Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Integrace vybraných modulů automatického zpracování dokumentů do portálu Porta fontium

Plzeň 2020 Jiří Plaček

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. srpna 2020

Jiří Plaček

Abstract

This thesis aims to integrate new modules for automatic documents processing into Porta fontium portal. The first part of this thesis is dedicated to exploring content management system (CMS) Drupal, which is used by Porta fontium web portal. In the next part analysis of this specific deployment is performed. In this analysis already installed modules, which could be used for development the new ones, are explored. Also the modules, that are affecting CMS behaviour in areas, where the new modules will be operating, are described. Next part of this thesis is describing design, creation and integration of newly created modules. Last part is dedicated to testing these modules and theirs modification for using in newer version of Drupal CMS.

Abstrakt

Cílem této práce je integrace nových modulů pro automatické zpracování dokumentů do webového portálu Porta fontium. První část práce se věnuje průzkumu redakčního systému Drupal, který je webovým portálem Porta fontium využíván. V další části je provedena analýza této konkrétní instalace redakčního systému. V rámci této analýzy jsou prozkoumány nainstalované moduly, které by mohly být využity při vývoji nových modulů, integrujících novou funkcionalitu. Zároveň jsou popsány moduly ovlivňující chování redakčního systému v částech, do kterých budou nové moduly zasahovat. Následuje návrh, vytvoření a integrace nových modulů do webového portálu. Závěr práce je věnován testování těchto modulů a jejich úpravě pro použití i v novějších verzích redakčního systému Drupal.

Obsah

1 Úvod

Státní oblastní archiv v Plzni a Generální ředitelství státních bavorských archivů v Mnichově realizují společný projekt, který se snaží sjednotit archivní fondy těchto dvou zemí do jednoho rozsáhlého virtuálního celku. V rámci tohoto projektu je provozován webový portál Porta fontium[1], umožňující bezplatné vyhledávání archiválií, uložených v různých západočeských a bavorských archivech. Také dochází k digitalizaci nového obsahu, jako jsou matriky či dobová periodika. Část těchto archiválií je již nyní alespoň částečně zpracována v podobě naskenovaných snímků, které je možné v rámci zmíněného webového portálu procházet.

V rámci prováděné digitalizace vznikají potřeby jednoduššího a rychlejšího zpracování takto digitalizovaných archiválií, aby bylo možné efektivně zpřístupnit další části veřejnosti. Cílem této práce je integrace nové funkcionality, umožňující automatizaci procesu přenosu textu z naskenovaných snímků do digitálního obsahu. Pro tento účel bude využito optického rozpoznávání znaků (OCR), které umožní právě urychlení a zjednodušení vytváření digitálního obsahu.

Zároveň má práce za cíl poskytnout přesnější vyhledávání informací uživatelům webového portálu, kteří budou chtít tento obsah procházet. Do portálu bude tedy také integrováno externí rozhraní (API), které k poskytnutému slovu dokáže určit jemu podobná slova (synonyma). O tato podobná slova je poté možné rozšířit vyhledávací dotaz a poskytnout uživateli portálu relevantnější výsledky obohacené o získaná synonyma.

Protože webový portál využívá redakční systému Drupal, bude nová funkcionalita poskytnuta vytvořením nových modulů, případně úpravou modulů již existujících. Zároveň budou tyto nově vzniklé moduly řádně otestovány a nasazeny do testovacího prostředí, kde je bude mít k dispozici provozovatel portálu. Protože aktuální verze redakčního systému používaná webovým portálem Porta fontium je již téměř deset let stará a v tuto chvíli existují dvě novější verze, budou také popsány úpravy potřebné pro provoz modulů v novější verzi redakčního systému.

Úvodní část práce se zabývá redakčním systémem Drupal jako takovým, jeho popisem a popsáním základních vývojářským pilířů, které by měl začínající vývojář znát. V další části práce je provedena analýza konkrétní instalace redakčního systému pro webový portál Porta fontium, včetně analýzy

[1] www.portafontium.eu

vybraných modulů relevantních pro splnění cílů této práce. Závěr práce je poté věnován vyvinutým modulům, jejich otestování a přípravě na integraci do novějších verzí redakčního systému.

2 Redakční systém Drupal

Drupal je open-source systém pro správu obsahu (CMS). Systém je distribuovaný pod licencí GNU General Public License, díky které je kompletní zdrojový kód zvěřejněný a je možné ho volně využívat a upravovat. První oficiální verze vznikla v roce 2001 [7] a v letošním roce by měla být uvolněna verze 9 [5]. Celý systém je napsán v jazyce PHP. Největší nárůst uživatelů zaznamenal Drupal s verzí 7, která vyšla v roce 2011. Tato verze je mezi uživateli tohoto systému nejrozšířenější, jak je možné vidět v tabulce 1 [10].

5.x	**6.x**	**7.x**	**8.x**
1 704	37 876	729 918	334 126

Tabulka 1: Počet unikátního užití jádra Drupal k 5.1.2020

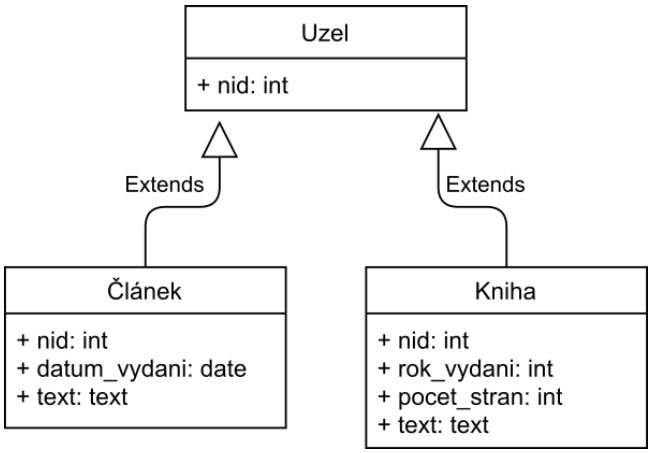
Čísla, zobrazená v tabulce, jsou pouze odhadem, protože se započítávají pouze stránky, které jsou dostupné z internetu a používají modul *Update*. Ačkoliv Drupal není natolik rozšířený jako například redakční systém Wordpress [4], používají Drupal společnosti jako BBC, MTV UK nebo Česká televize a Český rozhlas [1, 6]. Obecně je Drupal vnímán jako náročnější na počáteční orientaci, ovšem poté, co se uživatel naučí se systémem pracovat, dokáže Drupal poskytnout maximální míru flexibility. Aktuálně je pro Drupal nabízeno více jak čtyřicet pět tisíc rozšiřujících modulů.

Portál Porta fontium využívá Drupal ve verzi 7. Tato verze přinesla kompletně přepsané uživatelské rozhraní, které uživatelům zjednodušilo každodenní práci a zároveň přidala podporu pro databáze SQLite, PostgreSQL a MySQL/MariaDB.

Od verze 7 je většina redakčního systému založena na takzvaných entitách (*entity*). Tyto entity jsou následně rozdělovány na jednotlivé typy. Mezi základní typy entit v systému patří následující typy:

- uživatel (*user*)
- komentář (*comment*)
- soubor (*file*)
- slovník (*taxonomy vocalbulary*)
- termín (*taxonomy term*)
- uzel (*node*)

Mimo typy uvedené výše je možné přidat do systému vlastní typ entity. Typy entit mohou být dále tříděny do svazků (*bundle*). Některé typy entit mají ovšem pouze jeden svazek. Jako příklad lze uvést entitu *uživatel*, která obsahuje pouze stejnojmenný svazek. Každému svazku je dále možné nadefinovat vlastní pole (*field*), do kterých se ukládají informace.



Obrázek 2.1: Diagram členění entit a svazků

Obrázek 2.1 demonstruje, jak vypadá členění entit dle návrhu OOP. Třída Uzel je typ entity *uzel*. Třídy Článek a Kniha jsou svazky, které rozšiřují daný typ entity o další pole. V Drupalu je díky tomuto návrhu možné jednoduše vytvářet vlastní typy entit, které využívají funkce pro ukládání, načítání a obecně pro práci s entitami, poskytované přímo redakčním systémem [17].

2.1 Moduly

Celý redakční systém je založen na modulech. I samotné jádro Drupalu je složeno z modulů, které poskytují požadované funkce. Protože v tuto chvíli existují desítky tisíc modulů, je vždy vhodné nejdříve prozkoumat, zda některý z již existujících modulů nedokáže splnit požadavky na nové funkce a nebo alespoň poskytnout základ, pro vývoj nového modulu. Analýze existujících vhodných modulů se věnují podkapitoly v kapitolách (4 a 5) zaměřené na integraci modulů. Následující podkapitoly jsou věnovány vývoji nového modulu, aby měl čtenář pro nastínění alespoň základní náhled do jeho vývoje.

2.1.1 Vytvoření nového modulu

Pokud neexistuje žádný modul, který by dokázal poskytnout požadované vlastnosti, je nutné vytvořit modul vlastní. Tvorba nového modulu začíná vytvořením adresáře umístěného v adresáři určeném pro moduly, které nejsou součástí jádra redakčního systému. Adresář s těmito moduly je umístěn v *DRUPALROOT/sites/all/modules*. Jméno adresáře je zkráceným jménem modulu, které se bude používat i později v názvech funkcí. Dané jméno smí obsahovat pouze malá písmena a podtržítko. Po vytvoření adresáře je nutné vytvořit dva následující soubory:

- muj_modul.info
- muj_modul.module

První soubor *muj_modul.info* obsahuje informace o modulu. Minimální informace, které musí soubor poskytnout jsou název modulu, popis a verze, pro kterou je určen. Ukázka obsahu jednoduchého *.info* souboru:

```
name = Muj Modul
description = Popisek mého nového modulu
core = 7.x
```

Kromě výše zmíněných povinných vlastností lze modulu v *.info* souboru určit i další vlastnosti. Mimo jiné je zde možné určit soubory obsahující vlastní CSS styly nebo soubory s javascriptovým kódem, který se má přidat do každé stránky modulu. Také je zde možné nastavit moduly, na kterých je modul závislý [20].

Druhý soubor s názvem *nazev_modulu.module* obsahuje kód modulu. Tento soubor je uvozený PHP tagem `<?php`, neobsahuje ovšem jeho ukončení. Po vytvoření těchto souborů stačí vyčistit cache paměť (*Nastavení -> Vývoj -> Výkon*) redakčního systému a vlasní modul bude následně zobrazen v seznamu nainstalovaných modulů [19].

2.1.2 Systém háčků (*hooks*)

Redakční systém Drupal je založený na vytváření funkcí nazývaných háčky. V rámci syntaxe PHP jde o běžné PHP funkce. Tyto funkce ovšem umožňují interakci s ostatními nainstalovanými (a zapnutými) moduly nebo přímo s jádrem redakčního systému. Jádro redakčního systému obsahuje přes tři sta háčků [11]. V době psaní této práce existují dva typy háčků, první (*alter hook*) umožňuje změny konkrétních objektů tím, že jsou tyto objekty

předané jako reference. Druhý typ (*interception hook*) nedovoluje předávat proměnné referencí a je možné ho tedy využít pouze pro provádění konkrétních akcí. Každý z těchto háčků je možné v rámci vlastního modulu implementovat. Například systémový háček *hook_help()* slouží k zobrazení nápovědy k modulu. V případě, že modul implementuje tento háček, je v seznamu modulů zobrazeno tlačítko umožňující zobrazit nápovědu k modulu. Aby byl háček implementován, postačí když modul ve svém *.modul* souboru vytvoří stejnojmennou funkci a slovo *hook* nahradí svým názvem. Pokud by se tedy modul jmenoval například *muj_modul*, tak při implementaci háčku *hook_help* vytvoří funkci *muj_modul_help*:

```
function muj_modul_help($path, $arg) {
  if($path == "admin\help#muj_modul" ) {
    return t("Nápověda mého modulu")
  }
}
```

Takto implementovaný háček zajistí, že se v seznamu modulů, u daného modulu, objeví tlačítko *Nápověda* a toto tlačítko otevře stránku s textem, vráceným uvedenou funkcí. Díky dalším háčkům je poté možné měnit obsahy stránek (formulářů), menu nebo provádět akce například během vytváření nového obsahu. V implementaci háčku lze vidět použití funkce *t()*. Tato funkce slouží k identifikování textů, které se mohou vyskytovat ve více jazycích a umožňuje redakčnímu systému Drupal zobrazit předaný text ve správném jazyce.

Mimo implementace již existujících háčků je také možné vytvářet háčky vlastní. V místě, kde by měl být nový háček zavolán, jsou zavolány funkce *module_invoke_all* nebo *drupal_alter*. V případě, že modul přidává do redakčního systému nový formulář, je typické, že tento formulář je umožněno dalším modulům změnit právě vytvořením vlastního háčku. Jako příklad lze uvést následující ukázku kódu:

```
$form[] = ...;
drupal_alter("uprav_formular", $form);
```

V proměnné *form* je uložena definice formuláře. Po uložení hodnoty do této proměnné je zavolána výše zmíněná funkce *drupal_alter*, které je jako parametr předán název pro nový háček a proměnná, která bude tomuto háčku předána. Od této chvíle může jakýkoliv modul v systému implementovat háček *hook_uprav_formular_alter*, ve kterém bude dostupný obsah proměnné *form*. V momentě, kdy v redakčním systém dojde k provádění

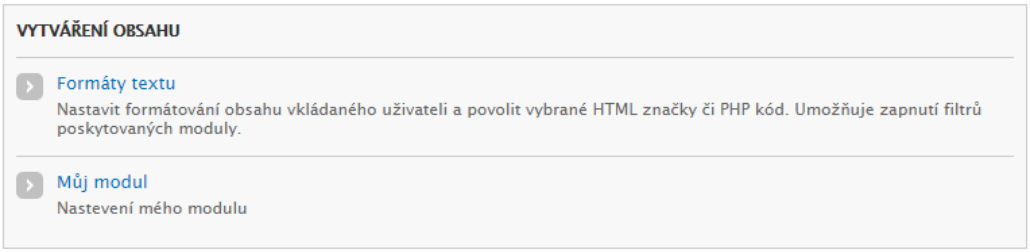
drupal_alter z ukázky výše, vyhledá redakční systém všechny moduly, které implementují háček *hook_uprav_formular_alter* a postupně jsou všechny tyto implementace zavolány. Obdobně je tomu i pro druhou uvedenou funkci *module_invoke_all*.

