Možnosti pokročilých úprav pomocí Wordpress API

Tomáš Doležal

Bakalářská práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Doležal**
Osobní číslo:	**A14081**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Možnosti pokročilých úprav pomocí Wordpress API**
Téma anglicky:	**The Possibilities of Advanced Customisation Using the Wordpress API**

Zásady pro vypracování:

1. **Prostudujte možnosti redakčního systému Wordpress a jeho další možnosti rozšíření pomocí nástroje Wordpress API, který popište v teoretické části.**
2. **Nastudujte také technologii REST API a možnosti využití této technologie v rámci CMS Wordpress pro tvorbu pluginů zobrazujících data ze vzdálených systémů za použití přenosového formátu JSON.**
3. **Navrhněte plugin, který konzumuje JSON data poskytnutá webovou službou a následně je zpracujte přímo pomocí Wordpress API v rámci redakčního systému Wordpress.**
4. **Naprogramujte navržený plugin tak, aby jeho funkcionalita byla využitelná i na mobilních zařízeních s možností pozdějšího rozšírení pluginu.**
5. **Plugin otestujte v testovací webové aplikaci.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **REST API Handbook. WordPress Developer Resources [online]. Wordpress [cit. 2017-11-28]. Dostupné z: https://developer.wordpress.org/rest-api/**
2. **MACDONALD, Matthew. WordPress: the missing manual. Second edition. Tokyo : O'Reilly, 2014. Missing manual. ISBN 9781449341909.**
3. **SABIN-WILSON, Lisa. Wordpress for dummies. 8th edition. Indianapolis, IN: John Wiley & Sons, 2017. ISBN 978-1119325925.**
4. **BIN UZAYR, Sufyan. Learning WordPress REST API. Packt Publishing, 2016. ISBN 9781786469243.**
5. **WILLIAMS, Brad. Professional WordPress Plugin Development. Indianapolis, IN: Wiley Publishing, 2011. ISBN 978-0470916223.**
6. **Wordpress Web Application Development – Third Edition: Building robust web apps easily and efficiently. 3rd Revised edition. Packt Publishing, 2017. ISBN 978-1787126800.**

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

Datum zadání bakalářské práce: **15. prosince 2017**

Termín odevzdání bakalářské práce: **25. května 2018**

Ve Zlíně dne 15. prosince 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

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

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské 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 bakalářská 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 bakalářskou 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/bakalářskou 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 bakalářské 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24.5.2018

.....
podpis diplomanta

ABSTRAKT

Cílem bakalářské práce je popsat a na základě praktických scénářů demonstrovat pokročilé přístupy customizace tohoto CMS s využitím WordPress API. Praktickým výstupem bude reálný modul, načítající do Wordpressu data pomocí webových služeb se standardním přenosovým formátem JSON. K tomu bude využita sekce WordPress API, která se nazývá REST API. Modul bude vypisovat importované data pomocí testovací responzivní webové aplikace.

Klíčová slova: WordPress, WordPress API, WordPress REST API, plugin, šablona, JSON

ABSTRACT

The aim of the bachelor thesis is to describe and, based on practical scenarios, demonstrate advanced approaches to adapt this CMS using the WordPress API. A practical output will be a real module that loads WordPress data through Web services with JSON standard transmission format. This will use the WordPress API section called REST API. The module will print the imported data using the Test Response Web Interface.

Keywords: WordPress, WordPress API, WordPress REST API, plugin, template, JSON

Zde bych rád poděkoval vedoucímu práce Ing. Radku Valovi, Ph.D. za pomoc při vedení bakalářské práce, připomínky, rady a trpělivost.

OBSAH

ÚVOD

V dnešní době, kdy většina společností vlastní webovou aplikaci a zároveň jim jde o úsporu čau, tak jsem se rozhodl vytvořit plugin nebo chcete-li modul, který se bude zabývat automatizovaným kopírováním dat z webové služby na webovou prezentaci. Konkrétně jsem si vybral import studijních oborů na všech fakultách Univerzity Tomáše Bati ve Zlíně.

Tyto studijní obory budeme importovat z webové služby informačního portálu UTB, kde je můžeme získat v datovém formátu typu JSON, který potřebuju. Koncová webová aplikace, na kterou budou data importovány bude fungovat na redakčním systému WordPress. Díky tomu, že to bude na redakčním systému WordPress, tak využijeme k importu WordPress REST API.

Výstupem bakalářské práce bude funkční plugin, který bude kopírovat potřebné data a následně je vypisovat pomocí testovací responzivní webové aplikace.

I. TEORETICKÁ ČÁST

1 WORDPRESS

WordPress je nejoblíbenější systém správy obsahu webu na světě. Více než 30 % webových stránek běží na této platformě, která se změnila z jednoduchého blogovacího systému na současnou podobu. Platforma mění způsob, jak webové stránky z celého světa pracují, nabízí více možností s větším výkonem, než tradiční webové stránky nebo většina ostatních systémů CMS. Vzhledem k tomu, že využívá WYSIWYG prostředí, tak má schopnost jednoduše měnit části webu a tím pádem ho využívá stále více lidí, jako jsou blogeři, webmasteři a internetoví obchodníci. [1] [2]

1.1 Informace o WordPressu

WordPress je systém pro správu obsahu nebo-li CMS, který pomáhá blogerům a správcům webu pravidelně upravovat obsah, aniž by bylo nutné používat tradiční HTML editor.

WordPress je sám o sobě softwarem s otevřeným zdrojovým kódem licencovaným podle GPLv2, což znamená, že je zdarma k použití a lze volně upravovat. Balíček je dostupný na stránkách WordPress.org. Tvůrce WordPressu Matt Mullenweg provozuje, pod svou firmou Automattic, doménu WordPress.com, kde nabízí WordPress na jejich serverech a nabízí i několik doplňků, například můžete použít svůj vlastní název domény, ale tyto možnosti jsou již placené. Existuje několik dalších možností, jak získat bezplatnou hostitelskou službu, avšak nejsou moc oblíbené.

Za to na WordPress.org jde stáhnout přímo balíček s WP, který můžeme spustit na svých vlastních serverech nebo zakoupit hostingový balíček od jedné z mnoha webhostingových společností, např. Wedos nebo Forpsi. Zde nám ale vyskočí i nějaké poplatky za provoz hostingu a domény, které jsou v řádu stokorun za rok.

Výhodou WP, který je hostován na vlastním hostingu, je přístup webmastera ke kódu na svých stránkách, stejně jako zhruba 32 000 bezplatných pluginů a 2 500 témat, která jsou uvedena na oficiálním úložišti (WordPress.org) - a ještě více z komerčních zdrojů.

Všechny tyto pluginy a témata jsou licencovány GPL, stejně jako WP samotný, což znamená, že majitelé stránek je mohou zdarma využívat. [1]

1.2 Historie

Historie WordPressu začíná jako většina ostatních. Na začátku byl obyčejný člověk s myšlenkou, kterou chtěl vyřešit problém, který aktuálně v té době měl. Jednalo se o studenta

vysoké školy Matta Mullenwega, který si v roce 2002 nainstaloval pro svou osobní potřebu blogovací systém b2/cafelog. Původní tvůrce b2/cafelog se bohužel musel vzdát jeho rozvoje kvůli osobním záležitostem a projekt a jeho komunita zůstaly bez vůdce. [1]

Dne 1. dubna 2003 Matt vytvořil novou větev b2/cafelog na SourceForge tím, že použil původní b2/cafelog systém a s pomocí Mike Littleho vytvořil vlastní verzi. Mattova kamarádka, Christine Tremouletová, doporučila název WordPress. Tento název se používá do teď.

Po stovkách nahraných verzích na úložiště SVN byla 27. května 2003 vydána první verze aplikace WordPress 0.7. Verze WordPress 1.0 byla vydána v lednu 2004, známá jako Davis. Mullenweg má sympatie k jazzovým velikánům. Všechny aktualizace pojmenovává po jazzových velikánech z minulosti a současnosti. Kromě toho Matt v každém vydání zahrnoval plugin nazvaný Hello Dolly. Tímto pluginem vzdal hold Louisovi Armstrongovi. [1]

Tato platforma byla navíc první platformou, která připomínala WordPress, který tu máme dnes. Příspěvky rozdělené do kategorií, pěkné URL pro vyhledávače, jednoduchá instalace s možností upgradu a moderováním komentářů byl software opravdovým blogovým systémem (a možná i CMS, avšak pokročilejší funkce byly pouze pro ty, kteří mají zkušenosti s programováním). [1]

V říjnu 2004 společnost CNET najala Mullenwega, aby jim pomohl s jejich blogy. Dále v říjnu založil společnost Automattic, ale to až po odchodu z CNET. Od investorů obdržel 1,1 milionů dolarů a to značilo rostoucí důvěru ve WordPress. V prosinci 2004 vyšla aktualizace 2.0 s názvem Duke. Jednalo se o zásadní úpravu administrace spolu se zvýšenou rychlostí a efektivitou přidávání příspěvků a nahráváním obrázků. [1]

V srpnu roku 2006 se v San Francisku konal první WordCamp, který dobrovolníci organizují pro uživatele a vývojáře, ti diskutují o všech věcech ohledně WP. Od roku 2006 máme registrováno 849 WordCamps v 70 městech a 65 zemích. Každoroční setkání v San Franciscu je hlavní událostí, kde Matt Mullenweg předkládá každoroční aktualizaci, které se říká State of the Word. [1] [3]

Roku 2008 byl spuštěn adresář šablon pro WP. Kdokoliv může vyvíjet a nahrávat své šablony a zpřístupnit je bezplatně desítkám tisíc návštěvníků, jediným kritériem je, že nejprve musí projít kontrolou kvality. Momentálně je v adresáři více než 2500 šablon zdarma ke stažení. [1]

Pokud bych měl vyjmenovat jednu konkrétní aktualizaci, která byla průkopnická, určitě by to byla verze 3.0 Thelonious, vydaná v červnu roku 2010. Zahrnutí funkce pro vlastní typy příspěvků otevřelo brány všem možnostem přizpůsobení typy obsahu, které můžeme spravovat pomocí aplikace WordPress. Dříve existovaly pouze příspěvky, stránky a taxonomie. To je přesně to, co odděluje WordPress CMS od jednoduchého blogového systému. Prostřednictvím nového WordPress API bylo mnohem jednodušší rozšířit funkce základní instalace aplikace a přizpůsobit je přesným potřebám. [1]

