media, 2020. ISBN 978-1-492-05413-9.

[23] *What is CSS - A Simple Guide to HTML* [online]. [cit. 2020-08-09]. Dostupné z: http://www.simplehtmlguide.com/whatiscss.php

[24] DOWDEN, Martine a Michael DOWDEN. *Architecting CSS: The Programmer's Guide to Effective Style Sheets*. 1. Apress, 2020. ISBN 978-1-4842-5749-4.

[25] *JavaScript basics - Learn web development | MDN* [online]. [cit. 2020-08-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[26] SVG ON THE WEB [online]. Breaking Borders, 2019 [cit. 2012-07-20]. Dostupné z: https://svgontheweb.com/

[27] LARSEN, Rob. *Mastering SVG: Ace web animations, visualizations, and vector graphics with HTML, CSS, and JavaScript*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78862-674-3.

[28] *Co je Vektorová grafika | Adaptic* [online]. [cit. 2020-08-09]. Dostupné z: https://www.adaptic.cz/znalosti/slovnicek/vektorova-grafika/

[29] *Best Vector Graphics Software* [online]. [cit. 2020-08-08]. Dostupné z: https://www.g2.com/categories/vector-graphics

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UTB	Univerzita Tomáše Bati
SVG	Scalable Vector Graphics
PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
XHTML	Extensible Hypertext Markup Language
CMS	Content Management System
SEO	Search Engine Optimization
URL	Uniform Resource Locator
RSS	Rich Site Summary
ATOM	Atom Syndication Format
XML	Extensible Markup Language
SQL	Structured Query Language
WCMS	Web Content Management System
WYSIWIG	What you see is what you get
MySQL	My Structured Query Language
GPL	General Public License
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
API	Application Programming Interface
MVC	Model-view-controller
CRON	Commands Run Over Night
IE	Internet Explorer
PDF	Portable Document Format
GIF	Graphics Interchange Format

JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
UI	User Interface
ID	Identification

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: SOUBORY ZDROJOVÉHO KÓDU NA CD

Zdrojový kód systému je na přiloženém CD.