2.1.3 Konfigurační formulář

Při vytváření modulů vzniká často potřeba umožnit administrátorovi redakčního systému měnit některé vlastnosti modulu bez toho, aby musel upravovat samotný kód. Pro tyto úpravy slouží konfigurační formuláře. Prvním krokem, pro vytvoření konfiguračního formuláře, je zaregistrování místa (adresy), kde bude formulář dostupný. Nejčastějším, a jedním z nejjednodušších způsobů, je využití háčku *hook_menu*. Tento háček umožňuje nejen vložení odkazu do menu, ale lze také určit zpětná volání (*callbacks*) pro konkrétní adresy. Tímto je možné konkrétní adrese (nebo konkrétním adresám) definovat funkci, která je při jejich otevření zavolána. Následující ukázka kódu zobrazuje jednoduché vytvoření nového odkazu v administrátorském menu redakčního systému.

```
function muj_modul_menu() {
  $items = array();

  $items["admin/config/content/muj_modul"] = array(
    "title"            => "Můj modul",
    "description"      => "Nastavení mého modulu",
    "page callback"    => "drupal_get_form",
    "page arguments"   => array("muj_modul_form"),
    "access arguments" => array("access administration pages"),
    "type"             => MENU_NORMAL_ITEM,
  );

  return $items;
}
```

Vracené pole obsahuje konkrétní adresu, na které bude zobrazen formulář s nastavením. Adresa také definuje, ve kterém menu bude odkaz zobrazen. Takto použitá adresa bude zobrazena v Nastavení -> Vytváření obsahu -> Můj modul (obrázek 2.2). Při definování tohoto menu, se kromě názvu a popisu, určuje klíčem *page_callback* také funkce, která se při otevření stránky zavolá. V ukázce kódu je zavolána standardní funkce redakčního systému Drupal, která jako argument (klíč *page_arguments*) získá název funkce, která



Obrázek 2.2: Nová položka v nastavení redakčního systému

vrací formulář. Tato funkce vytvoří celý formulář a vrátí ho pomocí funkce *system_settings_form*.

```
function muj_modul_form($form, &$form_state) {
  $form["my_module_size"] = array(
    "#type"          => "textfield",
    "#title"         => t("Velikost"),
    "#description"   => t("Velikost stránky"),
    "#default_value" =>
      variable_get("custom_ocr_batch_size", 5)
  );
  ...
  return system_settings_form($form);
}
```

Při definování zadavatelných hodnot ve formuláři stačí při použití této funkce nastavit prvkům výchozí hodnotu (*default_value*) a redakční systém se sám postará o to, aby se po uložení formuláře, uložily uživatelem zadané hodnoty do perzistentních proměnných. Tyto proměnné lze následně přečíst ve funkcích modulu. Automaticky je také zobrazena zpráva o úspěšném či neúspěšném uložení nastavení. Ačkoliv je možné prvkům formuláře nastavit například maximální délku, která je pak automaticky validována, je vhodné provádět vlastní kontroly zadaných hodnot. Pro zajištění vlastní validace stačí vytvořit háček, jehož název se sestává z ID formuláře a přípony *_validate*. Validační háček je automaticky volán při pokusu o odeslání formuláře. V rámci tohoto háčku je možné přečíst zadané hodnoty a v případě chyby ne-

dovolit hodnoty zapsat a zobrazit chybovou hlášku (funkcí *form_set_error*).

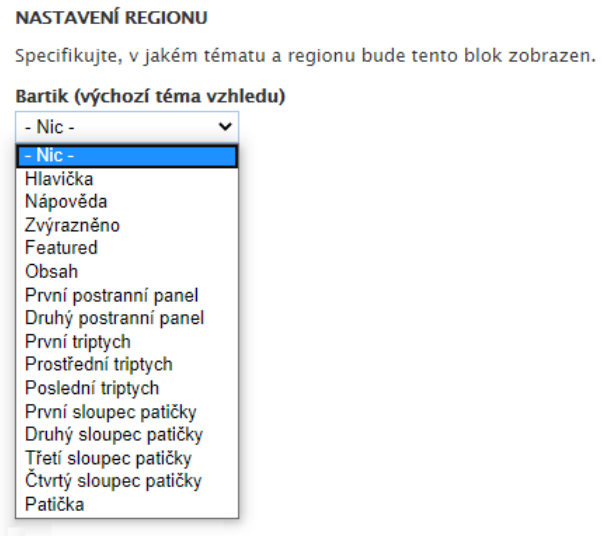
```
function my_module_form_validate($form, &$form_state){
  $size = $form_state["values"]["my_module_size"];
  if ($max_num > 9){
    form_set_error("my_module_size",
      t("Zadejte číslo menší než 10."));
  }
}
```

2.2 Bloky

Bloky jsou části, které mohou být vykresleny v různých částech stránky. Typickým příkladem je například zobrazení menu v levé části stránky nebo zobrazení patičky na konci stránky. Zobrazení bloků může administrátor redakčního systému upravit v nastavení bloků (*Struktura -> Bloky*). Zde se nachází seznam všech vytvořených a dostupných bloků. Blok může být přidán ručně přímo ze stránky se seznamem, případně může být vytvořen v kódu modulu. V obou případech se vytvořenému bloku nastavuje, ve kterém regionu a za jakých podmínek má být vykreslen. Zobrazení lze omezit pouze na konkrétní stránky, konkrétní typy obsahu nebo pro konkrétní role a jazyky (v případě, že stránka poskytuje vícejazyčnou verzi). Výše zmíněný region je část stránky definována vzhledem (obrázek 2.3). Autor konkrétního vzhledu rozděluje stránku na regiony, aby bylo možné do nich poté zobrazovat bloky nebo samotný obsah. Byl-li obsah vytvořen administrátorem formulářem, může mu administrátor nastavit obsah v HTML nebo PHP jazyce. Při vytváření bloku modulem jsou použity háčky redakčního systému a obsah je určen implementací těchto konkrétních háčků.

2.2.1 Vytvoření bloku modulem

Je-li blok vytvářen modulem, musí se o něm redakční systém dozvědět. Aby se blok dostal do povědomí, využívá se háček *hook_block_info*. V tomto háčku je určen název bloku a jeho vlastnosti. Následující kód přidá jednoduchý blok s identifikátorem *muj_blok*. Tento identifikátor je poté používán



Obrázek 2.3: Seznam regionů výchozího vzhledu

v dalších háčcích pro identifikaci daného bloku.

```
function muj_modul_block_info() {
  $blocks["muj_blok"] = array(
    "info" => t("Můj blok")
  );
  return $blocks;
}
```

Tato implementace je postačující k tomu, aby se blok zobrazil v seznamu dostupných bloků. Mimo jiné mohou být bloku nastaveny výchozí hodnoty, které jsou dostupné v uživatelském rozhraní, jako je například region, kde má být blok zobrazen nebo na kterých stránkách má být region zobrazen [3]. Tyto hodnoty slouží ovšem pouze jako výchozí hodnoty a administrátor je může v uživatelském rozhraní kdykoliv upravit.

Po informování redakčního systému o novém bloku je nutné vygenerovat jeho obsah. Obsah bloku je vracen háčkem *hook_block_view*. Parametrem tohoto háčku je identifikátor bloku, který byl zadán ve výše zmíněném *hook_block_info*. Při implementaci háčku je tedy potřeba zkontrolovat, zda je háček volán pro korektní blok a pokud ano, vrátit obsah bloku k vykreslení.

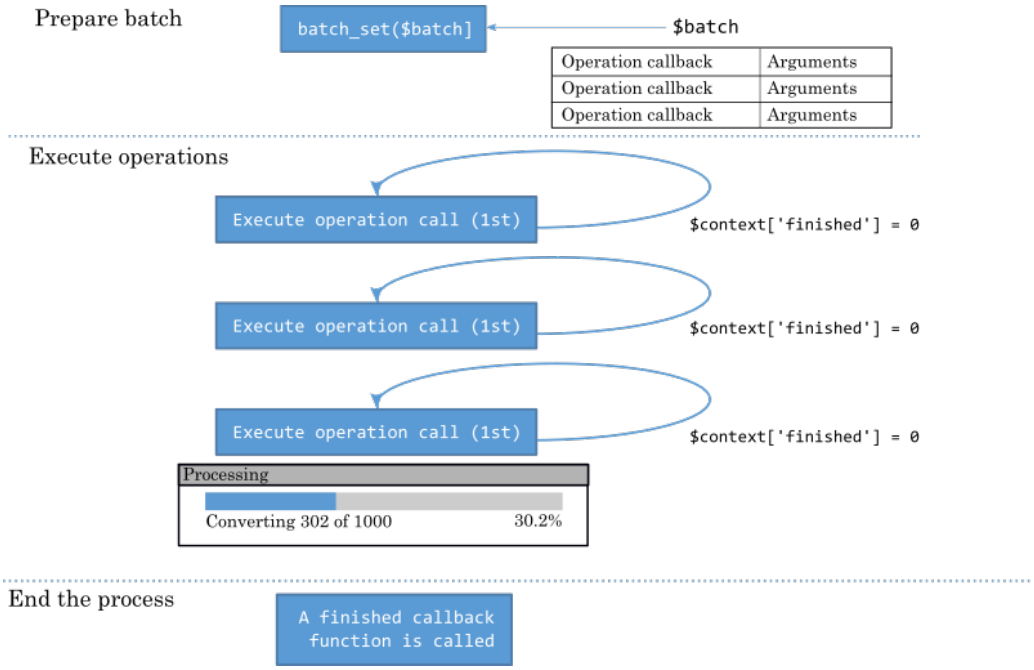
2.3 Dlouhotrvající operace

V některých případech je nutné provést déle trvající akci, jako je například hromadný import dat, hromadné mazání záznamů nebo indexace obsahu pro rychlejší vyhledávání. Protože taková akce může trvat déle než jen pár sekund, je důležité zobrazit uživateli informaci o tom, že operace stále probíhá a případně, kolik procent z operace je již hotovo. Pro tyto příležitosti existuje v rámci redakčního systému Drupal takzvané *Batch API*. Toto API poskytuje funkce umožňující dávkové spouštění operací a zajišťuje zobrazení informací o průběhu uživateli. Zároveň je zajištěno automatické obnovování stránky, díky kterému nedochází k vypršení časového limitu požadavku.

Operace (dávka) je spuštěna funkcí *batch_set*. Tato funkce má pouze jediný parametr a tím je pole obsahující informace o spuštěné operaci. Toto pole obsahuje název, seznam operací a název funkce, která je volána po dokončení. Seznam operací obsahuje názvy funkcí, které se mají volat a jejich parametry. Každé z těchto operací je kromě definovaných parametrů předán také parametr obsahující kontext. Tento kontext je pole sdružující informace o prováděné dávce. Měly by zde být ukládány výsledky jednotlivých operací a případně také data, která se mají přenášet mezi jednotlivými iteracemi. Kdyby se vývojář rozhodl ukládat data raději do globální proměnné session, mohlo by dojít k jejich nekonzistenci, pokud by uživatel během provádění dávky procházel další webové stránky redakčního systému. Operace ze seznamu jsou po spuštění dávky prováděny v pořadí dle seznamu. Každá operace se po dokončení začne provádět znovu a to až do doby, kdy je v kontextu nastaven příznak dokončení (*$context["finished"]*). Jakmile je operace dokončena a je nastaven příznak dokončení, posune se provádění na další operaci v seznamu. Při tomto přesunu je také aktualizována stránka s informacemi o průběhu dávky. Po dokončení všech operací je zavolána definovaná funkce, do které jsou předány výsledky. Provádění dlouhotrvající operace je přehledně znázorněno na obrázku 2.4.

2.4 Akce a spouštěče

Aby bylo možné provádět operace během různých systémových událostí, existují v redakčním systému takzvané akce. Kromě standardních akcí jako je například odeslání e-mailu nebo automatické publikování nového obsahu, je možné přidat vlastní akce. Nová akce musí být definována pomocí vlastního modulu (v tuto chvíli neexistuje možnost definovat nové akce pomocí uživatelského rozhraní). Akce reprezentuje funkci, která bude zavolána v případě, že v systému nastane konkrétní událost, jako je například přidání no-



Obrázek 2.4: Diagram provádění dlouhotrvající operace [2]

vého obsahu. Definice funkce se provádí pomocí háčku *hook_action_info*. Při přepisování tohoto háčku je nutné vrátit pole obsahují informace o nové akci, příklad lze vidět v následující ukázce kódu:

```
return array(
  "vlastni_akce" => array(
    "type" => "comment",
    "label" => t("Nová akce nad komentářem"),
    "configurable" => FALSE
  )
);
```

Akce, definovaná ukázkou kódu, je určena pro komentáře. Klíč *configurable* určuje, zda akce potřebuje další konfiguraci. V tomto případě je nastaven na hodnotu *FALSE*, tedy není potřeba další konfigurace. Pokud by měla akce nastaven opak, bylo by v rámci modulu potřeba definovat novou funkci s názvem *vlastni_akce_form*. Tato nová funkce by poskytovala formulář, který by umožňoval konfiguraci akce. Seznam dostupných akcí lze nalézt v menu Nastavení -> Systém -> Akce. Pokud by akce byla nastavena jako konfigurovatelná, nezobrazí se rovnou v seznamu akcí, ale je nutné ji přidat skrze volbu *Vytvořit pokročilou akci*, dostupnou ve spodní části stránky.

Nadefinované akce je možné dále přiřadit ke konkrétním událostem redakčního systému. Toto se děje na stránce Spouštěče (*Triggers*), dostupné v menu Struktura -> Spouštěče. Spouštěče jsou rozděleny dle modulů. Standardně jsou k dispozici moduly Komentář, Uzly, Systém, Kategorie a Uživatel. Každý z modulů má nadefinované spouštěče, ke kterým lze přiřadit akce. Například modul Uživatel nabízí spouštěče pro uložení nového uživatele nebo přihlášení již existujícího uživatele. Z vlastního modulu je samozřejmě možné přidat vlastní spouštěče. V následujícím kódu je pomocí háčku *hook_trigger_info* přidán nový spouštěč pro modul Uzel:

```
function muj_modul_trigger_info() {
  return array(
    "node" => array(
      "form_show" => array(
        "label" => t("Když je zobrazen konfigurační formulář")
      )
    )
  );
}
```

Při použití ukázky kódu je do stránky se seznamem spouštěčů přidán spouštěč nový. K tomuto spouštěči je možné přidat novou akci, která bude provedena ve chvíli, kdy je zavolána funkce *actions_do*, které jsou předány jako parametry ID funkcí, které se mají zavolat a objekt, nad kterým se spouštěč spustil.

2.5 Testování

Redakční systém Drupal nabízí vývojářům také řadu možností automatického testování vytvářených modulů. Pro účely automatického testování je v Drupalu verze 7 připraven modul *SimpleTest*, který nabízí tři různé typy testů:

- Funkční testy: Jde o nejběžněji používané testy. Při jejich použití se vytváří nová databáze a je testováno provádění akcí, které provádí uživatel webové portálu.
- Jednotkové testy: Slouží pro testování dílčích částí kódu a není pro ně nutné vytvářet novou databázi.
- Povyšovací testy: Tyto testy ověřují přechod na novější verzi redakčního systému Drupal.

Největší pozornost při testování v redakčním systému Drupal by měla být upřena na testování pomocí funkčních testů. Tyto testy testují modul a jeho funkcionalitu jako celek a jsou efektivnější než testování jednotlivých částí kódu samostatně.

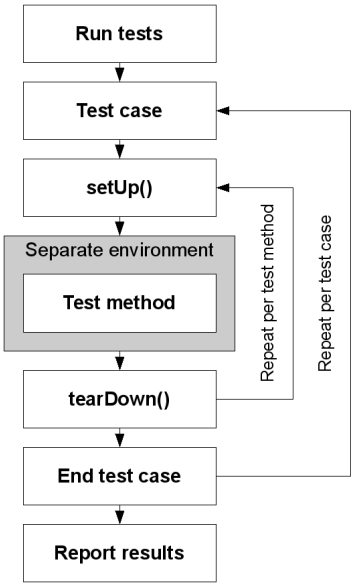
Tvorba nového testu začíná vytvořením souboru pojmenovaného ve stejných konvencích, jako tomu bylo doposud. Je tedy vytvořen soubor *nazev_modulu.test*, který je umístěn do adresáře modulu. Aby redakční systém získal informaci o tom, že je nutné tento soubor načíst, musí být informace o souboru zanesena také do *info* souboru modulu. Mimo dalších informací je tedy v *.info* souboru zapsán řádek *files[] = nazev_modulu.test*. V souboru s testy je pak definována nová třída, reprezentující testovaný scénář, která bude obsahovat jednotlivé testy. Název třídy je shodný s názvem modulu, pro který testy vznikají, s tím rozdílem, že namísto použití podtržítka jako oddělovače slov, se nepoužívá oddělovač žádný a každé slovo začíná velkým písmenem. Aby bylo možné využívat všech funkcí nabízených modulem pro testování, musí tato třída dědit od třídy *DrupalWebTestCase* (případně *DrupalUnitTestCase* pro jednotkové testování). Dále je v této třídě důležité definovat statickou funkci *getInfo*, která informuje redakční systém Drupal o detailech testovacího scénáře. Funkce vrací pole, jehož součástí je název a popis testovacího scénáře a název skupiny, do které bude scénář zařazen.