Mezi další funkce v nových verzích bych zařadil responzivitu administračního prostředí, automatické aktualizace samotného WordPressu, přidání vytvoření konceptu a plánovaného zobrazení příspěvku v budoucnu, dále třeba vylepšení editačního nástroje nebo nástroje přizpůsobení. [1]

1.3 Struktura

Balík WordPressu obsahuje mnoho různých souborů a adresářů. Obvykle se tyto soubory neupravují, ale pro nás je důležité mít o těchto souborech znalosti a vědět co všechno vykonávají. Níže se podíváme na některé z důležitých adresářů a souborů:

- wp-admin – Zde se nachází všechny administrační nástroje. Hlavní soubor tohoto adresáře je admin.php, který umožňuje připojení k databázi, zobrazení nástěnky a mnoho dalších funkcí, mezi které třeba patří ověření uživatele, jestli je např. administrátor.
- wp-content – Tento adresář je také hodně důležitý, protože obsahuje složky s pluginama a šablonama. Pluginy rozšiřují funkci WordPressu a šablony podkytují grafické rozhraní webových stránek. Dále bych ještě zmínil adresář uploads, kde se nahrávají všechny mediální soubory.
- wp-includes – Poslední složka v kořenu instalace umožňuje spuštění webu. Je zde většina klíčových souborů, obsahuje více než 140 souborů v hlavním adresáři a další podsložky obsahující například certifikáty, písma, JavaScript a widgety.
- wp-config.php – Soubor wp-config.php je konfigurační soubor, který upravujeme před provedením instalace. Pokud provedete skriptovanou instalaci, instalační skripty tento soubor upraví.
- index.php – Tato stránka je vstupní stránkou celého webu. Pokaždé, když někdo požádá o stránku z vašeho webu, zavolá se soubor index.php. Tento soubor pak zahájí proces načítání a zobrazí požadovanou stránku. [4] [5]

2 WORDPRESS API

WordPress API je zkratka rozhraní pro programování aplikací na WP. Lze ji rozdělit do několika sekcí API. Každá zahrnuje funkce obsažené v dané sekci, které můžeme používat. Společně vytvářejí to, co se dá nazvat WordPress API. Můžeme je využívat při tvorbě šablony, pluginu, add-onu. Níže vyjmenuju pár zajímavých sekcí WordPress API a též mezi ně patří právě i REST API, které nás zajímá nejvíc. [6]

- Plugin API – Poskytuje sadu tzv. "hooků", které umožňují přístup ke konkrétním částem aplikace WP. WordPress obsahuje dva různé typy hooků: Akce a filtry. Akční hook umožňuje spouštět vlastní kód v určitých bodech během spuštění. Můžeme například spustit vlastní funkci, která se spustí poté, co uživatel zaregistruje uživatelský účet v aplikaci WP. Filtr hook modifikuje text před přidáním nebo po načtení z databáze.
- Widgets API – Vytváří a spravuje widgety. Jsou k dispozici pro přidání do jakéhokoli registrovaného postranního panelu v šabloně. Rozhraní API umožňuje používat více instancí stejného widgetu ve všech postranních lištách.
- Shortcode API – Shortcode je jednoduchý hook, který vám umožňuje volat funkci PHP tím, že přidáte něco jako [shortcode] na příspěvek nebo stránku.
- HTTP API – Odesílá požadavky HTTP. Toto rozhraní načte obsah z externí adresy URL nebo odešle obsah na adresu URL. V současné době máte pět různých způsobů, jak odeslat požadavek HTTP. Toto API rozhraní standardizuje tento proces a testuje všechny metody před provedením. Na základě konfigurace serveru API použije příslušnou metodu a provede požadavek.
- Settings API – Primární výhodou použití rozhraní API pro nastavení je zabezpečení. Všechna data nastavení jsou očištěna, takže při ukládání nastavení nemusíte mít obavy ze útoků typu forgery (CSRF) a cross site scripting (XSS).
- Options API – Toto rozhraní API nabízí možnost vytvářet nové možnosti, aktualizovat stávající možnosti, odstraňovat a načítat všechny již definované možnosti.
- Dashboard Widgets API – Vytvoří widgety řídicího panelu administrátora. Widgety se automaticky zobrazují na panelu nástrojů WordPress a obsahují všechny standardní funkce přizpůsobení včetně minimalizace.
- Database API – Přistupuje k databázi WP. To zahrnuje vytváření, aktualizaci, mazání a načítání záznamů databáze. [7]

2.1 REST API

V této kapitole se budeme zabývat podsekcí Wordpress API, která má název REST API. Zjistíme základní informace o API, jak REST API funguje a jeho následné využití.

2.1.1 Základní informace

Aplikace WordPress REST API je pro WP revoluční z několika důvodů. Jedním z důvodů je především to, že je schopen zobrazovat a ukládat obsah z jiných webů a aplikací bez ohledu na to, zda používají WP. To znamená, že WP lze použít jako systém správy obsahu s jakýmkoli rámcem nebo libovolným programovacím jazykem.

Aplikace REST API usnadňuje vývojářům vytvářet nové typy aplikací pomocí aplikace WP. To znamená, že můžeme vytvořit spoustu nových generických editorů obsahu, specializovaných editorů a nástrojů pro správu stránek. Vytvoření těchto nástrojů již nevyžaduje vlastní rozhraní API.

Šablony mohou používat REST API pro dynamické načítání obsahu. WordPress REST API může také poskytnout efektivnější a standardizovanější způsob zpracování formulářů nebo zpracování front-end AJAX, na rozdíl od přepracování stávajícího admin-ajax. Front-end admin-ajax se využívá například v nekonečném načítání příspěvků.

Historicky je tento typ integrace omezen a je dostupný pouze prostřednictvím XML-RPC. Nové rozhraní WordPress REST API však používá rozhraní RESTful API – univerzální konektor pro data na internetu. To, co bylo zbytečně komplikované s XML-RPC, je zjednodušeno použitím běžnějšího standardu REST.

Zatímco API XML-RPC používají XML pro vytváření požadavků a odpovědí, RESTful API používají standardní notaci JavaScript nebo JSON.

Většina programovacích jazyků může snadno převést své standardní datové struktury na JSON a naopak. Například v PHP používáme json_encode() a json_decode() k překladu PHP arrayů nebo objektů do JSON.

To znamená, že na praktické úrovni může být WP nástrojem pro správu obsahu pro aplikaci napsanou v jakémkoli jazyce a může snadno odesílat a přijímat data pomocí standardu JSON.

Aplikace WordPress JSON REST API poskytuje nový způsob interakce s příspěvky, uživateli, komentáři a taxonomiemi v aplikaci WP pomocí populárního standardu API REST-

ful, který umožňuje vytvářet, číst, aktualizovat a mazat hlavní objekty aplikace WP a poskytuje infrastrukturu pro vytváření vlastního uživatelské rozhraní API.

Aplikační program WordPress REST API lze použít na jednostránkových aplikací na straně klienta – buď jako součást WP šablony nebo pluginu, nebo v samostatném front-end klientovi, který je zcela oddělen od backendu. Podobně také otevírá nové příležitosti pro dynamičtější rozšíření pluginů a témat, stejně jako pro integraci se samostatnými firmami založenými na SaaS.

Poslední co bych dodal je, že nám poskytne větší flexibilitu při vývoji vlastních pluginů nebo šablon a jejich integraci s dalšími službami. [8]

2.1.2 Jak REST API funguje?

Abychom pochopili, jak funguje WordPress REST API, musíme pochopit, jak fungují požadavky a reakce HTTP. Když zadáte adresu URL do adresního řádku prohlížeče, je to požadavek. Když server zobrazí webovou stránku nebo aplikaci pro danou adresu URL, je to odpověď.

Zde jsou čtyři hlavní typy metod HTTP, které web používá:

- **GET** – slouží k načtení dat ze serveru
- **POST** – slouží k odesílání dat na server
- **PUT** – slouží k změně nebo aktualizaci dat na serveru
- **DELETE** – slouží k odebrání dat ze serveru

S ohledem na tyto jednoduché definice je zadání adresy URL v prohlížeči požadavkem GET. Zadání přihlašovacích údajů pro webové stránky je žádost o POST. Změna vašeho aktuálního hesla na nový je žádost PUT, při odstranění vašeho účtu je požadavek DELETE.

Další termíny jsou "trasy" a "koncové body". Trasa je typicky adresa URL nebo část adresy URL, ke které se pokoušíme přistupovat, zatímco koncový bod je obvykle odpověď, kterou obdržíme od serveru.

Když externí zdroje odesílají požadavky HTTP serveru, kde hostujeme WP, REST API rozšiřuje vaše data bezpečně tak, že odpovídá na tyto požadavky se společnou architekturou a vlastní sadou protokolů.

To umožňuje, aby obsah WP, jako jsou příspěvky, stránky a komentáře, byly zpracovávány jako surová data. Celkovým cílem je umožnit změnu obsahu vašeho webu bez nutnosti přístupu do administrátorské oblasti aplikace WordPress. Takto můžete provádět změny na vašem webu pomocí nástroje JSON, jehož reakce dávají vývojářům řadu různých způsobů interakce s jejich stránkami. [9]

2.1.3 Využití REST API

Předtím než se dostaneme k využití REST API, tak si musíme říct několik klíčových pojmů, které jsou spojené s rozhraním API.