Následuje ukázka souboru testovacího scénáře:

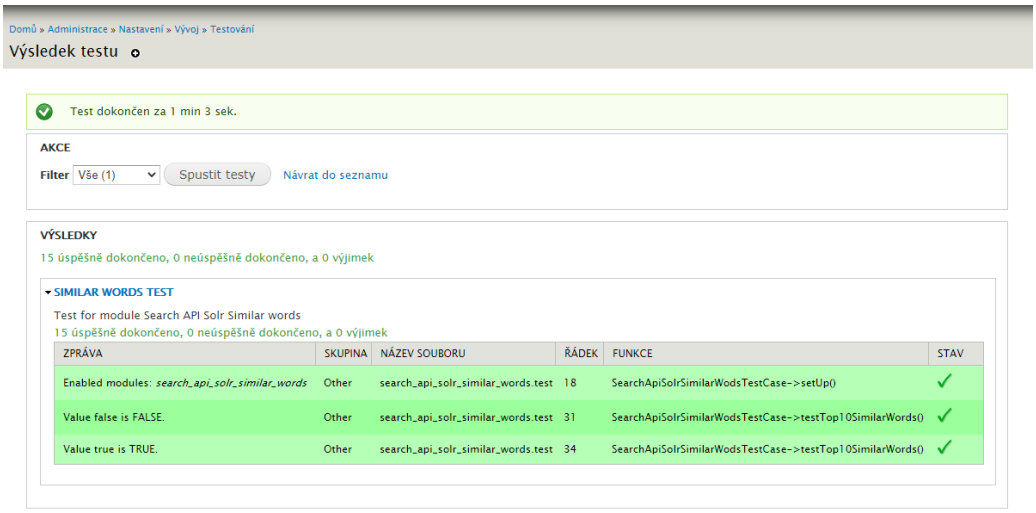
```
class MujModulTest extends DrupalWebTestCase {

  public static function getInfo() {
    return array(
      "name"        => "Ukázka testu",
      "description" => "Testovací scénář pro modul MůjModul",
      "group"       => "Příklady"
    );
  }

  public function setUp() {
    parent::setUp(array("muj_modul"));
  }

  public function testPrvniTest() {
    $this->assertTrue(TRUE);
  }
}
```

Dále je součástí třídy funkce *setUp*. Tato funkce je volána před provedením každého testu. Typicky je využívána pro vytváření potřebných dat (jako jsou například uživatelé nebo obsah webu) a aktivuje moduly potřebné pro testování. V ukázce kódu je vidět právě zmiňovaná aktivace modulu, který bude testován. Na konci třídy je funkce reprezentující jeden samostatný test (v tomto případě vždy úspěšný). Nově vytvořený testovací scénář se objeví v seznamu spustitelných testů po smazání dočasné paměti nebo po vypnutí a zapnutí modulu, jehož součástí test je. Tento seznam je dostupný v menu Nastavení -> Vývoj -> Testování. Seznam testů je dělen do skupin. Většina skupin kopíruje názvy modulů, ze kterých testovací scénáře pochází, ale není to pravidlem. Výše nadefinovaný scénář by se objevil ve skupině *Příklady*. Každý test ze scénáře je spuštěn nad novou prázdnou databází, aby jeho výsledky nemohly ovlivnit produkční prostředí nebo ostatní spouštěné testy. Samotné workflow si lze prohlédnout na obrázku 2.5 [15].

Po dokončení testování všech scénářů je zobrazen výsledek z testování. Ve výsledku jsou zobrazeny výsledky všech assert funkcí včetně souboru, třídy, funkce a čísla řádku, kde byla assert funkce zavolána (obrázek 2.6). Případně jsou zde zobrazeny také výjimky, které během provádění nastaly, či pomocné zprávy vložené autorem testu.



Obrázek 2.5: Workflow práce testů [9]



Obrázek 2.6: Ukázka výsledků po dokončení testování

Mimo assert funkcí, sloužících pro ověření výsledků testování, nabízí třídy testování i další funkce pro zjednodušení psaní testů. Mezi ně patří například:

- *drupalCreateNode*
- *drupalCreateContentType*
- *drupalCreateUser*
- *drupalLogin*
- *drupalPost*
- a další

Všechny tyto funkce usnadňují vývojářům práci a umožňují testovat způsobem, který se podobá chování uživatele na webovém portále.

V případě testování pomocí jednotkových testů, je většina těchto funkcí zbytečná. Jednotkové testy nepracují nad databází redakčního systému a jsou určené k testování samostatných částí kódu. V jejich případě se tedy ani neaktivují moduly, ale jsou pouze vkládány soubory vybraných modulů, aby bylo možné z nich volat požadované funkce.

3 Porta fontium

Webový portál Porta fontium vznikl v rámci projektu „Bavorsko-česká síť digitálních historických pramenů“, který se snaží o sjednocení archivních fondů. Mnoho dějinných událostí rozdělilo archiválie a spousta listin vtahujících se k České republice skončila po odsunu německého obyvatelstva za hranicemi. Stejně tak je mnoho listin, obsahujících sudoněmecké a bavorské dějiny, v místních českých archivech. Státní oblastní archiv v Plzni a Generální ředitelství státních bavorských archivů v Mnichově se v rámci projektu snaží o spojení těchto archivních souborů, jejich digitalizaci a zveřejnění prostřednictvím webového portálu pro širokou veřejnost.

V dalších kapitolách budou popsány části tohoto webového portálu, které byly důležité při návrhu nových modulů, vytvářených v rámci této práce.

3.1 Kolekce polí

Ačkoliv se obsah ve webovém portálu Porta fontium dělí na různé typy obsahu, existují taková pole, která jsou shodná pro velkou část těchto typů. Aby nebylo nutné zadávat pro každý nový typ obsahu pole znovu, využívají se kolekce polí (modul *Field Collection*, kapitola 3.3.1). Portál obsahuje čtyři kolekce (obrázek 3.1), z čehož dvě jsou pouze pro jeden typ obsahu (tudíž prozatím práci nezjednodušují) a zbylé dvě jsou obsaženy ve většině typů obsahu. První kolekce přidává k obsahu datumy. Obsahuje data od a do a dataci (ta většinou obsahuje rozsah let, například 1924 - 1928). Druhá kolekce obsahuje pole pro zadávání místa. Těchto polí je zde celkem třináct.

3.2 Obsah, výchozí vzhled a bloky

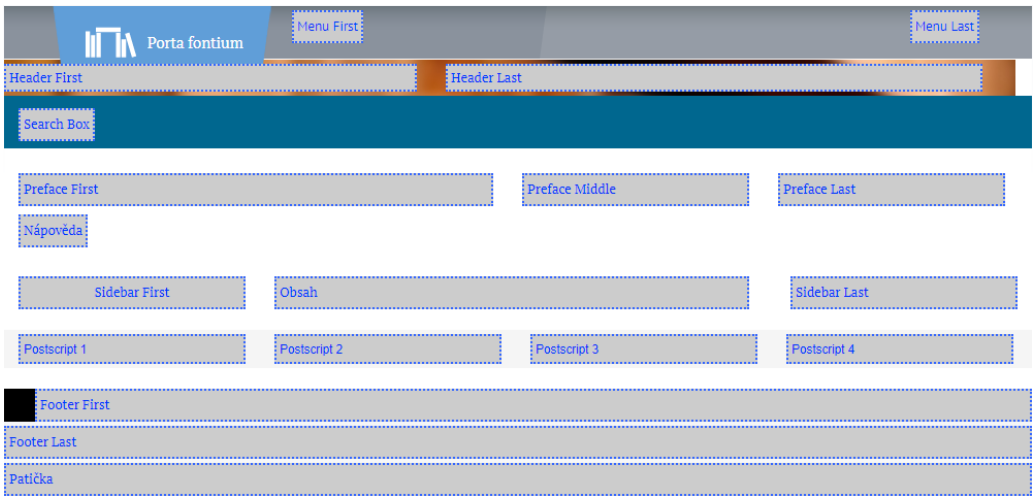
Během psaní této práce obsahoval portál Porta fontinum přibližně sedmnáct typů obsahu. Jako příklad lze uvést matriky, kroniky, fotografie nebo periodika. Každý z těchto typů má nadefinovaná různá pole a různé zobrazení při procházení existujícího obsahu. U většiny obsahu je například možné vidět fotky či obrázky k obsahu připojené. Periodika jsou pak při procházení členěna na jednotlivá periodika, rok vydání a konkrétní číslo v daném roce.

Jako výchozí vzhled portálu je použit vzhled *Porta Seven*. Tento vzhled vychází ze vzhledu *Seven*, který je jedním ze čtyř vzhledů dostupných přímo od vývojářů redakčního systému. Mimo jiné jsou zde oproti standardnímu

NÁZEV POLE	POUŽITO V	OPERACE	
field_doc_dates	Bavarika, Bohemika, Fotografie, Kronika, Listina, Lázeňští hosté, Matrika, Periodikum, Periodikum–číslo, Přihláška k pobytu, Sčítání lidu, Sčítání lidu – dům, Úřední kniha	správa polí	správa zobrazení
field_doc_place	Bavarika, Bohemika, Fotografie, Kronika, Listina, Lázeňští hosté, Matrika, Periodikum, Periodikum–číslo, Přihláška k pobytu, Sčítání lidu, Sčítání lidu – dům, Úřední kniha	správa polí	správa zobrazení
field_fc_accruals	Sbírka matrik západních Čech	správa polí	správa zobrazení
field_fc_map_page	Mapa	správa polí	správa zobrazení

Obrázek 3.1: Ukázka kolekcí polí portálu v portále Porta fontium

vzhledu přidány nové regiony. Celkem jich je nabídnuto devatenáct oproti pěti, které jsou dostupné ve standardním vzhledu (obrázek 3.2). Bloky přidané do těchto regionů umožňují zobrazovat specifické údaje pro některé typy obsahu (viz například zobrazení fotek jedním z modulů v kapitole 3.3.4).



Obrázek 3.2: Ukázka regionů ve vzhledu Porta Seven

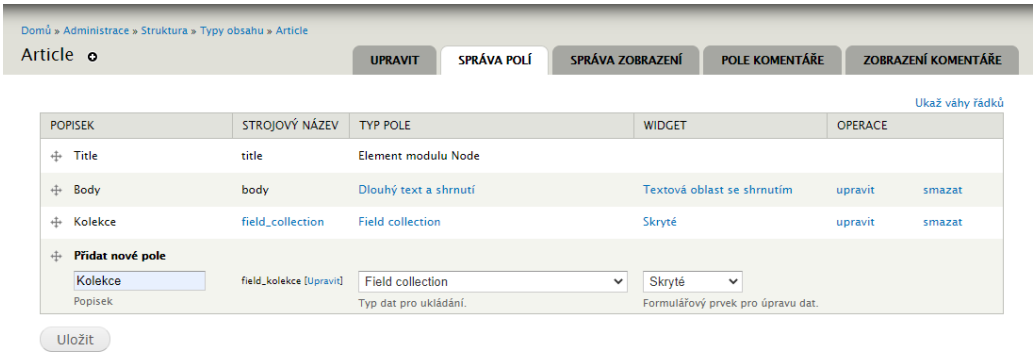
3.3 Relevantní moduly

Během realizace této práce bylo v redakčním systému Porta fontium nainstalováno více než tři sta modulů. Z toho jich bylo sto šedesát šest aktivních. Na začátku práce tedy bylo důležité moduly projít a prověřit, které z nich

by se daly využívat při integraci nové funkcionality. V rámci této kapitoly budou popsány moduly, které jsou relevantní a byly v rámci realizace vyžity nebo alespoň analyzovány pro případné využití. Protože portál Porta fontium využívá redakční systém ve verzi 7, jsou následující moduly popisovány tak, jak jsou distribuovány pro tuto verzi.

3.3.1 Field Collection

Field Collection je modul umožňující vytvářet kolekce polí tak, aby nebylo nutné přidávat shodné pole jednotlivě ke každému typu obsahu. Modul přidává novou entitu a nový typ pole, který je možné k volitelnému obsahu (entitě) přidat. Prvním krokem pro zahájení práce s modulem, je přidání tohoto nového typu pole (typ *Field collection* na obrázku 3.3). Na obrázku je vidět jedna již přidaná kolekce (první řádek s popiskem *Kolekce*) a další, která se zrovna bude přidávat (druhý a poslední řádek s popiskem *Kolekce*). Po přidání tohoto nového pole je možné dané pole přidat k dalším typům obsahů.



Obrázek 3.3: Přidání nové kolekce k obsahu

Po přidání pole tohoto typu se v seznamu kolekcí (v menu Struktura -> Field collections) zobrazí nová kolekce. V tomto seznamu je vidět název pole (z obrázku 3.3 jsou to pole *field_collection* a *field_kolekce*), a ve kterých typech obsahu je kolekce použita. Je zde také možnost přidat do této kolekce nová pole a nastavit jejich zobrazení. Tato nastavení jsou totožná s přidáváním a úpravou polí přímo pro konkrétní typy obsahu. Změny v kolekci se poté projevují ve všech typech obsahu, ke kterým je kolekce přidána.

3.3.2 Job Scheduler

Modul *Job Scheduler* poskytuje API pro plánování úloh v předem nastaveném čase nebo v pravidelných intervalech. Tento modul neposkytuje žádné vizuální nastavení, ani jiné formuláře. Poskytuje ovšem vývojářům dva háčky *hook_cron_job_scheduler_info* a *hook_cron_job_scheduler_info_alter*. Implementací prvního zmíněného lze vytvořit nový plánovač, který bude provádět definovanou akci. Ukázka použití háčku:

```
function muj_modul_cron_job_scheduler_info() {
  $schedulers = array();
  $schedulers["muj_modul_planovac"] = array(
    "worker callback" -> "muj_modul_funkce_planovace",
  );
  return $schedulers;
}
```

V ukázce kódu je zobrazeno nadefinování nového plánovače. Klíč do pole *schedulers* je identifikací nového plánovače. Tento plánovač poté definuje callback funkci, která je zavolána při spuštění. Kdy a jak často se má funkce volat, je možné definovat dvěma způsoby. Prvním způsobem je vložení dalšího pole k nadefinovanému plánovači. Toto pole je vloženo s klíčem *job* a obsahuje typ, ID a příznak, zda jde o periodickou akci. Jde-li o periodickou akci, musí být také obsažena perioda, po které se akce znovu naplánuje. Nejde-li o periodickou akci, musí být obsažena definice dle crontab syntaxe [23]. Příklad:

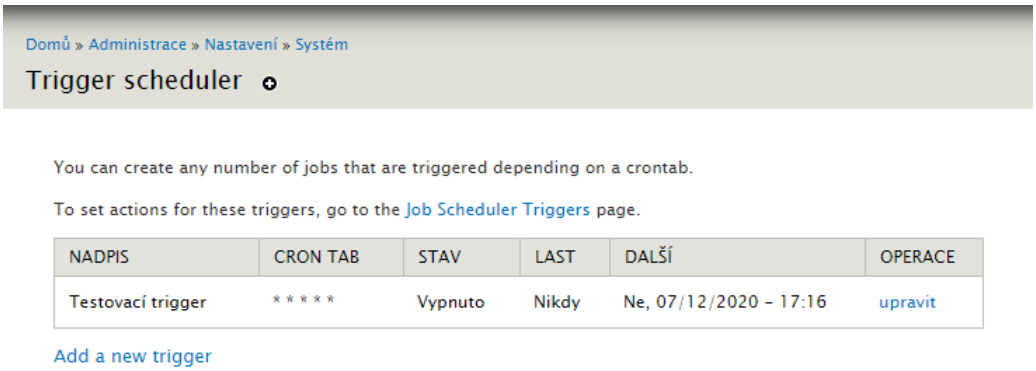
```
$job = array(
  "type" -> "story",
  "id" -> 2,
  "period" -> 1000,
  "periodic" -> TRUE,
);
```

Druhou možností je nastavení opakování přímo kódem a to zavoláním funkce *JobScheduler::get("example_unpublish")->set($job)*, kde *$job* je výše zmíněné pole. Je také možné namísto callback funkce použít klíč *queue name* a do něj zadat název fronty. Poté se namísto provedení funkce naplánuje zpracování této fronty.

Druhý háček umožňuje před spuštěním akcí změnit objekt plánovače. Do tohoto háčku je předáno pole se všemi plánovači a je tedy umožněno například upravit callback funkci, která má být provedena.

Job Scheduler Trigger

V rámci instalace modulu *Job Scheduler* se nainstaluje také modul *Job Scheduler Trigger*. Tento modul přidává do redakčního systému nový formulář s konfigurací vlastních spouštěčů. Formulář je dostupný v menu Nastavení -> Systém -> Trigger Scheduler. Uživatel v tomto formuláři může přidat nový spouštěč a nastavit mu, kdy se má spouštět. Takto přidaný spouštěč se objeví v seznamu spouštěčů a je mu možné přiřadit prováděnou akci. Spouštěče přidané tímto modulem se v seznamu spouštěčů zobrazují pod záložkou *Job Scheduler*.



Obrázek 3.4: Spuštěče vytvořené v modulu *Job Scheduler Trigger*

3.3.3 Feeds

Přidávání obsahu pomocí importu souborů zajišťuje v portálu Porta fontium modul *Feeds*. Instalací tohoto modulu se do redakčního systému nainstalují následující tři moduly:

- *Feeds*
- *Feeds Admin UI*
- *Feeds Import*

Hlavní modul *Feeds* poskytuje kompletní funkcionalitu pro import nového obsahu. Modul *Feeds Admin UI* tyto funkce rozšiřuje o uživatelské rozhraní. Importování nového obsahu je založeno na takzvaných importérech (*importers*). Seznam importérů je dostupný v menu Struktura -> Feeds importers. Zde je dostupný seznam všech dostupných importérů a lze je zde editovat nebo přidat zcela nové. Každý importér obsahuje několik důležitých nastavení. V základním nastavení se nastavuje jméno, popis importéru a doba, po

které se obsah periodicky importuje. Je zde také volba umožňující vytvoření obsahu automaticky ihned po importování dat. Uživateli je také umožněn výběr formuláře, ve kterém se volba pro import zobrazí. Je možné nastavit vlastní formulář, který je dostupný například v menu Obsah -> Import -> *Název importéru*. Případně je umožněno zvolit konkrétní typ obsahu a v takovém případě se na formuláři, při vytváření nového obsahu, zobrazí volba pro import dat.

Kromě základního nastavení se každému importéru nastavuje načítač (*fetcher*), parser a procesor. Načítače slouží k získání dat, která budou importována do redakčního systému. Ve výchozím stavu jsou k dispozici dva načítače. První slouží k importu souborů z konkrétní URL adresy. Druhý umožňuje import prostřednictvím uploadu souborů. V nastavení tohoto načítače je možné zvolit povolené přípony souborů, aby uživatel nemohl nahrávat libovolné soubory. Vybírá se také adresář, do kterého jsou soubory na server nahrány a zda se mají nahrané soubory ihned po importu smazat. Je možné soubory ihned po importu nemazat a ponechat je pro případné opakování importu bez toho, aby se soubor musel znovu nahrávat na server.

Po načítači je potřeba nastavit parser. Parser slouží k uspořádání dat do struktur, ze kterých je lze následně kódem dále zpracovávat. Další často prováděnou akcí parseru je kontrola, zda je struktura načtených dat v pořádku. Standardně jsou k dispozici čtyři různé parsery. Tři z těchto dostupných parserů slouží k parsování XML ve formátech běžných pro RSS. Tyto parsery nemají žádné další nastavení. Posledním nabízeným parserem je pak parser CSV souborů. U tohoto parseru je možné nastavit výchozí oddělovač a kódování. Data jsou pak z vybraného kódování převedena do UTF-8.

Posledním nastavením importéru je nastavení procesoru. Procesor zpracovává načtená data a provádí nad nimi definované akce. Nejčastěji z dat vytvoří nový obsah. K dispozici jsou výchozí procesory pro vytváření a aktualizaci obsahu, uživatelů a taxonomie. Každý z těchto procesorů má vlastní nastavení, definující, jak se bude při vytváření a aktualizaci objektů chovat. U všech tří procesorů lze nastavit, zda se má importem vytvářet nový obsah, či zda má dojít k aktualizaci již existujícího obsahu. Existuje také několik specifických nastavení pro jednotlivé procesory, jako například přidělení výchozí role vytvářeným uživatelům nebo nastavením, který typ obsahu se má vytvořit. Kromě tohoto nastavení je u procesorů také možné nastavit mapování. Mapováním je možné určit, který element z načtených dat se má uložit do kterého pole vytvářeného nebo aktualizovaného objektu.

Po nastavení importéru je možné začít importovat data z vybraných zdrojů. Kromě možného nastavení skrze uživatelské rozhraní, popsáno výše, je možné upravovat konfiguraci pomocí háčků. Modul nabízí háčky pro de-

finování vlastních načítačů, parserů i procesorů. Dále také poskytuje háčky pro úpravy již existujících, například háček volaný po importování souboru nebo před vytvořením nového obsahu.

Poslední z modulů, *Feeds Import*, obsahuje dva ukázkové importéry, díky kterým je možné vyzkoušet práci s celým modulem.

3.3.4 IIP Image

Moduly *IIP Image API*, *IIP Image Integration base* a *IIP Image Register* slouží k integraci prohlížeče obrázku. Moduly pracují s *IIPImage*, což je vysoce výkonný obrazový server, zaměřený na podporu webových aplikací a zobrazování obrázků s velmi vysokým rozlišením [12]. Obrázky na tomto serveru jsou ukládány ve formátu JPG2000. Pro zobrazování obrázků v prohlížeči je využíván prohlížeč *OpenSeaDragon*.

IIP Image API

První modul obsahuje funkcionalitu pro zobrazování obrázků a náhledů. Hlavním cílem tohoto modulu je generování HTML kódu, který je zobrazen uživateli v prohlížeči. Naprogramované funkce jsou volány z modulu *IIP Image Integration base*, který bude popsán dále. Do místních funkcí jsou v parametrech předávány údaje jako například obsah, který uživatel aktuálně prohlíží, a v těchto funkcích je na základě předaných údajů vytvářen obsah stránek, vracený zpět k zobrazení.

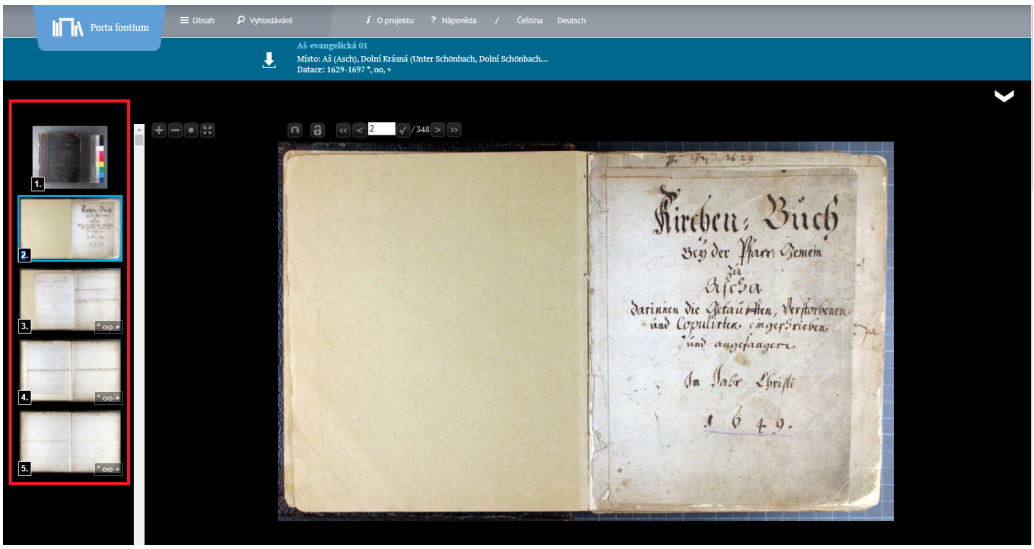
IIP Image Integration base

Tento modul slouží k integrování funkcionality (tj. zobrazení některých konkrétních funkcí uživateli) pro zobrazení obrázků v redakčním systému Drupal. V rámci modulu je vytvořena obrazovka s nastavením. V nastavení modulu je možné mimo jiné určit cestu k serveru s obrázky a adresář, kde jsou samotné obrázky umístěny. Modul pro zabudování funkcionality využívá funkcionalitu bloků, které redakční systém Drupal nabízí. Jsou zde vytvořeny tři bloky:

- *iipimage_base_thumbnails*
- *iipimage_base_download*
- *iipimage_base_tease*

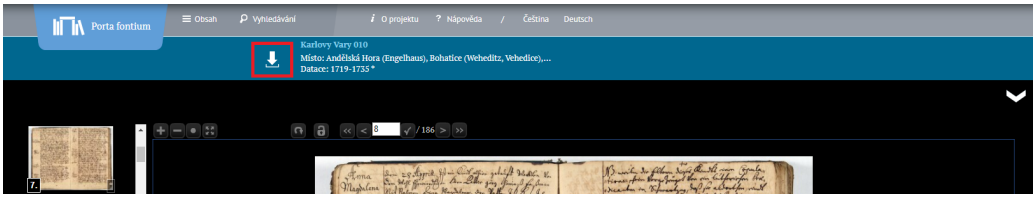
První zmíněný blok zobrazuje náhledy obrázků ke konkrétnímu obsahu. Náhledy jsou zobrazeny v bloku na levé straně obrazovky a lze je vidět

pouze na konkrétních (administrátorem redakčního systému nadefinovaných) stránkách. Například v případě, že jsou prohlíženy obrázky k matrice (obrázek 3.5).



Obrázek 3.5: Příklad zobrazení náhledu obrázků

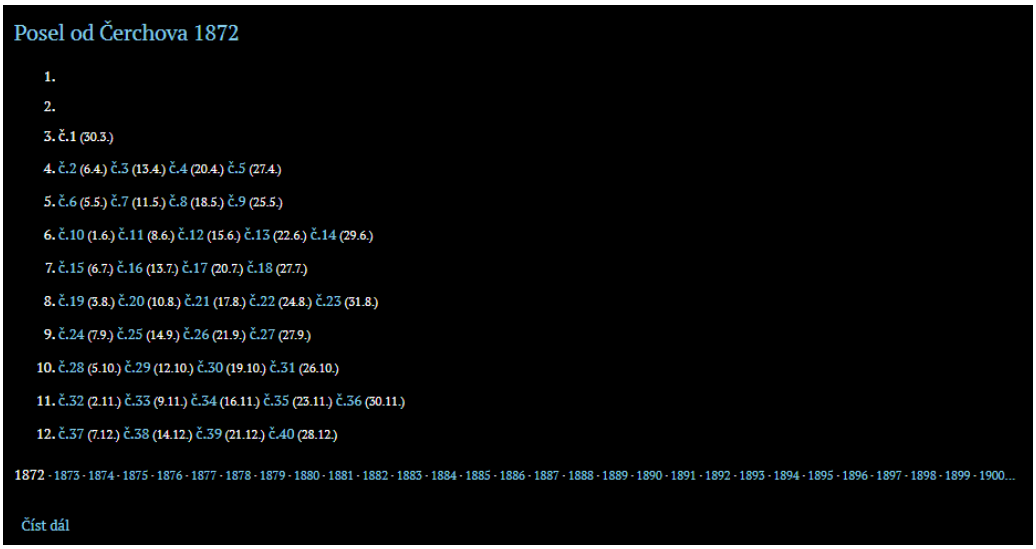
Druhý blok, *iipimage_base_download*, poskytuje možnost stáhnout některé z obrázků. Toto stažení je realizováno přidáním tlačítka pro stažení (obrázek 3.6). Toto tlačítko je zobrazeno v horní části stránky a pouze pro některé konkrétní obrázky. Konkrétně je zde nyní omezení na konkrétní adresy obrázků.



Obrázek 3.6: Příklad zobrazení tlačítka pro stažení obrázku

Poslední blok, *iipimage_base_teaser*, je umístěn v patičce stránky. Je zobrazován pouze při prohlížení obrázků a obsahuje informace o konkrétním obsahu. Například při prohlížení matrik jsou zde informace o matrice. Je zde vidět její datace, umístění nebo ze kterého archivu pochází. Naopak při prohlížení periodik je zde zobrazen seznam vydaných čísel v rámci jednotlivých let (obrázek 3.7).

Kromě zobrazených bloků, je v modulu dále implementována funkcionalita pro stažení obrázku. Při stažení je potřeba nasměrovat prohlížeč na správný server a získat z něj správný soubor. Při stažení se k uživateli dostane výřez obrázku tak, jak ho vidí v prohlížeči (tj. dle přiblížení, které si uživatel zvolil).