Zde jsou klíčové pojmy, které tvoří toto rozhraní API:

- Trasy / koncové body – Technická definice trasy je adresa URL, kterou lze mapovat pomocí různých metod HTTP. Mapování mezi trasou a metodou HTTP se nazývá "koncový bod". Také můžeme získat přístup k WordPress REST API a zjistit, které trasy a koncové body jsou k dispozici pro vaše stránky, přidáním cesty "/wp-json/" do konce vašeho URL.
- Žádosti – Aplikace WordPress REST API zpracovává požadavky s třídou s názvem WP_REST_Request. Je to primární třída v infrastruktuře REST API. Používá se k ukládání a načítání informací pro všechny vaše požadavky. Odesílání žádostí můžete odesílat vzdáleně pomocí metod HTTP nebo je můžete provádět interně prostřednictvím PHP.
- Odpovědi – Jsou zpracovány pomocí třídy WP_REST_Response. Odpovědí jsou údaje, které obdržíte ze žádosti, jak bylo uvedeno výše. API používá tuto třídu k vrácení dat odeslaných z koncových bodů. Může vrátit také chyby.
- Schéma – Je to koncept uvnitř WordPress REST API, který slouží různým účelům. Schéma API definuje koncové body datových struktur, které mohou používat a obsahuje seznam vlastností, které může vrátit REST API. Obsahuje také parametry, které rozhraní API může přijmout a zajišťuje jeho zabezpečení validací požadavků, které API přijímá.
- Třídy kontrolerů – Jak můžeme vidět, WordPress REST API má spoustu částí, které všechny musí spolupracovat. Řídicí třídy přinášejí všechny tyto prvky dohromady na jednom místě. Pomocí třídy kontrolerů můžete spravovat registraci tras a koncových bodů, zpracovávat požadavky, využívat schémata a generovat API odpovědi [9].

Mezi takové jednoduché využití REST API bych zařadil vypsaní obsahu článku. Využití REST API je nespočet, zařadil bych mezi ně aspoň pár zajímavých:

- Mám web, který není na WordPressu a chci na něm zobrazovat nejnovější příspěvky na WordPress blogu.
- Aktualizace části obsahu několika micrositů vždy dle hlavního webu.
- Tady se budu opakovat co v předchozím kroku, ale nechci pouze, aby se obsah/příspěvky zobrazovaly, ale aby se přímo nahrály, pokud je na druhé straně také WP.
- Chci na klientském portálu zobrazovat novinky z dané služby.
- Mobilní aplikaci, která je spravována přes WP. [10]

3 ŠABLONY A PLUGINY

Zde se stručně seznámíme s částmi WP, které jsou pro nás podstatné, jedná se o šablony a pluginy.

3.1 Šablony

Nejdůležitější částí šablony WordPressu je hierarchie. Šablona se skládá z mnoha typů souborů včetně konkrétních šablon daných stránek. Stránky šablony mají určitou strukturu a hierarchii. To znamená, že pokud jeden soubor šablony chybí, WordPress vyvolá soubor šablony další úrovně podle hierarchie a zobrazí ho. To umožňuje vývojářům vytvářet témata, která jsou detailní, různorodá a plně využívají všechny dostupné možnosti hierarchie šablony. Je také možné mít plně funkční šablonu WordPressu, která se skládá pouze ze souboru index.php a stylu. [12]

Obrázek č. 1 – Hierarchie šablony [11]

Ve většině stránek šablony v hierarchii (ne nutně všude) přidáváme kus kódu zvaný "smyčka". Smyčka je nezbytnou součástí šablony WordPressu. Zobrazuje příspěvky v chronologickém pořadí a umožňuje definovat vlastní vlastnosti zobrazení pomocí různých HTML značek. Podíváme-li se na "smyčku" stránky šablony, najdeme zde některé zajímavé části kódu zabalené v PHP značkách. Většina z nich jsou značky specifické pro Word-

Press a funkce šablony, které fungují pouze v rámci systému WordPress. Značky a funkce můžou mít různé parametry nastavení.

Ne všechny značky a funkce WordPressu se fungují uvnitř smyčky. Například v souboru header.php najdete několik značek, které fungují mimo smyčku. Konkrétně na stránce šablony header.php (stejně jako stránky šablony footer.php a sidebar.php) najdete také několik funkcí specifických pro WordPress, které jsou součástí Plugin API a Script API. [12]

3.2 Pluginy

Pluginy jsou balíčky kódu, které rozšiřují základní funkce aplikace WP. Pluginy WP se skládají z kódu PHP a dalších věcí, jako jsou obrázky, CSS a JavaScript.

Tím, že vytvoříme svůj vlastní plugin, rozšiřujeme WP, vytváříme další funkce nad to, co WP již nabízí. Můžete například napsat plugin, který zobrazuje odkazy na 10 nejnovějších příspěvků na vašem webu.

Většina pluginů WP se skládá z mnoha souborů, ale plugin opravdu potřebuje pouze jeden hlavní soubor se speciálně formátovaným DocBlock v záhlaví.

Hello Dolly je jeden z prvních pluginů, je dlouhý jen 82 řádků. Hello Dolly ukazuje texty ze slavné skladby v administraci WP. [13]

3.2.1 Hooky

Hooky jsou páteří aplikace WordPress. Umožňují vývojářům pluginů "hook" (navázání) do pracovního postupu aplikace WP, aby změnili, jak funguje, aniž by přímo upravoval kód jádra. To umožňuje uživatelům snadno přejít na novější verze WP bez ztráty modifikací.

Pokud vývojář změnil kód jádra, budou tyto úpravy při příštím aktualizaci WP ztraceny. Aktualizace by přepsala všechny tyto změny. Použití hooků umožňuje vyvíjet pluginy v samostatných složkách kromě jádra, takže kód pluginu je oproštěný od aktualizací.

Bez hooků by pluginy neměly žádný způsob modifikovat to, jak WP funguje. Jakmile se naučíme používat hooky, pochopíme přesně, proč je WP tak silnou platformou a má tisíce pluginů postavených pro miliony uživatelů.

WordPress má dva hlavní typy hooků: akční hook a filtr hook. Prvotní funkce umožňuje vykonávat funkci v určitém bodě a druhý umožňuje manipulaci s výstupem procházejícím hookem. [7]

4 POUŽITÉ TECHNOLOGIE

Zde popíšeme základní technologie, které WordPress používá a které budeme používat při tvorbě pluginu. Jedná se o HTML, CSS a JavaScript.

4.1 HTML, CSS, JavaScript

HTML (HyperText Markup Language), v překladu něco jako hypertextový tagový jazyk. Také se mu říká značkovací jazyk pro hypertext. HTML vznikl z původního univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Jeho vývoj je úzce propojen s vývojem web prohlížečů. HTML v podstatě programovacím jazykem, u nějž se využívá k programování tagů (znaků) namísto příkazů. Tudíž není nutno jej překládat do strojového kódu. Takový HTML dokument je soubor složený z HTML tagů (znaků), a lze jej vytvořit v běžném textovém editoru a v také textovém prohlížeči číst. Samotný jazyk HTML se vyznačuje množstvím tagů (značek) a jejich vlastností (atributů). Mezi tyto tagy (značky) se vkládají části textu a tím se určuje obsah a znění HTML dokumentu. Následně se jednotlivé značky s jejich vlastnostmi uzavírají mezi úhlové závorky <a>. Každá jednotlivá část dokumentu je tvořená tzv. otevírací značkou, nějakým textovým obsahem a vhodnou ukončovací značkou. Značky na začátku a na konci tohoto fragmentu jsou většinou shodné, jen před koncovou je vložen znak lomítka, kterým se tato část HTML dokumentu uzavře. A několik takových fragmentů (částí, elementů, prvků) tvoří celý dokument HTML. [14]

Se vznikem standardu CSS3 nyní CSS nabízí dynamickou interaktivitu, kterou dříve podporoval pouze JavaScript. Například můžeme stylovat libovolný HTML element, u kterého se změní jeho rozměry, barvy, rámečky, mezery a tak dále, ale za to nyní můžete také přidávat animované přechody a transformace pomocí několika řádků CSS. [15]

Technologie JavaScript byla vytvořena pro umožnění skriptování přístupu ke všem prvkům dokumentu HTML. Poskytuje prostředky pro dynamickou interakci s uživatelem, jako je například kontrola platnosti e-mailové adresy ve vstupních formulářích nebo zobrazování pokynů. V kombinaci s CSS je JavaScript vytváří dynamické webové stránky, které se mění rychleji, než když se musí načíst celá stránka ze serveru. [15]

II. PRAKTICKÁ ČÁST

5 PLUGIN LOAD STUDY COURSES

V praktické části navrhneme plugin, který jsem nazval Load Study Courses, pro webovou aplikaci na doméně www.utb.cz, která funguje na CMS WordPress. Budeme se zde zabývat navrhnutím pluginu, frontendovou implementací a následným finálním testováním.

5.1 Analýza požadavků

Hlavním důvodem, proč vytvořit tento plugin bylo to, aby se desítky studijních programů a oborů nemusely ručně kopírovat na nově vytvořenou webovou aplikaci Univerzity Tomáše Bati. Další výhodou pro tvorbu pluginu bylo to, že UTB má vytvořenou webovou aplikaci na webové služby nad IS/STAG, které získávají celou řadu dat pomocí návrhového vzoru REST. Zde jsem si našel všechny JSON soubory, které potřebuju na správné fungování pluginu, následného importu a výpisu.

Mezi hlavní požadavky bych zařadil:

- Import všech fakult a jejich následných studijních programů
- Rozřazení typu programu mezi bakalářské, magisterské a doktorské
- Studijní programy musí být též rozděleny na formu studia (prezenční a kombinované)
- Import všech platných oborů napříč fakultami, který musí správně zařadit typ programu, správnou fakultu, formu studia a jazyk oboru
- Možnost přiřadit klíčová slova (zájmy) k daným oborům
- Následný výběr oboru pomocí klíčových slov (zájmů), jednoduchého výběru podle fakulty a seznam všech oborů s možností filtrace

Za nefunkční požadavky uvedu tyto:

- Řešení pomocí pluginu
- Ukládání dat o studijním oboru pomocí tzv. custom post type
- Hierarchie, fakult, studijních programů, formy studia a jazyků
- Řešení pomocí WordPress REST API
- Filtrace oborů v rámci uživatelské části aplikace bude řešena pomocí technologie AJAX

5.2 Vývoj

Požadavky máme vydefinované, tak můžeme přejít k vývoji samotného pluginu. Nejprve musíme začít vytvořením struktury pluginu. K tomu jsem použil webovou aplikaci WordPress Plugin Boilerplate Generator, která mi základní strukturu pluginu vytvoří automaticky. [16]