Obrázek 3.7: Příklad v patičce stránky zobrazených informací o konkrétním obsahu

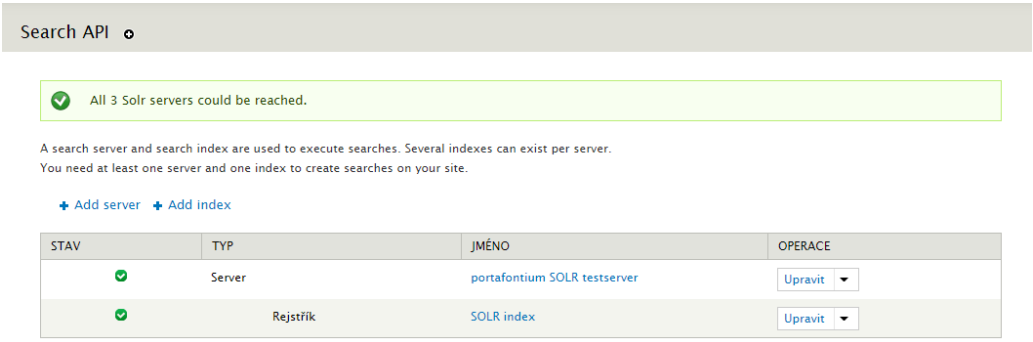
IIP Image Register

Jde o poslední modul, obsahující pouze funkce pro kontrolu platné adresy a návrat značky typu matriky. Aktuální seznam značek, které jsou vraceny, je následující:

- oo matrika oddaných
- * matrika narozených
- + matrika zemřelých
- ooi index oddaných
- *i index narozených
- +i index zemřelých

3.3.5 Search API

Pro redakční systém Drupal ve verzi 7 je *Search API* spíše než samostatný modul kolekcí modulů, rozšiřujících možnosti vyhledávání. V rámci instalace jsou nainstalovány moduly *Search Facets* a *Search Views*. Oba tyto moduly jsou ve verzi pro Drupal 8 zakomponovány do jednoho modulu. *Search API* jako takové je framework, umožňující jednoduché vytváření vyhledávání nad mnoha vyhledávacími enginy. Modul je napsán tak, aby bylo možné skrze tento modul jednoduše rozšířit možnosti vyhledávání a zobrazování výsledků a bylo možné jej použít pro jakýkoliv backend server, zajišťující samotné vyhledávání výsledků. Protože modul obsahuje kolem třiceti různých háčků, nebudou v této práci tyto háčky konkrétně popsány, jako tomu bylo například v kapitole 3.3.4. Po instalaci modulu je kromě těchto háčků, které mohou vývojáři využít pro vytváření vlastních modulů (ať už pro implementaci backendu pro vyhledávání nebo pro úpravy uživatelského rozhraní, jako je například automatické dokončování slov nebo řazení výsledků) [18] dostupná také konfigurace, ve které se konfigurují vyhledávací servery a indexy (obrázek 3.8).



Obrázek 3.8: Nastavení modulu *Search API*

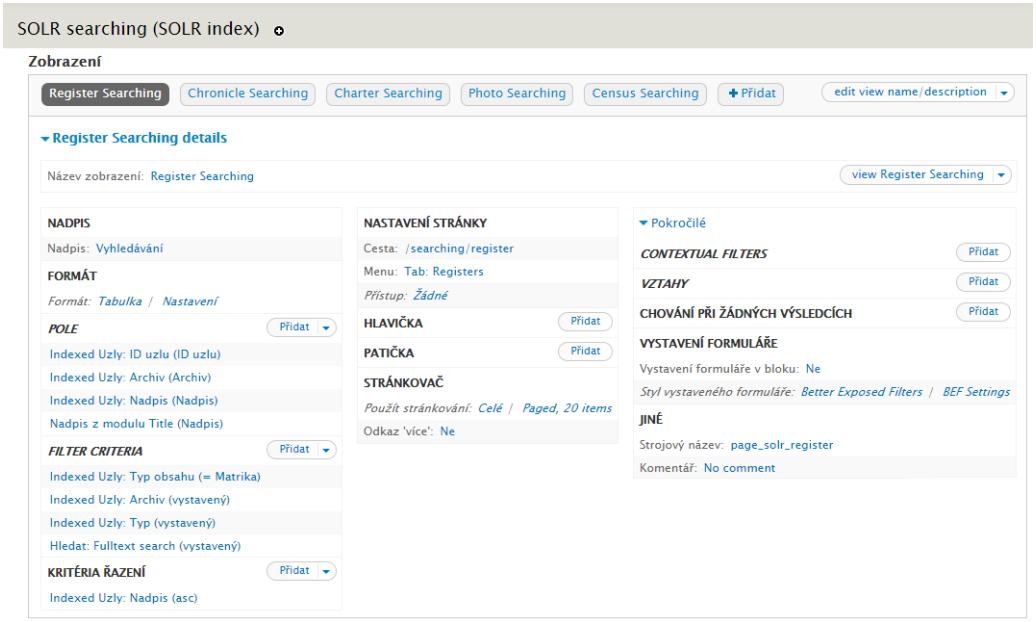
Search Facets

Tento modul poskytuje podporu pro vyhledávání za pomoci takzvaných *Faceted search* [22], což je typ navigace rozšiřuje vyhledávání o jednodušší a intuitivnější možnosti filtrování obsahu. Klasické filtry omezí obsah a zobrazí uživateli omezený počet záznamů. Nad tímto omezeným obsahem poté *Facets* umožňují zobrazit kontextový filtr, který nabídne omezení na vlastnosti, které má již zobrazený omezený obsah. Například při vyhledávání mobilního telefonu si může uživatel pomocí klasického filtru vybrat, že chce pouze telefony, které stojí více než deset tisíc korun. Nad tímto omezeným počtem

výsledných záznamů se poté může zobrazit nový *Facets* filtr, zobrazující například všechny velikosti displejů, které jsou ve vyhledaných mobilních telefonech na výběr. Tento modul je sice v redakčním systému Porta fontium nainstalován a povolen, ale není v žádném vyhledávání používán a proto zde nebude více popisován.

Search Views

Posledním modulem, instalovaným s modulem *Search API*, je *Search Views*. Tento modul využívá modul *Views*, který mimo jiné umožňuje vytváření nových stránek, zobrazujících přidávaný obsah nebo úpravu hlavní stránky tak, aby bylo možné řadit obsah, na ní zobrazený, jinak, než podle data přidání. Modul *Search Views* přidává do systému možnost v uživatelském rozhraní vytvářet stránky pro vyhledávání obsahu (obrázek 3.9).



Obrázek 3.9: Ukázka vytváření stránek pro vyhledávání skrze modul *Search Views*

V horní části stránky lze najít seznam již vytvořených stránek. V tomto konkrétním případě jsou vytvořeny stránky pro vyhledávání matrik, kronik a tak dále. Každá z daných stránek obsahuje definici polí, která se mají zobrazit a definici filtrů, pomocí kterých je možné vyhledávat. Seznam polí, která jsou ve stránce zobrazena, je definován v levé části stránky, v sekci *Pole*. Tato pole nejčastěji korespondují s poli, která jsou přímo na konkrétním nalezeném obsahu (například zobrazení nadpisu vyhledaného obsahu). Lze

ovšem nastavit například pole obsahující vlastní PHP kód, vlastní text či relevanci výsledku. Možnosti se liší podle nainstalovaných modulů, například poslední zmíněná možnost je přidána modulem *Solr search*, který je popsán v kapitole 3.3.6. Přidaná pole mají nastavitelné vlastnosti, lišící se podle typu pole. Pole je například možné omezit na určitý počet znaků nebo přepsat jeho hodnotu. Nastavená pole také nemusí být nutně uživateli portálu zobrazena. Jednou z možností je vložení některých polí, ale jejich skrytí před uživatelem, protože pro uživatele nejsou důležitá. Tato skrytá pole mohou být dále použita pro skládání více načtených polí do jednoho zobrazeného. U portálu Porta fontium je toto využito u odkazů na obrázky z kronik a dalšího obsahu. Kronika může obsahovat odkaz na externí obrázek, nebo odkaz na interní obrázek (uložený na serveru portálu Porta fontium). Každý odkaz je uložen ve vlastním poli kroniky (tím je rozlišeno, zda je nebo není externí) a ve výsledcích vyhledávání se zobrazuje sloučení těchto dvou polí do jednoho pole. Navíc se zde odkaz na obrázek obalí tagem *<a>*, díky čemuž je z něj vytvořen HTML odkaz, na kterém si uživatel může obrázek prohlédnout.

Další sekcí je *Filter Criteria*. V této sekci se definují filtry, pomoci kterých má uživatel portálu možnost omezit vyhledávaný obsah. Nejčastěji se zde opět nachází filtry korespondující přímo s poli obsahu. Případně například fulltextové vyhledávání skrze více těchto polí. U zvolený filtrů je pak možné nastavit další vlastnosti. Lze například vložit nový filtr, který bude obsahovat výčet hodnot, ze kterých uživatel vybere vždy jednu (například typ článku) a filtr předá do vyhledávání informaci o tom, co uživatel vybral. Je možné určit, zda je konkrétní filtr povinný a zda se má uživatelem zadaná hodnota pamatovat při příštím otevření stránky. Opět lze také filtry případně skrýt, aby s nimi uživatel nemohl manipulovat.

Poslední sekcí v levé části jsou *Kritéria řazení*. Zde administrátor určuje, jak se budou vyhledané výsledky řadit a zda s řazením může uživatel manipulovat.

V prostřední a pravé části stránky je možné nastavit další vlastnosti. Může zde být hlavička či patička, zobrazující například pevně daný text nebo informace o celkovém počtu nalezených záznamů. Další konfigurovatelnou volbou je definice stránkování (kolik výsledků bude zobrazeno na stránce nebo maximální počet vrácených výsledků). V pravé části jsou pokročilé možnosti konfigurace, které nejsou v portálu Porta fontium používány, a proto nebudou dále popisovány.

3.3.6 Solr search

Modul *Solr search* poskytuje podporu pro vyhledávání s využitím platformy Apache Solr. Instalací modulu je v nastavení modulu *Search API* zpřístupněna možnost přidat nový Solr server (obrázek 3.8). V rámci konfigurace se nastavují informace o serveru jako je adresa nebo přístupové údaje, aby mohl modul *Search API* volat korektní vyhledávací dotazy. Součástí modulů je také sada háčků, které lze využít v dalších modulech. Poskytnuty jsou háčky například pro úpravu obsahu před odesláním k zaindexování, pro úpravu vyhledávacího dotazu před odesláním na server nebo pro úpravu výsledků vyhledávání před zobrazením uživateli.

4 Modul podobných slov

První modul, který bude integrován do webového portálu Porta fontium, je modul podobných slov. Tento modul je reprezentován externím API, které je detailněji popsáno v další podkapitole. Cílem modulu podobných slov je získání podobných slov (synonym) z tohoto API a přidání těchto slov k vyhledávacímu dotazu, díky čemuž uživatel portálu získá relevantnější výsledky. Uživatel musí mít možnost některá slova z hledání odstranit, pokud je vyhledávat nechce nebo pokud nedávají smysl. Nejdůležitější částí integrace tohoto modulu bylo zobrazení podobných slov uživateli. Zde bylo možné vybrat z několika variant implementace této funkcionality, rozdíly mezi těmito variantami jsou popsány v dalších podkapitolách.

4.1 Dodané API

V době vývoje modulu bylo API dostupné na adrese *147.228.127.98:8089* a nabízelo čtyři dostupné operace. Všechny tyto operace očekávají volání metodou POST.

4.1.1 Operace *detectLanguage*

Operace *detectLanguage* umožňuje rozpoznat jazyk vstupního řetězece. Podporované jazyky jsou němčina a čeština.

4.1.2 Operace *nearestQuery*

Tato operace vrací pro vstupní řetězec (vyhledávací dotaz) dotaz, rozšířený o podobná slova. Například pokud je operace zavolána pro vyhledávací dotaz "král koruna česká", je výsledkem vyhledávací dotaz "panovník Král Koruna německá slovenská".

4.1.3 Operace *nearestWords*

Nejdůležitější operace pro realizaci modulu podobných slov je operace *nearestWords*. Vstup pro tuto operaci je:

- vybrané slovo
- jazyk, ze kterého slovo pochází

- jazyk, ve kterém se mají nalézt podobná slova

Podporované jazyky jsou znovu němčina a čeština. Operace vrací vždy deset nejpodobnější slov a jejich podobnost (ta je definovaná desetinným číslem v intervalu (0,1), kde 1 je maximální podobnost). Slova jsou seřazena od nejpodobnějšího k nejméně podobnému.

4.1.4 Operace *search*

Poslední dostupnou operací je operace *search*. Tato operace vyhledává v databázi portálu Porta fontium. Vstupem je pouze vyhledávací dotaz a počet výsledků, které se mají vrátit.

4.2 Již existující moduly

Před zahájením práce na vytvoření modulu byl proveden průzkum, zda neexistuje modul, který by dokázal zamýšlenou funkcionalitu poskytnout. Bohužel neexistuje modul, který by nabídl alespoň podobnou funkcionalitu. Existovala ovšem podobná řešení využívající taxonomii [16]. Redakční systém Drupal nabízí modul jádra, který umožňuje třídit obsah do různých kategorií. Pro název kategorie může být v systému vytvořeno synonymum (případně více synonym), dle kterého lze poté nalézt konkrétní obsah. Toto řešení ovšem nesplňuje původní požadavek, kterým je získávání slov on-line pro konkrétní hledaný výraz.

4.3 Integrace podobných slov do formuláře

Hlavní část integrace tohoto modulu byla analýza a výběr vhodného řešení, jak uživateli webového portálu zobrazit podobná slova a umožnit mu jejich výběr. V dalších kapitolách jsou popsány tři varianty, které byly v rámci práce analyzovány. Pro realizaci byla nakonec zvolena varianta z kapitoly 4.3.3, která zobrazení slov ve formuláři provádí pomocí dostupných háčků.

4.3.1 Překreslení již existujícího pole

První variantou bylo využití existující uživatelské konfigurace. Protože modul *Search API Solr* nabízí možnost uživatelsky vytvořit formulář a definovat v něm filtrovací kritéria, nabízelo se jako možné řešení vložit nové filtrovací kritérium pomocí uživatelského rozhraní. Mezi dostupnými filtrovacími kritérii se dokonce nabízelo zvolit možnost *Fulltext search*. Toto kritérium

umožňuje vybrat pole, ve kterých se má fulltextově vyhledávat. Během konfigurace je možné zvolit také formát zobrazení, kde jedním z formátů jsou zaškrtávací políčka. Modul by využil systémem vykreslený prvek a pouze by dle aktuální potřeby nahradil zobrazené volby. Takovéto řešení se zdálo jako nejlepší volba a proto bylo implementováno jako první. Bohužel v redakčním systém Drupal ve verzi 7, kterou portál Porta fontium využívá, se vyskytuje chyba znemožňující toto řešení využít. Tato chyba se projeví v případě, že se uživatel rozhodne odškrtnout všechna podobná slova a spustí znovu hledání. V danou chvíli se hledání nepovede a portál zobrazí chybu v jednom ze souborů jádra. K této chybě dochází i v případě, že je filtr nastaven jako nepovinný.

4.3.2 Naprogramování nového formuláře

Druhou zvažovanou variantou bylo přepsání celého formuláře do kódu modulu. Přínosem této možnosti by byla úplná kontrola nad vzhledem i funkcionalitou. Velkou nevýhodou by ovšem byla potřeba přepsat každý formulář, ve kterém by se měl modul a zobrazování podobných slov používat. Stejně tak by takové řešení znemožnilo budoucí změny ve formuláři administrátorem portálu bez toho, aby musel měnit samotný kód modulu.

4.3.3 Doplnění podobných slov kódem

Poslední analyzovanou možností bylo zobrazení podobných slov za pomoci implementace háčků, umožňujících formulář před vykreslením upravit a které jsou poskytnuty redakčním systémem a nainstalovanými moduly. Tento případ by nevyžadoval přepisování celého formuláře do zdrojového kódu. Stejně tak by nebylo nutné upravovat formulář v uživatelském rozhraní a přidávat do něj další filtrovací kritéria. Aby bylo možné tuto variantu využít, bylo by potřeba implementovat minimálně dva háčky. První z nich by sloužil k získání podobných slov a upravil by formulář tak, aby byla podobná slova zobrazena uživateli. Druhý háček by zajistil úpravu samotného Solr dotazu. Oba tyto háčky by měly být dostupné. Největší výhodou této varianty je možnost přidat požadovanou funkcionalitu bez toho, aby bylo nutné jakkoliv měnit již existující formulář a v případě, že by se modul měl přestat používat, stačilo by ho vypnut a tím by došlo ke zrušení veškeré funkcionality.

4.4 Struktura modulu

Adresář modulu má následující strukturu (adresáře jsou v textu zobrazeny tučně a samotné soubory kurzívou):

- **search_api_solr_similar_words**
  - **tests**

    Adresář obsahující pomocný modul sloužící k testování modulu
    - *search_api_solr_similar_words_test.info*

      Soubor s informacemi o pomocném modulu
    - *search_api_solr_similar_words_test.module*

      Soubor obsahující kód pomocného modulu
  - **translation**

    Adresář obsahující soubory pro překlad modulu
    - *search_api_solr_similar_words.pot*

      Soubor obsahující texty modulu, slouží jako vzor pro budoucí překlad
    - *cs_CZ.po*

      Soubor s překladem všech textů modulu do češtiny
  - *search_api_solr_similar_words.info*

    Soubor s informacemi o modulu
  - *search_api_solr_similar_words.module*

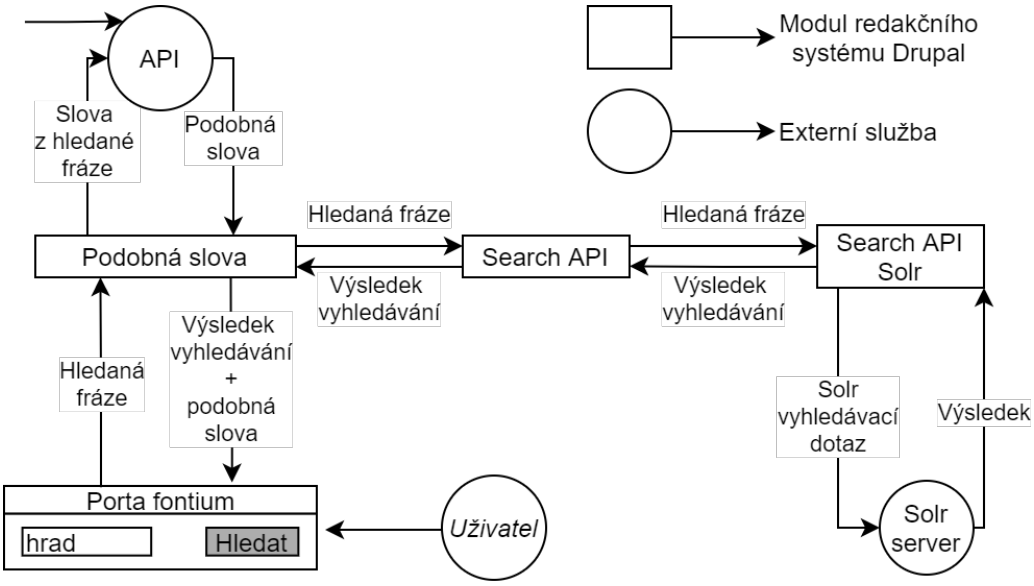
    Soubor obsahující kód modulu
  - *search_api_solr_similar_words.test*

    Soubor obsahující testy modulu

4.5 Integrace modulu

V rámci integrace vznikl v redakčním systému nový modul *Search API Similar Words*. Komunikace modulu s dalšími moduly redakčního systému je znázorněna na obrázku 4.1. Tento modul pro instalaci vyžaduje modul *Solr search* a byl vytvořen pro Drupal 7. V seznamu modulů je zařazen do sekce *Hledat*. Modul obsahuje vlastní konfigurační stránku, kde lze nastavit základní parametry. Mezi tyto parametry patří například nastavení maximálního počtu podobných slov. Protože dostupné API vrací maximálně deset

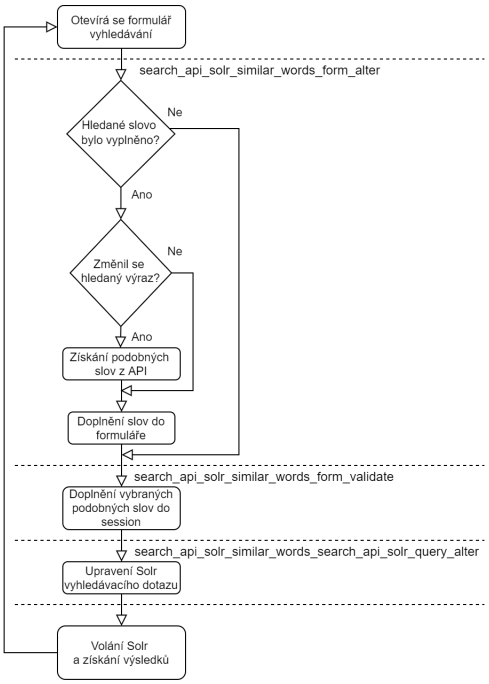
podobných slov pro jedno slovo, je takto nastavena maximální možná hodnota pro zadání. Pokud je parametr nastaven na nulu, nepokouší se modul vůbec volat API a získávat tak podobná slova.



Obrázek 4.1: Diagram komunikace modulu podobných slov s dalšími moduly redakčního systému

Druhým parametrem je adresa API, kterou je možné změnit v případě, že by se API přesunulo na novou adresu. Dále existuje parametr určující typ výběru podobných slov. Na výběr jsou volby *Nejpodobnější* a *Poměrem*. Rozdíl mezi těmito volbami je popsán dále v kapitole. Posledním parametrem je výběr formulářů a filtrů v těchto formulářích, které se mají používat. Výběr je určen textově, kde se do pole v konfiguračním formuláři zadává dvojce *id_formulare,id_filtru*. Při otevření vyhledávacího formuláře je kontrolováno, zda je ID zobrazeného formuláře zadáno v tomto nastavení. Pokud není, nedochází v daném formulář k vyhledávání podobných slov. Druhá část, ID filtru, určuje pole, pro jehož vstup se podobná slova vyhledávají. Aby bylo možné dohledat tato dvě ID, je v konfiguračním formuláři dostupná volba *Zobrazit ID formuláře ve webových formulářích*. Po jejím zaškrtnutí se ve všech formulářích webového portálu začne na stránce zobrazovat ID daného formulář. Vypsání je realizováno triviálně a není nijak zakomponováno do vzhledu formulářů. Nepředpokládá se, že by tato možnost byla používaná v ostrém provozu a je zde pouze pro jednodušší dohledání ID formulář při prvním nasazení. ID filtru je možné nalézt v nastavení vyhledávacího formuláře (kapitola 3.3.5) v nastavení konkrétního filtru, pro který má funk-

cionalita podobných slov používána.



Obrázek 4.2: Funkce modulu podobných slov

Na obrázku 4.2 je zobrazen dále popisovaný průběh implementované funkcionality. Prvním krokem pro zobrazení podobných slov byla úprava formuláře. K tomu je využit háček *hook_form_alter*. Implementací tohoto háčku je upraven zobrazovaný formulář. V rámci háčku je důležité nejdříve smazání proměnné, která v session obsahuje vybraná podobná slova (tato proměnná je plněna v rámci validace, která je popsána v dalších odstavcích). Toto se provádí pro všechny formuláře redakčního systému. Pokud by ke smazání nedošlo, mohla by nastat situace, že se podobná slova vykreslí ve formuláři, ve kterém se vyskytovat vůbec nemají.

Poté je zkontrolováno, zda je zobrazovaný formulář zahrnut v seznamu formulářů, ve kterých se mají podobná slova zobrazovat. Pokud tomu tak není, nic dalšího se v rámci modulu neděje. V opačném případě modul provede kontrolu uživatelem zadaného vyhledávacího dotazu. Zkontroluje se, zda vybraný filtr (filtr, který je v nastavení modulu určen jako vstup pro získání podobných slov) není prázdný. Pokud by prázdný byl, není důvod hledat podobná slova. Modul také kontroluje, zda aktuálně hledaný výraz není shodný s výrazem uloženým v session. Cílem této kontroly je nevolat získávání podobných slov pokaždé, kdy je formulář odeslán. Může se totiž stát, že uživatel pouze změnil výběr podobných slov, ale hledaný výraz

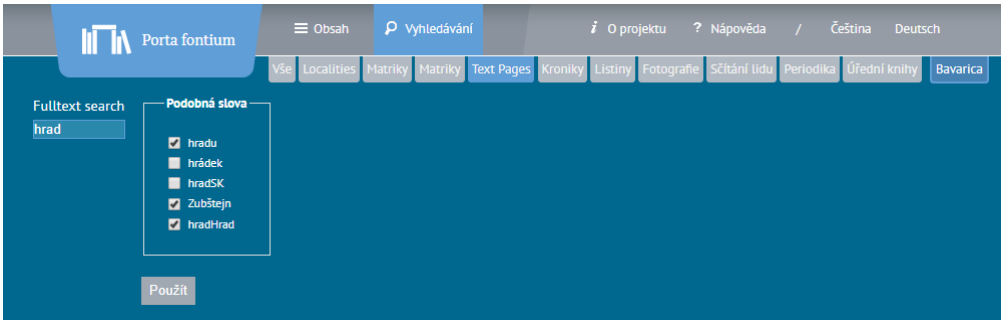
ponechal. Případně uživatel mohl pouze doplnit další filtrovací kritéria pro vyhledávání. Stalo-li se, že se hledaný výraz změnil, jsou vyhledána podobná slova.

Pro získání podobných slov je voláno externí API a jeho operace *nearestWords*. Volání je provedeno funkcí *drupal_http_request*, která je dostupná v rámci funkcí jádra redakčního systému. Požadavek na API je zavolán tolikrát, kolik je uživatelem hledaných slov a frází. Navíc, protože API podporuje podobná slova ve dvou jazycích a není možné rozpoznat, zda je volaný výraz česky či německy, je toto volání na API nutné volat zároveň pro oba jazyky. Jako příklad lze uvést, že pokud uživatel hledá `hrad Plzeň`, je API zavoláno celkem čtyřikrát. Dvakrát pro každé slovo (jednou se předpokládá, že je slovo česky a podruhé, že je německy). Předpokládá se, že běžný uživatel nebude hledat více než tři slova, a proto by ho toto volání nemělo výrazně zdržovat. Výsledkem těchto volání je pole, obsahující hledané výrazy a ke každému z nich dvacet podobných slov (deset pro češtinu a deset pro němčinu). Volané API vrací ke každému podobnému slovu i jeho podobnost (definovanou jako desetinné číslo v intervalu (0,1), kde 1 je maximální podobnost). Podobná slova pro každý vstup jsou seřazena dle této podobnosti a je vybráno pouze N prvních nejpodobnějších (počet lze definovat v nastavení modulu). Tato slova jsou uložena do session. Například při hledání podobných slov pro slovo `hrad` by vrácená podobná slova vypadala následovně:

```
[hrad] => Array
      (
        [0] => stdClass Object
          (
            [word] => hradu
            [similarity] => 0.75869757
          )
        [1] => stdClass Object
          (
            [word] => hrádek
            [similarity] => 0.68009233
          )
        ...
      )
```

Po získání podobných slov musí být vybrána slova, která se uživateli zobrazí. Maximální počet zobrazených podobných slov je určen nastavením modulu. Modul nyní podporuje dvě možnosti výběru zobrazených podobných

slov. První možností je vybrat ta nejpodobnější. Při zvolení této možnosti jsou všechna slova (v příkladu uvedeném výše by po zavolání API modul získal dvacet podobných slov a pro zpracování jich předal deset) seřazena dle jejich podobnosti a jsou zobrazena pouze ta nejpodobnější. Může tedy nastat situace, kdy pro některé z hledaných slov nebude zobrazeno žádné jemu podobné, protože nebude mít dost vysokou podobnost, aby se dostalo mezi několik prvních vybraných. Druhá volba vybírá podobná slova tak, aby každé hledané slovo mělo stejný počet zobrazených podobných slov ve formuláři. V tomto případě je spočten počet slov pro každý výraz a poté je tento počet nejpodobnějších slov pro každý výraz vybrán a zobrazen. V případě hledání dvou výrazů je tedy pro každý výraz zobrazeno pět nejpodobnějších slov (obrázek 4.3).



Obrázek 4.3: Podobná slova zobrazená ve formuláři

K vykreslovanému formuláři je kromě podobných slov přidána také vlastní validace. Tato validace je vytvořena jako *callback*, to znamená, že je volána pouze pro určené formuláře. Validace se na formulář přidá pouze pokud jsou splněny následující dvě podmínky:

- Pro tento formulář se mají vyhledávat podobná slova (určeno v nastavení modulu)
- Formulář obsahuje filtr, pro který se mají podobná slova vyhledávat

Validační háček provádí uložení slov, která uživatel vybral. Během provádění tohoto háčku jsou již k dispozici hodnoty zadané uživatelem. Slova, která tedy zůstala ve formuláři označená, jsou uložena do session, aby s nimi bylo možné dále pracovat při úpravě vyhledávání.

Poslední důležitou součástí je úprava vyhledávání. K tomu je využito háčku *hook_search_api_solr_query_alter*. Tento háček je poskytnut modulem *Solr search* a je volán před odesláním vyhledávacího dotazu. Aby nebyl

dotaz upravován pro formuláře, které nevyužívají funkcionalitu podobných slov, je zkontrolováno vyplnění proměnné v session, obsahující vybraná slova.

Tato proměnná je při vykreslení každého formuláře smazána a znovu vyplněna při jeho validaci. Pokud tedy proměnná existuje, může být dotaz upraven. Protože pole, které je využito pro vyhledávání podobných slov může být použito jak v hlavním vyhledávacím dotazu, tak ve filtrovacím dotazu (*filter query*), bylo nutné ošetřit oba případy. V prvním případě, je uživatelem zadaný vyhledávaný výraz přímo součástí vyhledávajícího dotazu. Modul tedy provede porovnání obsahu vyhledávacího dotazu a posledního hledaného výrazu, který je uložen v session. Jsou-li shodné, rozšíří se vyhledávací dotaz o podobná slova. Pokud by bylo použito vyhledávání z obrázku 4.3 a fulltextové pole by bylo používáno ve vyhledávacím dotazu, byl by rozdíl v dotazech následující:

`"hrad"` $\longrightarrow$ `("hrad" OR "hradu" OR "Zubštejn" OR "hradHrad")`

Druhou variantou je, že je pole využíváno jako filtrovací vyhledávací dotaz. V tomto případě je parametr *fq* (filter query) vyplněn poli, ve kterých se má zadaný výraz vyhledávat. Například pro filtrování polí *tělo* a *poznámka* vypadá hodnota parametru následovně:

```
((tm_body$value:(hrad)) OR (tm_field_note:(hrad)))
```

Rozšíření této části je tedy o něco komplikovanější. Filtrovacích dotazů může být více. Je tedy nejdříve nalezen ten správný. Bohužel součástí objektu nesoucího informace o hledání není informace o tom, z jakého vyhledávacího pole ve formuláři je filtr vytvořen. Dohledání filtru, který bude rozšířen o podobná slova, tedy probíhá pomocí vyhledání hledaného výrazu (ten je uložen v session) a kontroly, že se v rámci filtru vyhledává v poli *tělo*. Pokud je tomu tak, je filtr nahrazen novým filtrem, rozšířeným o podobná slova. Zde je ošetřeno, že se vyhledává ve stejných polích jako vyhledával původní nerozšířený filtr.