Když už bychom základní strukturu měli, tak se můžeme pustit do samotné tvorby funkcí pluginu. Nejprve bych začal vytvořením custom post types, což v překladu znamená vlastní typ příspěvku. Je to vlastně jak normální příspěvek akorát s jinou hodnotou u post_type v databázi. Funkcí custom post types si vytvoříme typ příspěvků, které si pojmenujeme Study Courses, zde budou poté všechny studijní obory.

```
1  register_post_type( 'study_course',
2          array(
3              'labels' => [
4                  'name' => __( 'Study Courses', 'utb-textdomain' ),
5                  'all_items' => __( 'All Study Courses', 'utb-
   textdomain' )
6              ],
7              'menu_icon' => 'dashicons-list-view',
8              'public' => true,
9              'supports' => array('title', 'editor', 'thumb-
   nail','page-attributes','post_formats'),
10             'rewrite' => array('slug' =>
   '/univerzita/studium/nabizene-obory/seznam-oboru'),
11         )
12     );
```

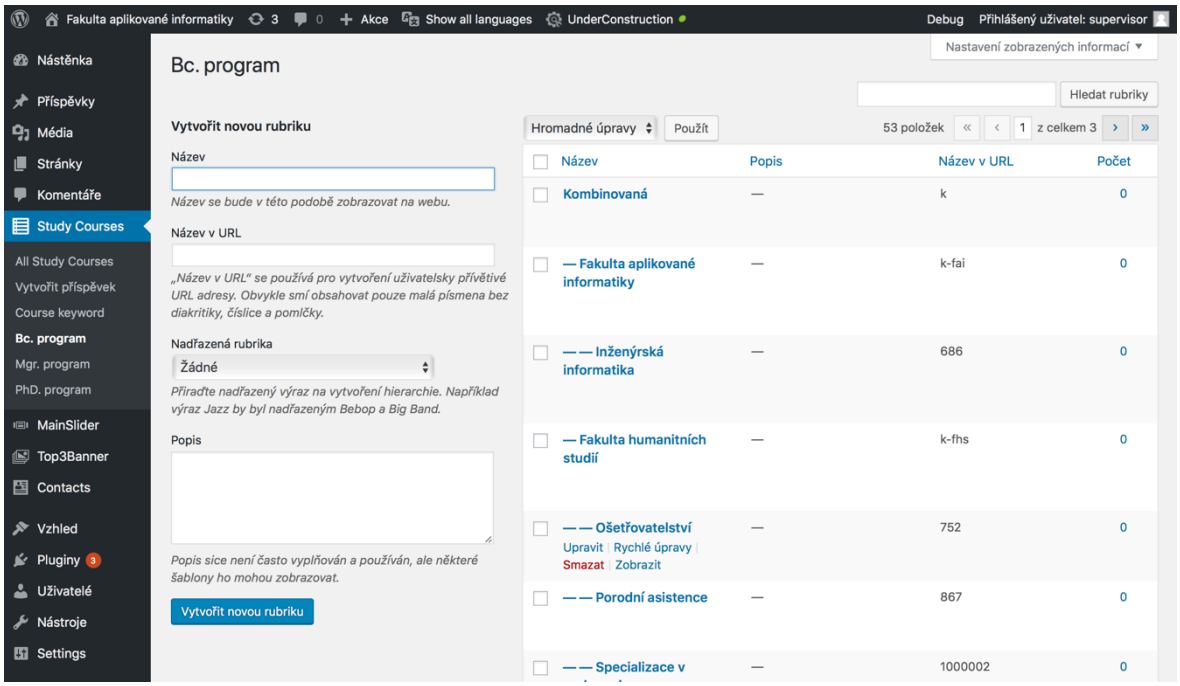
U výše zmíněného custom post types registrujeme post type study_course, který má ruzné možnosti nastavení co všechno daný příspěvek může obsahovat (název, HTML editor, náhledový obrázek) nebo taky například zobrazení vlastní ikony v menu.

Dále si musíme vytvořit custom taxonomies, zjednodušeně řečeno to jsou kategorie, které můžou být zaškrtávány v příspěvku daného custom post types, v našem případě to bude Bc. program, Mgr. program, PhD. program, Course language a Course keyword. Jsou zde zase možnosti nastavení, nás nejvíce zajímá jestli mají být kategorie hierarchické nebo plošné. Hierarchicky budou nastaveny všechny programy. Klíčové slova a jazyky budou plošně, protože tam není potřeba hierarchie. Níže jsem vybral kus kódu, který registruje taxonomii Bc. programu.

```
register_taxonomy(
        'bakalar',
        'study_course',
        array(
            'label' => __( 'Bc. program' ),
            'rewrite' => array( 'slug' => 'bakalar' ),
            'hierarchical' => true
        )
    );
```

Obě funkce jsou vytvořeny v souboru load-study-courses.php, který se nachází v kořenu pluginu. Teď máme vytvořené všechny podklady k tomu, abychom se mohli pustit do vývoje importu potřebných dat.

Nejprve musíme importovat fakulty, její programy a typ studia, které musí být ještě rozřazené do typu programu, což je bakalářské, magisterské a doktorandské. Zde už budeme využívat REST API. U programů bude řazení hierarchické a to tím způsobem, že máme vytvořené 3 taxonomie forem studia, do kterých budeme postupně řadit studijní programy. Nejprve bude rozdělení podle typu studia, tudíž prezenční nebo kombinované, pak budou zařazeny fakulty a nakonec do fakult studijní programy.



Obrázek č. 2 – Rozřazení studijních programů v bakalářské formě studia

Výše vidíme už naimportovány a rozřazeny studijní programy, níže ještě stručně popíšu kód importu a rozřazení bakalářské formy, pro formu magisterskou a doktorandskou je kód obdobný, akorát se liší v podmínkách řazení.

```
1  foreach ($formaArray as $forma){
2      if(($forma['zkratka'] == 'P') || ($forma['zkratka'] == 'K')){
3          $this->formaStudia = 'bakalar';
4          $parent = null;
5          $this->insertTermToTaxonomy($forma['nazev'], '', $for-
   ma['zkratka'], $parent);
6          foreach ($pracovisteArray as $pracoviste){
7              $zkratka = $forma['zkratka'].'-'.$pracoviste['key'];
8              $this->insertTermToTaxonomy($pracoviste['nazev'], '',
   $zkratka, $forma['zkratka']);
9                  foreach ($programyArray as $program){
10                     $zkratka1 = $forma['zkratka'].'-
   '.$program['fakulta'];
11                     if($program['typ'] == 'Bakalářský'){
12                         if($zkratka == $zkratka1){
13                             if($forma['nazev'] == $pro-
   gram['forma']){
14                                 $this-
   >insertTermToTaxonomy($program['nazev'], '', $program['stprIdno'],
   $zkratka);
15                                 continue;
16                             }
17                         }
18                     }
19                 }
20         }
21     }
22 }
```

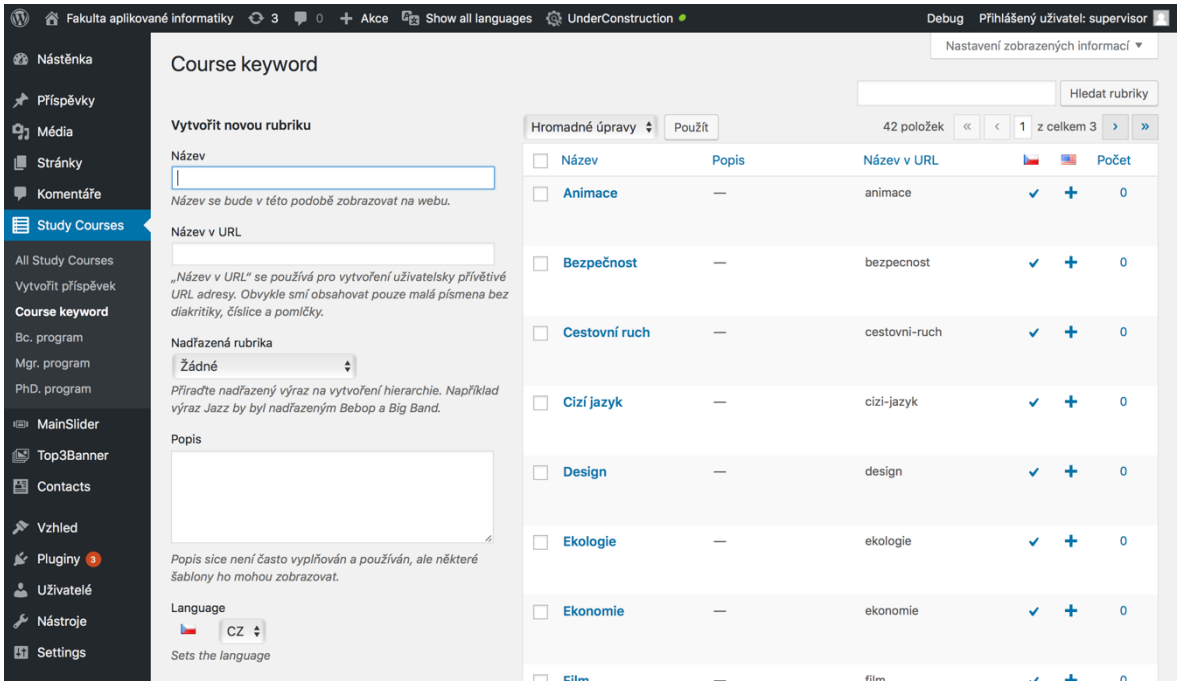
V prvním cyklu projíždíme dekódovaný JSON formát, který obsahuje formu studia, další podmínkou vyloučíme všechny další formy studia, zůstanou nám jenom prezenční a kombinované. Dále máme funkci insertTermToTaxonomy( ), která následně vkládá danou hodnotu do taxonomie. V dalším cyklu budeme procházet fakulty, které už budeme hierarchicky řadit pod formu studia pomocí její zkratky. V posledním cyklu máme už dané studijní programy fakulty, kde v první podmínce ověříme, zda je bakalářský, v druhé jestli

sedí fakulta a poslední podmínce se ověřuje, jestli je program prezenční nebo kombinovaný. Následně se opět danou funkcí vloží program do hierarchie taxonomie.

V současné třídě jsem ještě nechal načítat klíčová slova a jazyky. Klíčová slova jsem importoval už z přímo předdefinovaného pole prvků.

```
1 foreach ($keywordArray as &$keyword){
2     $this->formaStudia = 'study_course_keyword';
3     $this->insertTermToTaxonomy($keyword, '', '', null);
4 }
```

Jazyk se poté importoval pomocí JSON formátu z webové služby UTB, vybrali jsme omezení jen na češtinu a angličtinu.



Obrázek č. 3 – Taxonomie study_course_keyword

Na obrázku č. 2 a č. 3 vidíme, jaký je rozdíl mezi taxonomií hierarchickou a plošnou.

Nyní se už dostáváme k nejdůležitější části kódu, která bude importovat již finální obory. Tyto obory budou následně při importu řazeny. První řazení je do příslušného typu programu, dále do formy studia, správné fakulty a správného jazyku. Zde budeme ještě kontrolovat, jestli obor bude platný v následujícím roku, když ne, tak bude nastaven jako koncept.

Zde načteme první JSON, který obsahuje všechny platné studijní programy na celé univerzitě.

```
1 $str1 = file_get_contents('https://stag-
  ws.utb.cz/ws/services/rest/programy/getStudijniProgramy?pouzePlatne=
  true&outputFormat=json');
2 $json1 = json_decode($str1, true);
3 $programyArray = $json1[0]['programInfo'];
```

Následně budeme cyklem procházet všechny programy. Zde využijeme ID programu, je důležité k tomu, abychom získali všechny obory daného programu.

```
1 $obory = $program['stprIdno'];
2 $str2 = file_get_contents('https://stag-ws.utb.cz/ws/servi
  ces/rest/programy/getOboryStudijnihoProgramu?outputFormat=json&stprI
  dno='.$obory);
3 $json2 = json_decode($str2, true);
4 $oboryArray = $json2[0]['oborInfo'];
```

Dále bude procházet již dané obory studia, rozřazovat je a zapisovat do custom post typu.

Mezi první podmínku patří kontrola, jestli je obor platný, je zde ošetřeno i to, jestli bude platný příští rok, jinak nebude publikován a bude uveden v systému WP jako koncept.

```
1 $postStatus = 'publish';
2 if(isset($obor['neplatnyOd'])){
3    $neplatnyOd = $obor['neplatnyOd'];
4    if(intval($neplatnyOd)<=intval(date('Y')+1)){
5         $postStatus = 'draft';
6    }
7 }
```

Zde máme podmínky rozřazování jazyků a oborů do daných typů studijních programů.

```
1 if($obor['typ'] == 'Bakalářský'){
2     $this->formaStudia = 'bakalar';
3 } elseif(($obor['typ'] == 'Magisterský')||($obor['typ'] == 'Navazu-
  jící')) {
4     $this->formaStudia = 'magistr';
5 } elseif($obor['typ'] == 'Doktorský' || $obor['typ'] == 'Rigorózní')
  {
6     $this->formaStudia = 'doktorand';
7 } else {
8     continue;
9 }
10 $parent = $obor['stprIdno'];
11 $language = $obor['jazyk'];
12     if($language == 'AN'){
13         $lang = 'en';
14     } else if ($language == 'CZ') {
```

```
15          $lang = 'cs';
16      }
```

To bychom z podmínek měli všechno. Dále se budeme zabývat tím, aby se při opakovaném importu dat obory nepřidávaly znovu, ale jen se aktualizovali podle ID. Když zadané ID nebude v custom post typu, tak se vytvoří nový příspěvek, v našem případě obor.

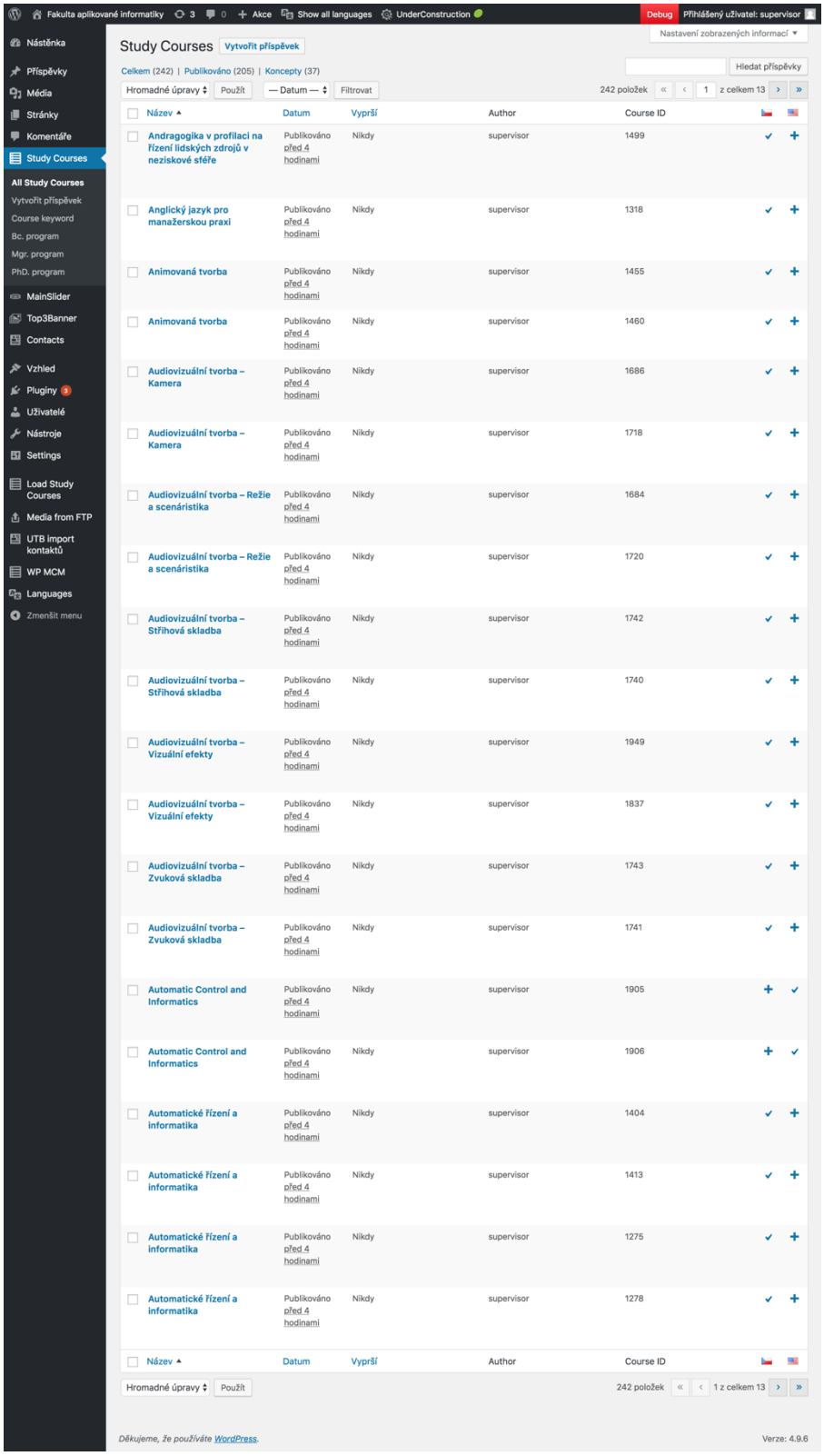
```
17 $post_id = null;
18 $status = '';
19 if ($query->have_posts()) {
20     // update existing custom post
21     $post_id = $query->post->ID;
22     $query->post->post_status = $postStatus;
23     $query->post->post_title = $postTitle;
24     wp_update_post($query->post);
25     foreach ($metaToStore as $metaKey=>$value) {
26         update_post_meta($post_id, $metaKey, $value);
27     }
28     // update existing language
29     pll_set_post_language($post_id, $lang);
30     $status = 'updated';
31 } else {
32     // insert new custom post
33     $new_post = array(
34         'post_title' => $postTitle,
35         'post_status' => $postStatus,
36         'post_date' => date('Y-m-d H:i:s'),
37         'post_author' => $user_ID,
38         'post_type' => 'study_course'
39     );
40     $post_id = wp_insert_post($new_post);
41     // update existing language
42     pll_set_post_language($post_id, $lang);
43     foreach ($metaToStore as $metaKey=>$value) {
44          add_post_meta($post_id, $metaKey, $value);
45     }
46     $status = 'imported';
47 }
```

Mezi poslední funkční požadavky, které nám zbývají, tak patří, ať se u daného oboru automaticky označí zařazení jazyku oboru a zařazení do studijního programu. U studijního programu, jak bylo výše zmíněno, bude řazení fungovat tak, že se vybere bakalářský, magisterský nebo doktorandský program, dále v JSONu máme k dispozici u každého oboru ID daného studijního programu, to využijeme při označení studijního programu u oboru. Tady se nám hodí i to, že máme u bakalářské, magisterské nebo doktorandské taxonomie hierarchické řazení. Díky tomu můžeme označit všechno rodičovské kategorie, takže se označí přímo i fakulta a forma studia.

```
48 //set language
49 $language_term = get_term_by('slug', $language,
   self::LANGUAGE_TAXONOMY); // wp Term Object
50 $language_term_id = $language_term->term_id; // get numeric term id
51 $language_set = array( intval($language_term_id) );
52 wp_set_post_terms( $post_id, $language_set,
   self::LANGUAGE_TAXONOMY);
53
54 //set program
55 $parent_term = $term = get_term_by('slug', $parent, $this-
   >formaStudia); // wp Term Object
56 if(is_object($parent_term)){
57     $parent_term_id = $parent_term->term_id; // get numeric term id
58     $tag = array( intval($parent_term_id) );
59     wp_set_post_terms( $post_id, $tag, $this->formaStudia);
60     //assign all parent terms
61     $terms = wp_get_post_terms($post_id, $this->formaStudia);
62     foreach($terms as $term){
63         while($term->parent != 0 && !has_term( $term->parent, $this-
   >formaStudia, $post_id )){
64             // move upward until we get to 0 level terms
65             wp_set_post_terms($post_id, array($term->parent), $this-
   >formaStudia, true);
66             $term = get_term($term->parent, $this->formaStudia);
67         }
68         continue;
69     }
70 }
```