4.6 Překlad modulu

Redakční systém Drupal ve verzi 7 obsahuje modul *Locale* poskytující takzvané *Localization API*, které vývojářům nabízí funkce pro usnadnění vývoje vícejazyčných modulů. Obecným pravidlem při vývoji modulů a vzhledů je psaní textů v anglickém jazyce [8], z tohoto důvodu modul vytvořený v rámci této práce obsahuje anglické texty. Tyto texty jsou používány vždy v kombinaci s funkcí *t()*, která informuje redakční systém o tom, že daný text je zobrazen uživateli a měl by být zobrazen překlad textu ve správném jazyce (tj. jazyce, ve kterém je konkrétní instalace redakčního systému, případně,

pokud jde o vícejazyčný portál, v jazyce, který má nastavený uživatel). Díky využití této funkce je možné některé moduly nepřekládat ručně, a namísto toho využít databázi překladů, kterou udržují vývojáři redakčního sytému. Databáze obsahuje mnoho jazyků a dostupné překlady se v případě shody textů aplikují i na texty vlastních modulů, které nemají vlastní překlad. Redakční systém totiž každý text, který je zobrazován funkcí *t()*, zkontroluje oproti uloženým překladům a v případě, že není pro tento text vlastní překlad, je aplikován překlad z databáze.

Kromě toho je možné v uživatelském rozhraní redakčního systému najít informaci o celkovém stavu překladu konkrétní instalace redakčního systému (obrázek 4.4). K této informaci zde lze také nalézt seznam všech nepřeložených textů, včetně možnosti doplnit k nim překlad ručně. Všechny řetězce mohou být vyexportovány ve formátech *.po* (obsahuje původní texty i jejich překlady) a *.pot* (obsahuje pouze původní texty).

JAZYK	VESTAVĚNÉ ROZHRANÍ
Česky	4744/8543 (55.53%)
Angličtina (vestavěný jazyk)	není

Obrázek 4.4: Ukázka procenta přeložených textů pro český jazyk

Webový portál Porta fontium je provozován ve dvou jazycích, češtině a němčině. Protože většina textů použitých při vývoji nebyla obecného charakteru, ale šlo o texty specifické pro vyvíjený modul, bylo nutné vytvořit překlady alespoň pro český jazyk. Standardní funkce redakčního systému umožňují exportovat pouze všechny dostupné řetězce, což nebylo vhodné pro překlad textů jednoho vytvářeného modulu. Obdobně tomu bylo u varianty, kdy by došlo k překladu textů pomocí uživatelského rozhraní. Tato varianta by neumožnila jednoduchý převod překladů z testovacího prostředí do produkčního.

Po provedeném průzkum byl tedy do redakčního systému nainstalován nový modul *Translation template extractor*, umožňující export textů z konkrétních nainstalovaných modulů nebo vzhledů. Za pomoci tohoto modulu byly vyexportovány všechny texty z nově vytvořeného modulu do souboru *.pot*. Tento soubor posloužil jako vzor pro překlad textů do českého jazyka a je možné ho využít jako vzor pro překlady do dalších jazyků. Součástí modulu je tedy adresář *translation*, ve kterém se nachází jak zmíněný vzorový soubor *.pot*, tak soubor *cs_CZ.po* obsahující kompletní překlad modulu do

češtiny.

Po instalaci modulu lze vytvořené překlady aplikovat pomocí uživatelského rozhraní, konkrétně v obrazovce dostupné v menu Nastavení -> Regionální a jazyková nastavení -> Překlad rozhraní -> Import.

5 Modul OCR

Druhým modulem, který měl být v rámci této práce integrován do redakčního systému Drupal portálu Porta fontium, byl modul, který provádí OCR nad již existujícími obrazovými dokumenty z obsahu, který je již zadán v redakčním systému. Z těchto obrazových dokumentů je nutné vytvořit pomocí OCR nový obsah, ve kterém bude možné fulltextové vyhledávání a z výsledků tohoto vyhledávání bude poté možné dohledat původní obrazové dokumenty. Tento modulu by měl umožnit výměnu metody OCR tak, aby nebylo v budoucnosti potřeba přepisovat kompletně celý modul.

5.1 Dodaná funkcionalita

Na začátku integrace modulu se OCR bude provádět pomocí enginu *Tesseract* [14]. *Tesseract* je open source engine dostupný pod licencí Apache 2.0. Je možné jej používat přímo jako spustitelnou aplikaci (jak v operačním systému Windows, tak v Linuxu) nebo využít dostupné API k vytvoření vlastní OCR aplikace.

5.2 Již existující moduly

V průběhu vytváření práce byly nalezeny dva moduly poskytující funkcionalitu OCR pro redakční systém Drupal. Oba moduly využívají pro rozpoznání textu engine *Tesseract*.

První z modulů se jmenuje *Webform OCR*. Bohužel tento modul poskytuje funkcionalitu pro rozpoznání textu pouze pro vytváření webových formulářů. Tento modul tedy nemá smysl využívat pro implementaci požadované funkcionality.

Druhý nalezený modul, *OCR for Drupal*, se zdál být pro požadované použití mnohem vhodnější. Modul umožňuje vytváření obsahu načtením obrazových dokumentů. Po instalaci může administrátor vybrat, jak se bude generovat název vznikajícího obsahu. Na výběr je možnost generování dle názvu obrazového dokumentu, nebo ruční generování. Ruční generování v tomto případě znamená, že administrátor poskytne text, který se nastaví jako název každému vzniklému obsahu a modul za tento text bude přidávat automaticky rostoucí číslo. Modul umožňuje importovat obrazové dokumenty interaktivně po jednom, a v tomto případě je administrátor po nahrání obrazového doku-

mentu přesměrován na stránku pro vytváření obsahu, kde je předvyplněno tělo dle rozpoznaného textu. Případně je možné importovat obrazové dokumenty dávkově s tím, že je vybrán adresář, ve kterém se všechny požadované obrazové dokumenty nachází. V tomto případě vzniká obsah rovnou a titulek se předvyplňuje dle nastavení. Tento modul by bylo možné využít pro implementaci, nebýt několika nedostatků. Prvním nedostatkem je nemožnost jednoduše vyměnit OCR engine. Pokud by měl být engine měněn, byl by nutný zásah do kódu modulu a to v případě, že jde o modul třetí strany není vhodné. Dalším nedostatkem je absence jakýchkoliv háčků. Při tvorbě obsahu dávkově (což je předpokládaný scénář) není možné obsah před vytvořením upravit a například doplnit vazbu na původní obsah nebo obrazový dokument, ze kterého nový obsah vznikl. Aby bylo těchto možností dosáhnuto, muselo by se opět zasahovat do kódu modulu.

Nakonec tedy ani jeden z nalezených modulů nebyl na základě analýzy vyhodnocen jako vhodná volba pro splnění požadavků zadání a nejlepší volbou bude vytvořit modul vlastní.

5.3 Struktura modulu

Adresář modulu má následující strukturu (adresář jsou v textu zobrazeny tučně a samotné soubory kurzívou):

- **custom_ocr**
  - **scripts**

    Adresář obsahující shellové skripty volané z modulu
    - *absolute_path.sh*

      Skript, který pro předanou cestu, vrátí absolutní cesty všech souborů (včetně souborů z podadresářů)
    - *tesseract.sh*

      Skript, který provede OCR pomocí Tesseract enginu
  - **tests**

    Adresář obsahující nový modul sloužící k testování modulu
    - *custom_ocr_test.info*

      Soubor s informacemi o pomocném modulu
    - *custom_ocr_test.module*

      Soubor obsahující kód pomocného modulu
  - **translation**

Adresář obsahující soubory pro překlad modulu

  * *custom_ocr.pot*

    Soubor obsahující texty modulu jako vzor pro budoucí překlad

  * *cs_CZ.po*

    Soubor s překladem všech textů modulu do češtiny

- *custom_ocr.info*

  Soubor s informacemi o modulu

- *custom_ocr.module*

  Soubor obsahující kód modulu

- *custom_ocr.test*

  Soubor obsahující testy modulu

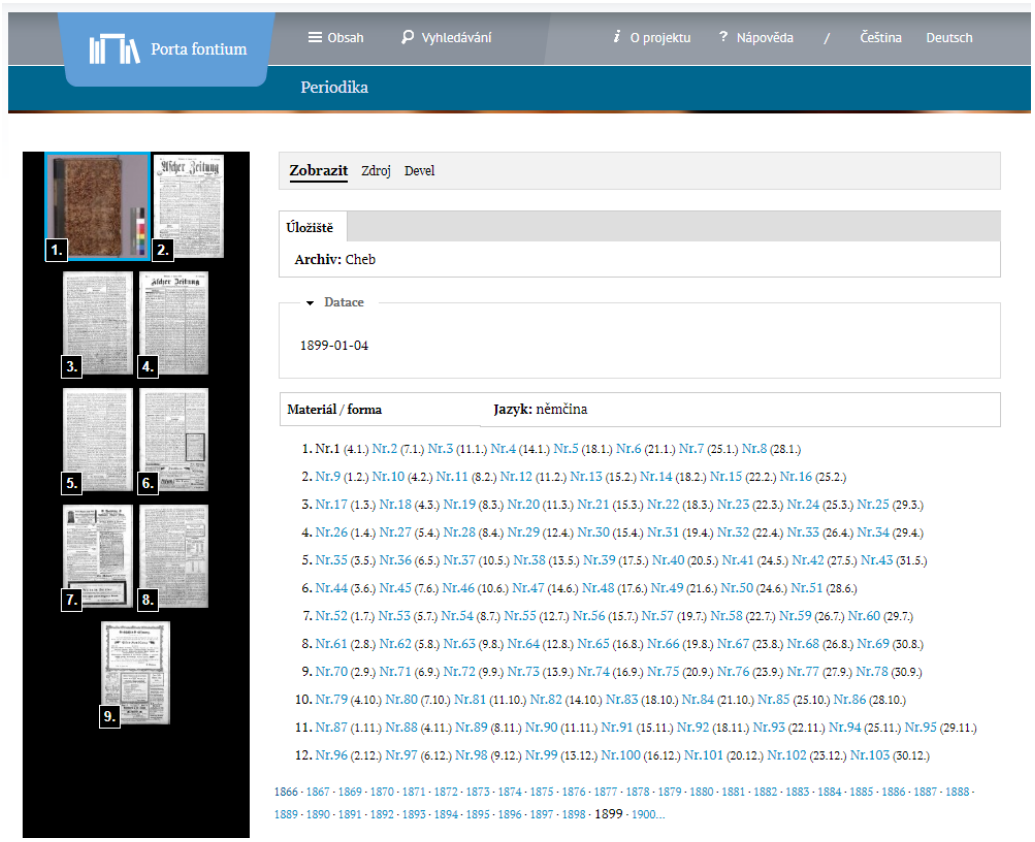
5.4 Zpracovávaný obsah a obrazové dokumenty

Typy obsahu, pro které je požadováno zpracování textu z obrazových dokumentů, jsou v redakčním systému pojmenovány *Periodikum* a *Periodikum-číslo*. Tyto typy obsahu reprezentují časopisy vydávané v minulosti na území Česka a Německa. Oba typy obsahu obsahují shodná pole a obecně i z pohledu zobrazení údajů se chovají totožně. Jediným vypozorovaným rozdílem je použití typů obsahu při vytváření nového obsahu. První číslo každého roku, kdy byl časopis vydáván, je vytvořeno jako typ obsahu *Periodikum*. Ostatní čísla v daném roce jsou následně vytvořena jako typ *Periodikum-číslo*. V dalším textu budou oba typy obsahu označovány shodně slovem *periodikum*, reprezentujícím jedno číslo časopisu.

Při zobrazení periodika v portále Porta fontium je zobrazen souhrn informací, jehož hlavní částí je blok, zobrazující obrazové dokumenty jednotlivých stránek (obrázek 5.1). Tyto obrazové dokumenty mají být vstupem pro rozpoznávání textu. V době vytváření modulu, nebylo v redakčním systému připraveno členění na jednotlivé stránky. Jednotlivá periodika obsahují pouze cestu do konkrétního adresáře, ve kterém jsou uloženy všechny stránky jednoho čísla. Tato cesta je uložena v poli s popiskem *Obr. data* (název v redakčním systému je *field_digi*). Cesta uložená v tomto poli je relativní cesta z adresáře */data/archiv/public*, který se nachází na totožném serveru jako redakční systém samotný.

Pro zpracovávání obrazových dokumentů připadaly v úvahu dva možné přístupy. Bylo možné zvolit cestu, kdy uživatel vybere konkrétní periodika

a pro tato vybraná periodika se dohledají všechny obrazové dokumenty ke zpracování, nebo uživatel zvolí konkrétní adresář a modul bude vyhledávat periodika dle zvoleného adresáře. Po důkladném zvážení kladů a záporů byla zvolena druhá cesta, jelikož poskytuje jednodušší kontrolu nad výběrem. Ačkoliv není možné vybrat například dva rozdílné roky, je možné takto



Obrázek 5.1: Obsah typu *Periodikum* v portále Porta fontium

jednoduše vybrat jedno číslo, celý rok nebo kompletně všechna periodika. V případě, že by výběr probíhal pouze výběrem periodik, nebylo by jednoduché zvolit více let najednou bez vytvoření speciálně upraveného formuláře, který by později nemusel být použitelný u případného dalšího typu obsahu.

5.4.1 Nový typ obsahu - *Stránka textu*

Jak bylo zmíněno v předchozí kapitole, v době psaní této práce byly obrazové dokumenty ke zpracování členěné do adresářové struktury dle jednotlivých čísel. Mimo tohoto členění jsou obrazové dokumenty rozděleny také dle jednotlivých let a dle oblasti, ve které vycházely. Ukázka struktury:

- periodical
  - soap-do
    - posel-od-cerchova-1872
      - 03-30-n1
      - 04-06-n2
    - posel-od-cerchova-1873

V ukázce je zobrazena struktura adresářů na serveru. Prvním adresářem je typ obsahu. Práce se zabývá zpracováním periodik, ale v budoucnosti je možné rozšíření i na zpracovávání dalších typů obsahu. Následuje adresář s oblastí, ve které periodikum vycházelo. Pod oblastí se nachází jednotlivé roky (v názvu adresáře je také název periodika) a posledními adresáři ve struktuře jsou jednotlivá čísla. Název adresáře s jednotlivými čísly je složen z data vydání a čísla periodika. V tomto adresáři se nacházejí obrazové dokumenty jednotlivých stránek a toto je také adresář, jehož cesta je uložena v poli *field_digi*, popisovaném v předchozí kapitole.

Aby bylo umožněno vyhledávání ve zpracovaných obrazových dokumentech, bylo nutné vytvořit nový typ obsahu, který bude obsahovat text získaný pomocí OCR. Vznikl tedy typ obsahu *Stránka textu*. Tento typ obsahu obsahuje znatelně méně polí, než ostatní typy obsahu, a to:

- Nadpis

  Název obsahu. Je složen z názvu čísla periodika a čísla, které je na konci názvu souboru obrazového dokumentu. Například *Posel od Čerchova 1873, č.51 (1080)*, kde *1080* je číslo z názvu obrazového dokumentu identifikující konkrétní stránku.

- Body

  Text obsahu. Do tohoto pole se ukládá text získaný z OCR a v tomto poli se poté bude vyhledávat.

- Titul

  Odkaz na konkrétní číslo. Toto pole odkazuje na konkrétní číslo časopisu, ze kterého stránka pochází.

- Obrázek (cesta)

Cesta k obrazovému dokumentu. Textový údaj obsahující relativní cestu z adresáře */data/archiv/public* do adresáře s obrazovými dokumenty konkrétního čísla časopisu. Například *periodical/soap-do/posel-od-cerchova-1873/12-20-n51*.