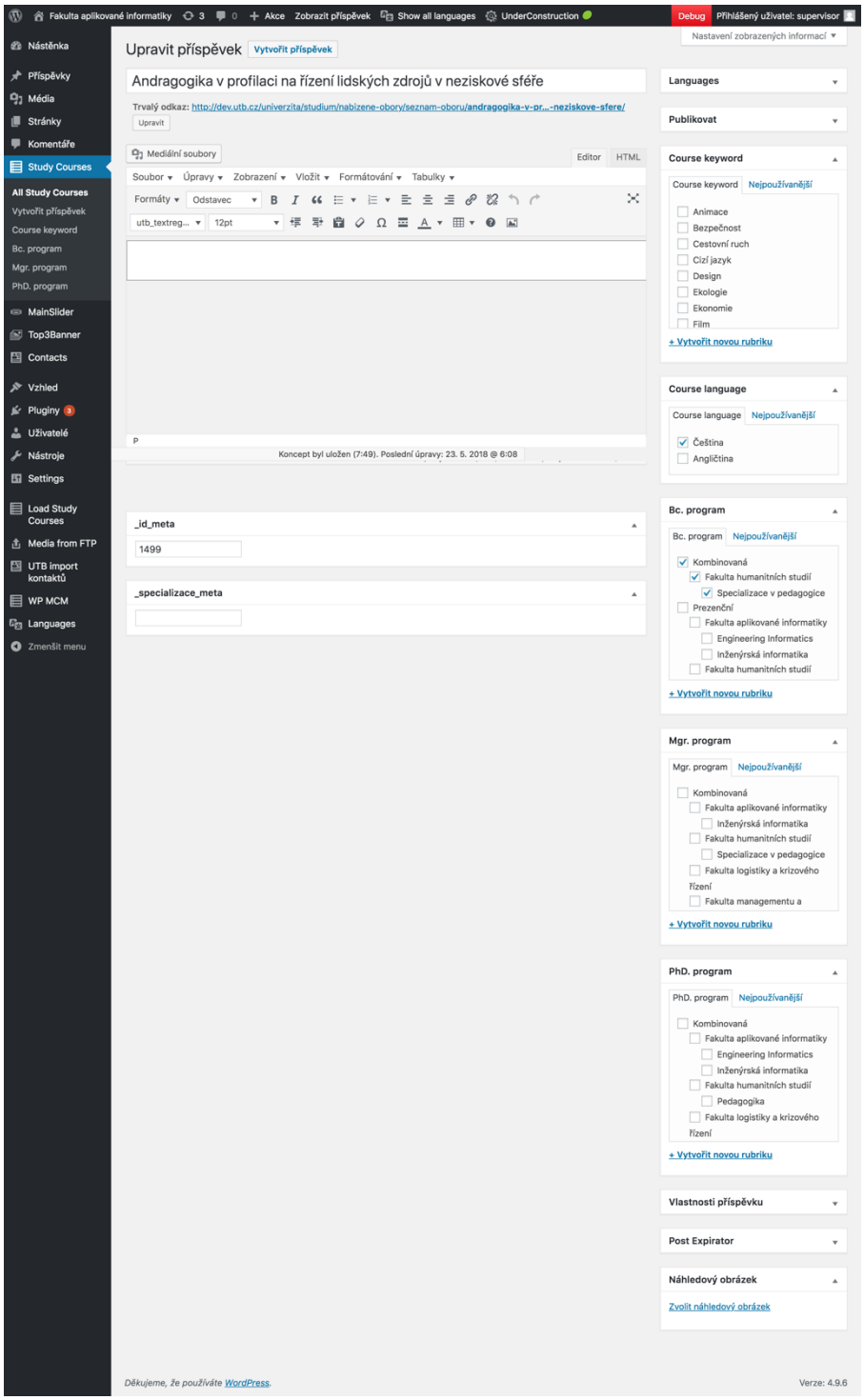
Na obrázku č. 4 můžeme vidět, jak jsou v administraci WordPressu naimportovány všechny obory, máme zde i nějaké koncepty, což znamená, jak jsem už zmiňoval, že obory už

nejsou platné. Vidíme, že některé obory tu máme duplicitně, to neznamená, že by byly špatně naimportovány, ale například to, že obor se vyskytuje jak v kombinované, tak i prezenční formě studia.



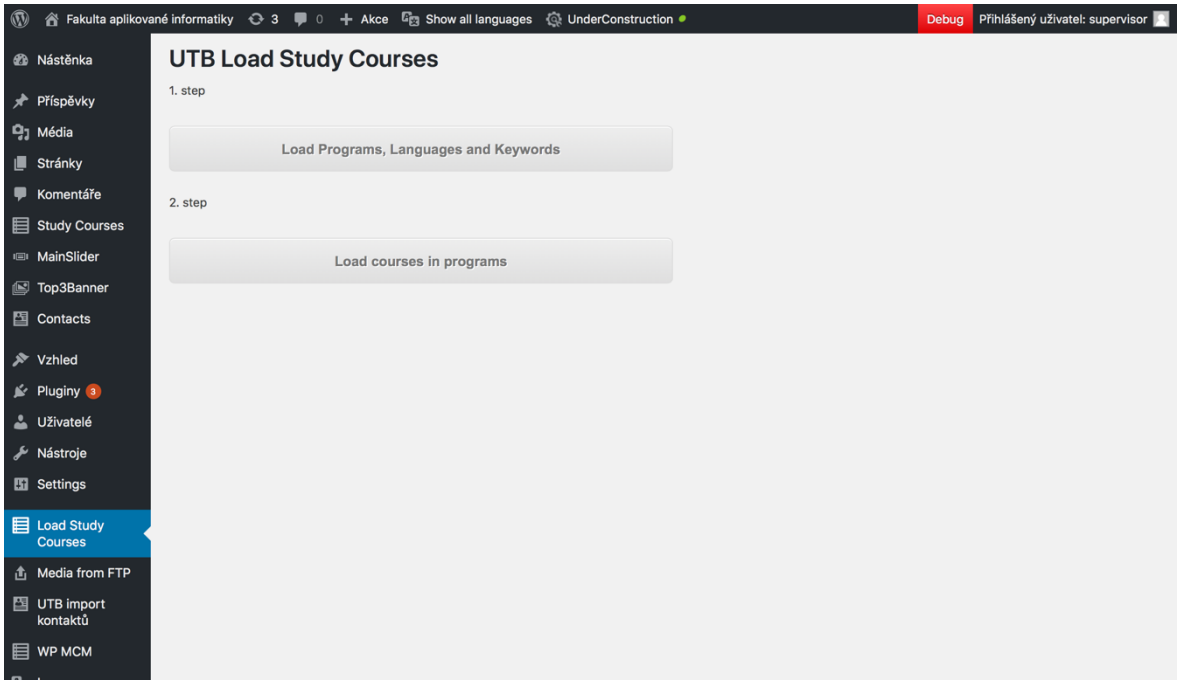
Obrázek č. 4 – Taxonomie study_course_keyword

Na obrázku č. 5 máme detail oboru, kde můžeme vidět, že byl automaticky vybrán jazyk studia, Bc. program a následné označení studijního programu hierarchicky až po nejvyšší stupeň. Zde se taky dostáváme ke klíčovým slovům, tady si je můžeme manuálně vybrat a poté se podle nich bude obor vyhledávat.



Obrázek č. 5 – Detail oboru

Tímto bychom měli splněné všechny požadavky ohledně importu dat. Tyto funkce se budou spouštět v menu administraci WP pod záložkou Load Study Courses. Jak na obrázku č. 6 vidíme, tak zde máme dva kroky. Důležité je dodržet postup importu, aby byla zachována správná funkcionalita, tudíž se musí první provést krok 1 a poté 2. V prvním kroku naimportujeme studijní programy, jazyky a klíčová slova. V kroku druhém se naimportují studijní obory.

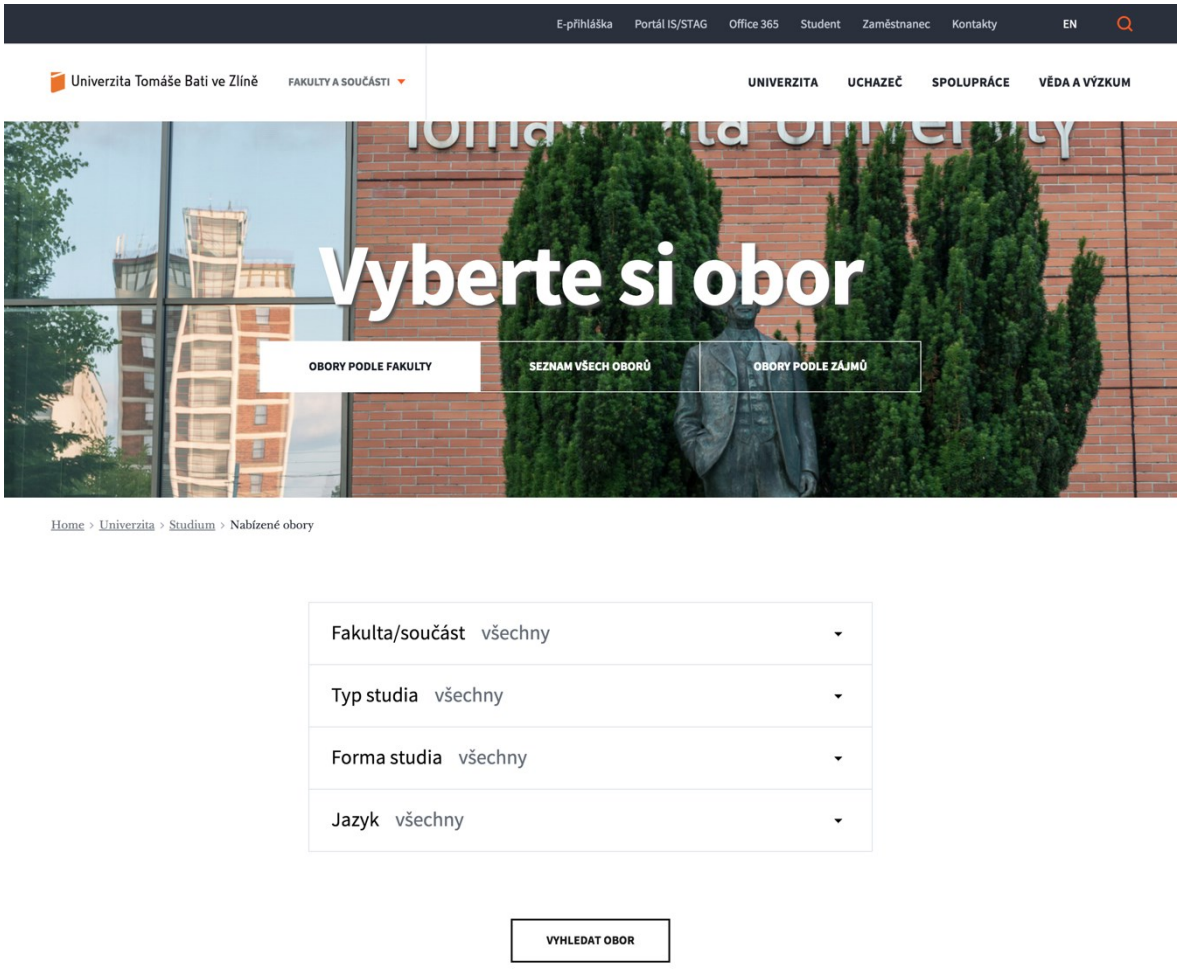


Obrázek č. 6 – Administrační stránka pluginu Load Study Courses

5.3 Implementace na frontendový návrh

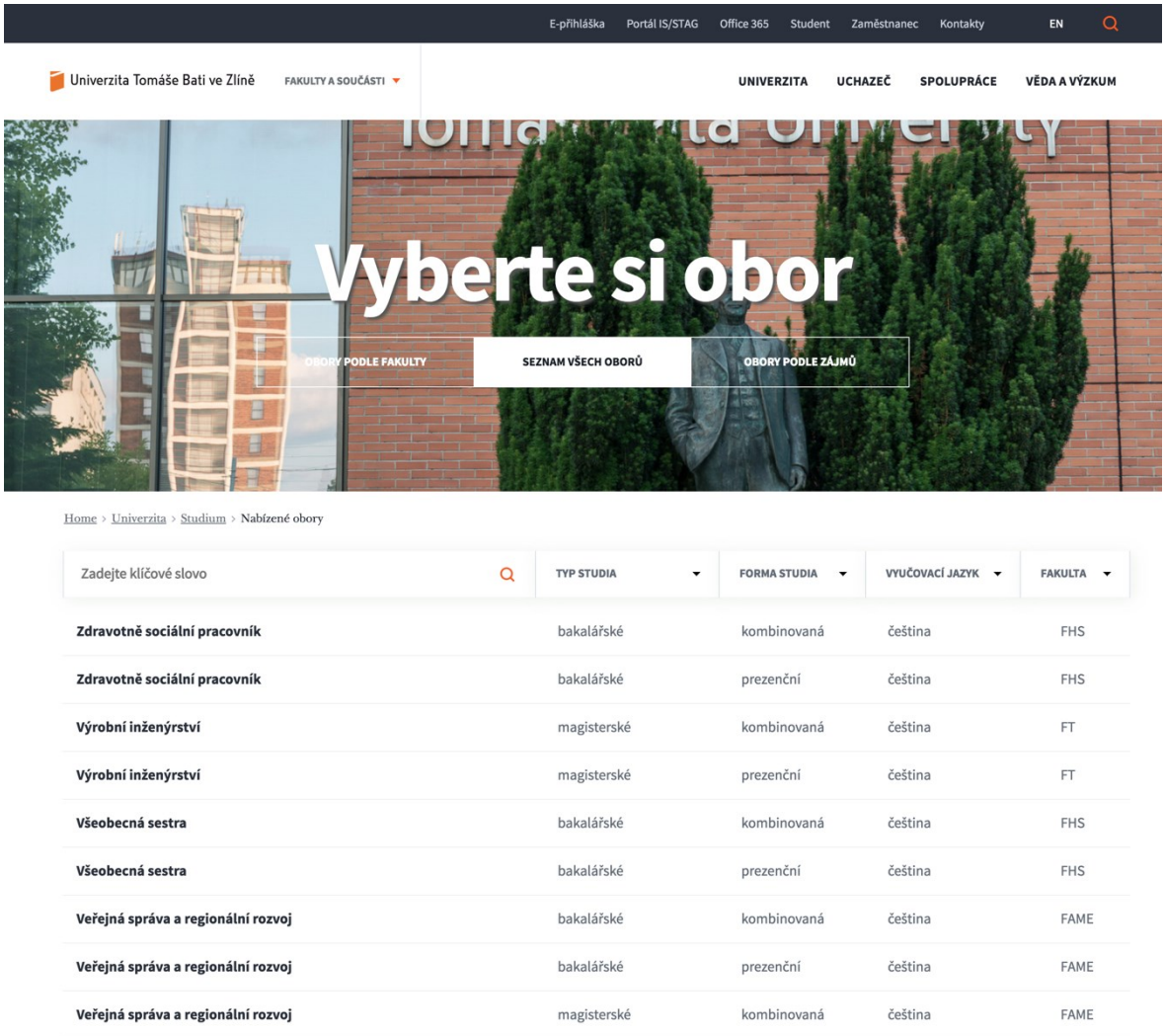
Grafický návrh jsem již měl připravený od UTB, na mě bylo naimplementovat 3 typy výběru oborů. První výběr oboru je podle fakulty, kde vidíme na obrázku č. 7, že je zde možnost filtrace podle:

- Fakulty nebo součásti – nachází se zde všech 6 fakult UTB, navíc je zde Univerzitní institut
- Typ studia – bakalářské, magisterské nebo doktorské
- Forma studia – prezenční a kombinovaná
- Jazyk – čeština a angličtina



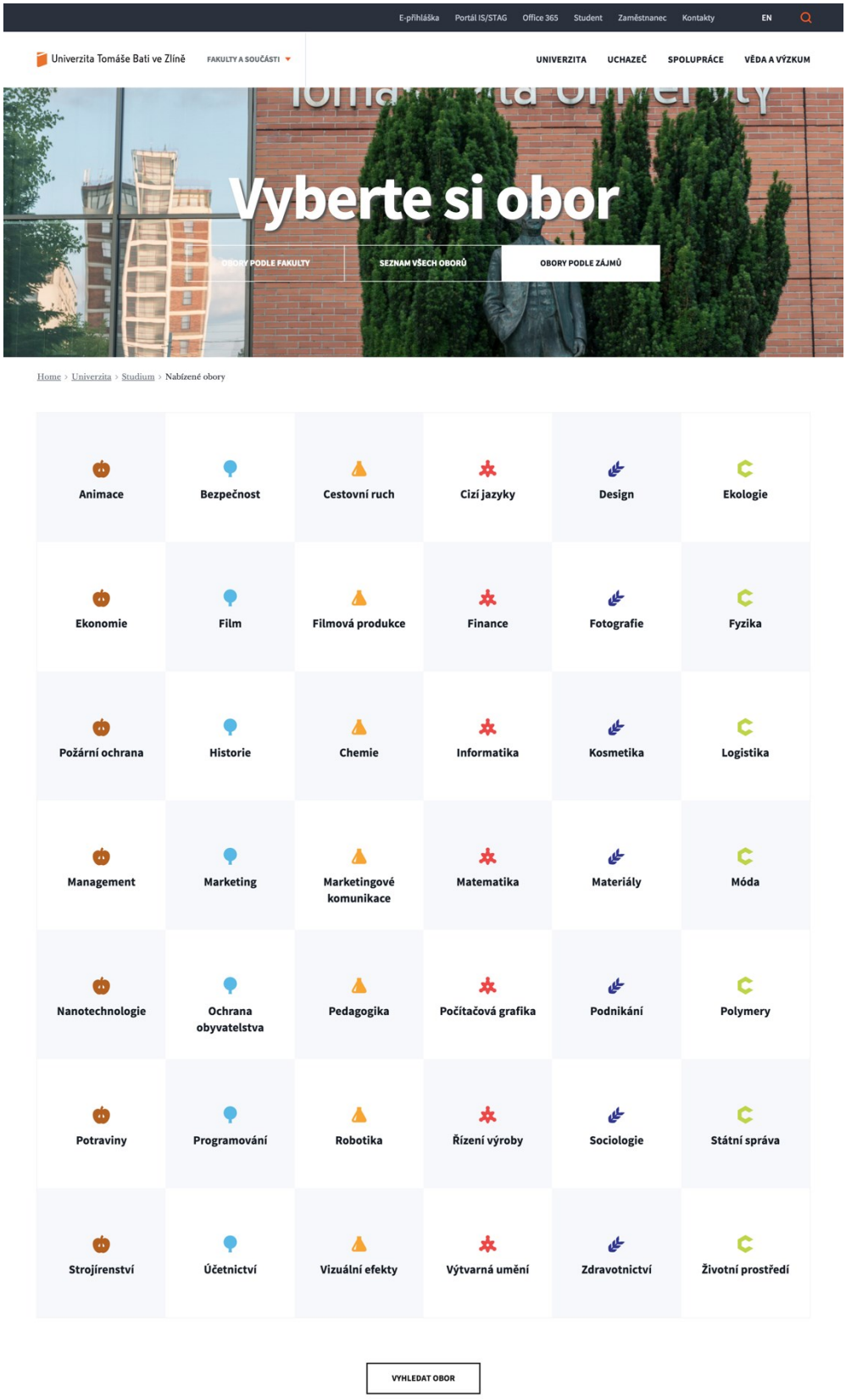
Obrázek č. 7 – Detail výběru oboru podle fakulty

Druhý typ výběru oboru je nejkomplexnější, je to seznam všech oborů. Pomocí filtrací a vyhledávání si může vybrat přesně to, co potřebujeme. Výběr je postaven na technologii AJAX, takže to časově usnadní výběr oboru. Možnosti filtrace tu máme obdobné jako u prvního výběru, navíc je zde akorát možnost vyhledávání. Na mobilní verzi je výběr omezen pouze na vyhledávání, typ studia a formu studia.



Obrázek č. 8 – Seznam všech oborů

Poslední na výběr oboru je dle mě nejpraktičtější, protože se jedná o výběr dle zájmů. Jak můžeme vidět na obrázku č. 9, můžeme si vybrat libovolný počet zájmů a vyhledávání nám najde všechny obory, u kterých byly tyto klíčová slova označena.



Obrázek č. 9 – Výběr oborů dle zájmů

5.4 Testování

Testování aplikace proběhlo na lokálním serveru, kde byl nainstalován WordPress. Následně jsme v administraci WordPressu aktivovali naimplementovaný plugin Load Study Courses.

První testování probíhalo tak, že jsem pomocí pluginu naimportoval nejprve studijní programy, které jsem následně kontroloval pomocí informační portálu UTB IS/STAG, kde jsou seznamy všech studijních oborů. Dále proběhl import studijních oborů, tady už kontrola byla složitější, protože bylo naimportováno 242 oborů, z toho 37 ve formě konceptu (neplatný obor). Následně jsem si náhodně vybral 20 oborů, které jsem kontroloval opět pomocí informačního portálu IS/STAG, zde se kontrolovalo především to, jestli je obor správně zařazen do Bc., Mgr. nebo Ph.D. programu, jestli má označenou správnou hierarchickou strukturu.

Druhý typ testování probíhal v rámci front-endu, zde máme tři typy výběru, u kterých se kontrolovala správná filtrace oborů. Ve výběru podle fakulty jsem postupně zkoušel všechny možnosti výběru i jejich různé kombinace, které jsem poté kontroloval pomocí administrace WordPressu, kde jsem měl data naimportována.

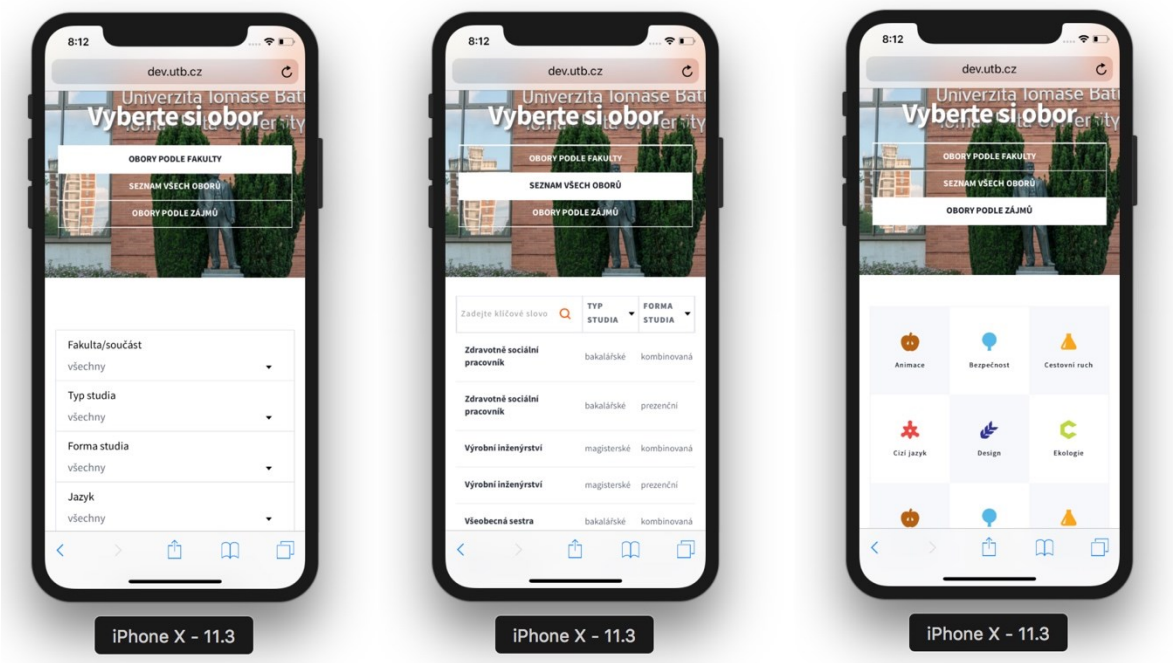
Výběr oborů pomocí filtrací je z nabízených možností výběru technicky nejsložitější, protože se zde data vybírají pomocí technologie AJAX. Jinak principiálně je tento výběr a následná moje kontrola obdobná jak u první volby. Pomocí filtrace jsem vybíral obory, kde jsem pak zkontroloval v administraci WP, jestli výběr a označené možnosti u daného oboru souhlasí.

Kontrola výběru podle zájmů byla na testování nejpřívětivější, protože jsem si vybral v administraci WordPressu obory, u kterých jsem označil daný zájem a poté ve front-endu vyhledával.

Jako poslední bych nechtěl zapomenout na otestování responzivity pluginu a její front-endové části. Jelikož aplikace funguje na lokálním serveru, tak zde nebyla možnost aplikaci otestovat přímo fyzicky na mobilním zařízení, proto jsem k testování využil aplikaci Simulator, která je součástí balíčku aplikace Xcode.

Na obrázku č. 10 můžeme vidět, že responzivita ve front-endové části funguje v pořádku.

Testování proběhlo v pořádku, všechny importovaná data souhlasila s informačním portálem IS/STAG. Následné vyhledávání oborů opět souhlasilo s možnostmi, které byly u oboru označeny.



Obrázek č. 10 – Mobilní zobrazení front-endové části

ZÁVĚR

Cílem bakalářské práce bylo vytvořit plugin, který bude importovat data pomocí datového formátu JSON a následně tyto data vypisovat.

V teoretické části jsem popisoval informace o redakčním systému WordPress, jako jsou základní informace, historii a strukturu. Dále jsem zde zmínil důležité informace o WordPress API a jeho podsekce WordPress REST API, která byla pro nás podstatná kvůli importu dat do administrace WordPressu. Jsou zde taky základní informace o šablonách a pluginech.

V praktické části jsme programovali plugin, který byl zadán určitými požadavky. Mezi ně patřilo importovat všechny obory Univerzity Tomáše Bati ve Zlíně, následně je hierarchicky zařadit do formy, typu a jazyku studia. Těmto importovaným datům jsme pomocí front-endové části webu dali možnost výběru. Máme 3 typy výběru, kde u prvního vybíráme podle fakulty, kde si můžeme vybrat formu, typ a jazyk studia. Další typ vyhledávání je pomocí filtrace seznamu oborů, která je založená na technologii AJAX. Tento typ filtrace bude podle mě z uživatelského pohledu dost používaný, protože je díky technologii AJAX nejrychlejší. Poslední typ výběru oboru je podle typu zájmů, kde si uživatel vybere předdefinované zájmy a pomocí přiřazení zájmů na daný obor se vyhledají obory s označenými zájmy.

V posledním bodě proběhlo testování aplikace, jak administrační, tak i front-endové části

SEZNAM POUŽITÉ LITERATURY

[1] GAVALDA, Mark. *The History of WordPress, its Ecosystem and Community* [online]. [cit. 2018-05-20]. Dostupné z: https://kinsta.com/learn/wordpress-history/

[2] *Usage Statistics and Market Share of Content Management Systems for Websites, May 2018* [online]. [cit. 2018-05-20]. Dostupné z: https://w3techs.com/technologies/overview/content_management/all

[3] *WordCamp Central: About WordCamps* [online]. [cit. 2018-05-20]. Dostupné z: https://central.wordcamp.org/about/

[4] *WordPress Files and Directory Structure - Interserver Tips* [online]. [cit. 2018-05-20]. Dostupné z: https://www.interserver.net/tips/kb/wordpress-files-directory-structure/

[5] *Build Your Website: WordPress File And Directory Structure Explained* [online]. [cit. 2018-05-21]. Dostupné z: http://www.build-your-website.co.uk/wordpress-file-directory-structure/

[6] *WordPress Codex: WordPress APIs* [online]. [cit. 2018-05-21]. Dostupné z: https://codex.wordpress.org/WordPress_API%27s

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

[8] POLLOCK, John. *THE ULTIMATE GUIDE TO THE WORDPRESS REST API* [online]. Torque Magazine, 2015 [cit. 2018-05-21]. Dostupné z: http://cdn2.hubspot.net/hubfs/298401/Documents_for_download/WP-API-ebook_final_09162015.pdf

[9] PA Quick Start Guide To The WordPress REST API. *WP Superstars* [online]. [cit. 2018-05-21]. Dostupné z: https://www.wpsuperstars.net/wordpress-rest-api/

[10] *Michal Blažek: WP REST API – Praktický průvodce od úplných začátků* [online]. [cit. 2018-05-21]. Dostupné z: https://www.michalblazek.cz/wordpress/wp-rest-api-prakticky-pruvodce-pro-zacatecniky

[11] WordPress template hierarchy. In: *WordPress Developer Resources: Template Hierarchy | Theme Developer Handbook* [online]. [cit. 2018-05-21]. Dostupné z: https://developer.wordpress.org/files/2014/10/wp-hierarchy.png

[12] MCCOLLIN, Rachel a TESSA BLAKELEY SILVER. *WordPress 3.2 Theme Development*. 3rd ed.,New Edition. Birmingham: Packt Publishing, Limited, 2013. ISBN 9781849514224.

[13] *WordPress Developer Resources: What is a Plugin? | Plugin Developer Handbook* [online]. [cit. 2018-05-21]. Dostupné z: https://developer.wordpress.org/plugins/intro/what-is-a-plugin/

[14] *Co je to HTML a jak jej využít na svém webu?* [online]. [cit. 2018-05-21]. Dostupné z: http://podpora.goneo.cz/535451-7Co-je-to-HTML-a-jak-jej-využ%C3%ADt-na-svém-webu

[15] NIXON, Robin. *Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5*. Fourth edition. Sebastopol, CA: O'Reilly Media, 2014. ISBN 978-1-491-91866-1.

[16] *WordPress Plugin Boilerplate Generator* [online]. [cit. 2018-05-23]. Dostupné z: https://wppb.me

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CMS	Content Management System
API	Application Programming Interface
JSON	JavaScript Object Notation
REST	Representational state transfer
WYSIWYG	What you see is what you get
HTML	HyperText Markup Language
WP	WordPress
GPL	General Public License
SVN	Subversion
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
CSRF	Cross-site Request Forgery
XSS	Cross-site scripting
AJAX	Asynchronous JavaScript and XML
XML-RPC	eXtensible Markup Language remote procedure call
XML	eXtensible Markup Language
PHP	Hypertext Preprocessor
CSS	Cascading Style Sheets
UTB	Univerzita Tomáš Bati
IS/STAG	Informační portál
Bc.	bakalář
Mgr.	magistr
PhD.	doktor
ID	IDentification

SEZNAM OBRÁZKŮ