- Obrázek (soubor)

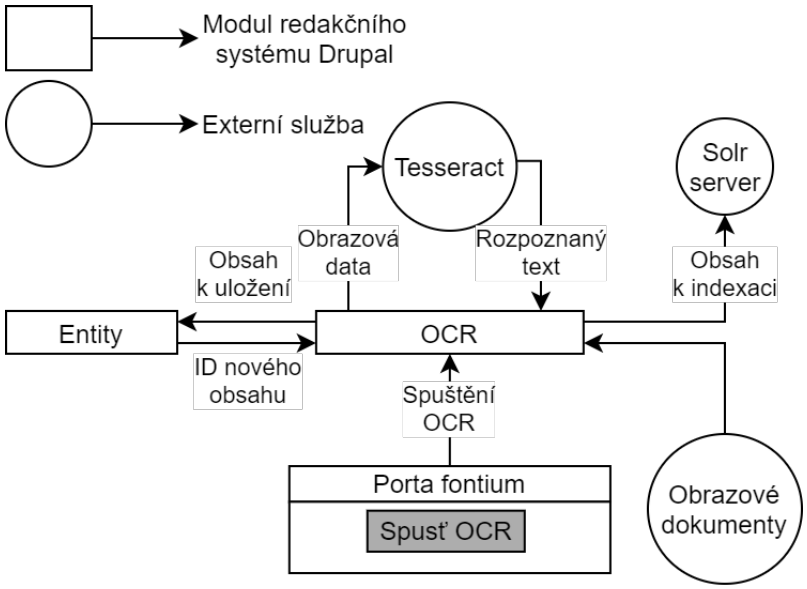
  Název souboru obrazového dokumentu. Obsahuje celý název souboru obrazového dokumentu (bez přípony), který se zpracovával. Například *posel-od-cerchova-1873-12-20-n51_1080*.

5.5 Integrace modulu

V rámci integrace nové funkcionality, kterou nebylo možné dodat použitím již existujících modulů, vznikl nový modul s názvem *Custom OCR*. Modul nevyžaduje instalaci žádných dalších modulů a je zařazen v sekci *Vytváření obsahu*. Komunikace modulu s dalšími moduly redakčního systému je znázorněna na obrázku 5.2. V rámci konfigurace (dostupné z menu Nastavení -> Vytváření obsahu -> OCR) je možné nastavit adresář, ze kterého se budou načítat obrazové dokumenty pro rozpoznávání textu. Dále se zde vybírá jazyk obrazových dokumentů. Jazyk je důležitý pro korektní rozpoznávání textů. Výběr jazyka se předává dále a je dle něj vybrán správný model pro OCR engine. Administrátor má také možnost nastavit počet obrazových dokumentů pro zpracování v jedné dávce.

Posledními nastaveními jsou dvě zaškrtávací volby. První rozhoduje o tom, zda se bude aktualizovat již existující obsah, tedy zda má smysl provádět rozpoznávání textu i pro obrazové dokumenty, ke kterým už je textový obsah vytvořen a text u těchto obrazových dokumentů aktualizovat. Toto nastavení je vhodné využít v případech, kdy se například změní OCR engine a je potřeba aktualizovat texty bez toho, aby musel být všechen již dříve vytvořený obsah smazán. Druhá volba zajistí automatické zaindexování nového či aktualizovaného obsahu. Ve spodní části formuláře se poté nachází tlačítko *Spustit OCR*, které spustí rozpoznání textu.

Po stisku tohoto tlačítka je zavolána funkce *run_ocr_button_callback*. Tato funkce přečte nastavení a připraví dávkovou úlohou (kapitola 2.3) pro spuštění. V rámci přípravy je načten seznam všech souborů z adresáře zadaného v nastavení modulu. Seznam absolutních cest je získán zavoláním skriptu *absolute_path.sh*, který vrátí absolutní cesty pro všechny soubory (včetně souborů v podadresářích) s příponou **.jp2*. Z tohoto seznamu a dalších nastavení modulu je vytvořena dávková úloha, která je automaticky



Obrázek 5.2: Diagram propojení modulu OCR s dalšími moduly redakčního systému

spuštěna. Dávková úloha prochází absolutní cesty k obrazovým dokumentům a každý obrazový dokument postupně zpracovává.

Prvním krokem je odstranění názvu souboru z absolutní cesty. Cesta bez názvu souboru je použita pro dohledání obsahu, ke kterému obrazový dokument patří. Typicky se jedná o periodika, která mají k jednomu číslu několik stránek. Číslo periodika (v portále Porta fontium jde o samostatný typ obsahu) se tedy dohledá dle této cesty. Pokud obsah není nalezen, je do logu redakčního systému zapsán záznam o chybě. Je-li obsah nalezen, záleží další chování na nastavení aktualizace již rozpoznaných textů. Povolil-li administrátor aktualizaci, je nyní provedeno OCR, tzn. je spuštěn skript *tesseract.sh*. Skript spustí engine Tesseract a výsledný text, získaný z obrazového dokumentu, uloží do dočasného adresáře (v Porta fontium se jedná o adresář */tmp*). Pokud aktualizace existujícího obsahu povolena nebyla, je tento krok přeskočen a modul pokračuje kontrolou, zda zpracovávaný obrazový dokument nebyl již v minulosti zpracován, a tudíž k němu neexistuje odpovídající obsah. Skončí-li kontrola negativně a jde tedy o nový obsah, je v případě potřeby provedeno OCR (pokud bylo provedeno v předchozím kroku, je použit výsledek z tohoto kroku) a vzniká nový obsah typu *Stránka textu*. Do nového obsahu je uložen rozpoznaný text a do příslušných polí vazby na obrazový dokument a periodikum, ke kterému obrazový dokument patří. Název obsahu vzniká z názvu nadřazeného periodika a čísla, které je na

konci názvu souboru každého obrazového dokumentu. Existoval-li již obsah zpracovávaného obrazového dokumentu, je dle nastavení aktualizován nebo přeskočen. Nově vytvořený nebo aktualizovaný obsah je také automaticky zaindexován do vyhledávání, pokud je modul takto nastaven.

Dávková úloha obsahuje pouze jednu operaci provádějící postup popsaný výše. V rámci provádění je pomocí cyklu nastaven počet zpracovaných obrazových dokumentů v jednom provedení operace tak, aby byl v jednom provedení operace zpracován počet obrazových dokumentů nastavený v nastavení modulu. Po dokončení dávkové operace je uživatel navrácen na obrazovku s nastavením modulu a je mu zobrazena hláška o úspěšném dokončení.

5.5.1 Zobrazení výsledku vyhledávání

Po implementaci mechanismu zpracování obrazových dokumentů pomocí OCR bylo také důležité umožnit vyhledávání v nově vytvořeném obsahu. Formulář pro vyhledávání byl vytvořen pomocí uživatelského rozhraní, které poskytuje modul *Search Views* (popsán v kapitole 3.3.5). Nově vytvořený formulář obsahuje pouze jeden filtr pro vyhledávání. Tento filtr nabízí hledání v poli *Body*, které u typu obsahu *Stránka textu*, na který je vyhledávání omezeno, obsahuje zpracovaný text obrazových dokumentů. Po dokončení vyhledávání jsou zobrazeny výsledky obsahující názvy nalezených stránek textu (pole *Nadpis*) a náhled původního obrazového dokumentu. Náhled obrazového dokumentu také slouží jako odkaz. Odkaz je vytvořen složením polí *Obrázek (cesta)* a *Obrázek (soubor)*, která jsou součástí vyhledávacího formuláře, ale nejsou pro uživatele viditelná. Odkaz má například následující tvar:

`/iipimage/bysearch/posel-od-cerchova-1872-03-30-n1_0055`

První část odkazuje určuje, že se o zobrazení obrazových dokumentů postarají funkce modulu *IIP Image Integration base* (popsán v kapitole 3.3.4). Tento modul umožňuje zobrazit galerii obrazových dokumentů, ovšem s tím, že jsou zobrazeny obrazové dokumenty ke konkrétnímu typu obsahu. Standardně galerie zobrazuje pro jedno číslo periodika všechny obrazové dokumenty jeho stránek. Aby bylo možné odlišit, zda má být obrazový dokument otevřen tímto způsobem, nebo zda má být otevřen novým způsobem, kde budou zobrazeny obrazové dokumenty nalezené vyhledáváním, je přidána druhá část odkazu. U standardního řešení je v odkazu na druhém místě definováno ID obsahu, ke kterému mají být zobrazeny obrazové dokumenty. Odkaz ve vyhledávání ve stránkách textu namísto tohoto ID obsahuje konstantní text *bysearch*. Poslední částí odkazu je název konkrétního obrazového dokumentu, který má být zobrazen.

Modul *IIP Image Integration base* musel být dále upraven, aby uměl zohlednit druhou část odkazu a zobrazil správný obsah. Zobrazení, respektive přípravu dat pro zobrazení, obrazových dokumentů zajišťuje v tomto modulu funkce *iipimage_base_get_data*. Při běžném zobrazení tato funkce dohledá dle vybraného obrazového dokumentu adresář, ve kterém se obrazový dokument nachází, a z něj přečte ostatní obrazové dokumenty, které se mají zobrazit. Pro účely nového zobrazení byla funkce upravena. V případě, že se v odkazu na druhé pozici nachází již zmíněné klíčové slovo *bysearch*, je namísto hledání adresáře načten obsah z proměnné *images_ids* uložené v globální proměnné session. V této proměnné jsou uložená ID nalezených stránek textu. Tato ID se postupně projdou, načte se konkrétní obsah a z polí *Obrázek (cesta)* a *Obrázek (soubor)* je složena cesta k jednotlivým obrazovým dokumentům. Tyto cesty jsou uloženy do pole, které se později prochází a jsou z něj načítány obrazové dokument k zobrazení. Uložení ID nalezených stránek textu do proměnné je zajištěno pomocí háčku *hook_search_api_solr_search_results_alter*, implementovaného v nově vytvořeném modulu *Custom OCR*. Tento háček při každém vyhledávání smaže proměnou *images_ids* a poté do ní nastaví ID nalezených záznamů.

6 Testování

Během vytváření a integrace modulů do redakčního systému webového portálu Porta fontium bylo prováděno pravidelné testování. Implementovaná funkcionalita byla testována jednak autorem práce, a následně také zaměstnanci Západočeské univerzity v Plzni, konkrétně kolegy z Katedry informatiky a výpočetní techniky. Mimo tohoto uživatelského testování byly vytvořeny automatické testy pro urychlení hledání strojově odhalitelných chyb. Vytvořené automatické testy jsou dvojího typu. Prvním typem jsou testy v kontextu redakčního systému Drupal (viz. kapitola 2.5), druhým typem jsou testy s využitím nástroje *Selenium* [13].

6.1 Modul podobných slov

Pro modul podobných slov vznikla sada testů v obou výše zmíněnými technologiích. První část testů využívá modul redakčního systému *SimpleTest*. Jak lze vidět v popisu struktury adresáře modulu (kapitola 4.4), součástí modulu je soubor *search_api_solr_similar_words.test*, který obsahuje třídu *SearchApiSolrSimilarWordsUnitTestCase*. Tato třída dědí je potomkem *DrupalWebTestCase*, díky čemuž získává funkce pro testování.

V rámci této třídy je vytvořen testovací scénář s názvem *Similar Words units tests*. Název a popis testu nejsou přeloženy do českého jazyka, protože tyto texty by dle konvencí redakčního systému Drupal neměly být překládány. Scénář obsahuje následující čtyři testy:

- testCallSimilarWords

  Test, který kontroluje, zda funkce *callSimilarWords* vrátí deset slov, získaných z externího API, ve správném formátu a korektně seřazených dle podobnosti.

- testCallSimilarWordsPhrase

  Provádí také kontrolu funkce *callSimilarWords*. V tomto případě ovšem testuje vrácená slova pro zadanou dvouslovnou frázi. Aby byl test splněn, musí být vráceno deset nejpodobnějších slov pro každé slovo z testované fráze.

- testGetTopSimilarWords

  Slouží k otestování funkce *getTopSimilarWords*, u které zkontroluje, že

ze všech podobných slov pro jednu dvouslovnou frázi je vráceno deset nejpodobnějších.

- testGetSimilarWordsForEverySearched

  Provede test funkce *getSimilarWordsForEverySearched*. Cílem je otestování, zda funkce ze všech podobných slov pro jednu dvouslovnou frázi vrátí podobná slova ve správném poměru - tedy pět nejpodobnějších pro první slovo fráze a pět nejpodobnějších pro druhé slovo fráze.

- testCallSimilarWordsPhraseUnsucessful

  Poslední test slouží ke kontrole, že v případě selhání některého z volání externího API nejsou vrácena žádná podobná slova. Testován je krajní případ, kdyby pro dvouslovnou frázi došlo k výpadku API po získání podobných slov pro první slovo, a pro druhé už by se podobná slova získat nepodařilo.

Ačkoliv třída testovacího scénáře dědí od třídy *DrupalWebTestCase*, která slouží spíše k funkčním testům, jsou vytvořené testy jednotkové. Protože ale testy využívají konfiguraci modulu uloženou v databázi, musí být použit tento předek poskytující kontext redakčního systému.

Aby bylo testování nezávislé na využívaném externím API, byl pro testování vytvořen modul *search_api_solr_similar_words_test*, uložený v podadresáři *tests*. Tento modul slouží pouze pro testování, je tedy skrytý a nenachází se v seznamu modulů redakčního systému. Modul obsahuje funkci *custom_http_request*, která vrací hodnoty ve stejném formátu jako funkce *drupal_http_request*, využívaná pro volání HTTP požadavků. Návratové hodnoty této funkce jsou ovšem zúžené na několik málo možností, sloužících pro účely testování.

Před každým spuštěným testem je tento modul aktivován a v redakčním systému je nastaveno, že se má namísto standardní funkce pro volání HTTP požadavků využívat funkce poskytnutá tímto modulem.

Druhá část testů, jak již bylo zmíněno, využívá nástroj *Selenium*. V programovacím jazyce Java byl vytvořen nový projekt, využívající tento nástroj a obsahující jednu sadu testů. Předpokladem pro správnou funkčnost testů je korektní natavení modulu a funkční externí API pro získávání podobných slov. V rámci testů se postupně provede následující testovací scénář:

- Přihlášení do redakčního systému Drupal.

- Otevření konfiguračního formuláře testovaného modulu. Z formuláře se poté přečte adresa externího API a nastaví se zobrazovaný počet podobných slov na deset slov.
- Získání podobných slov z externího API.
- Otevření vyhledávacího formuláře a ověření, že se všechna získaná podobná slova zobrazila ve formuláři.

6.2 Modul OCR

Pro automatické testování modulu OCR byl, na rozdíl od předchozího modulu, vytvořen testovací scénář pouze pomocí modulu *SimpleTest*. Opět tedy vznikl soubor *.test*, obsahující novou třídu *CustomOCRUnitTestCase*, dědící od třídy *DrupalWebTestCase*. Zároveň byl vytvořen skrytý modul (*custom_ocr_test*), který je před testováním aktivován a který obsahuje funkci *custom_perform_ocr*. Tato funkce pro potřeby testování nahrazuje původní funkci *perform_ocr*. Ta v testovaném modulu volá skript provádějící OCR. Nahrazující funkce namísto provádění skutečného OCR, vytvoří výsledný soubor s předem určeným textem. Test lze tedy provést bez závislosti na funkčním OCR enginu.

Protože hlavní činností testovaného modulu je vytváření nového obsahu, je před spuštěním testu nutné připravit testovací prostředí. Ve funkci *setUp* jsou proto vytvořeny dva nové typy obsahu (*Periodikum* a *Stránka textu*). K typům obsahů jsou vytvořeny pole pro uložení všech informací využívaných testovaným modulem, jako je například cesta k obrázkům periodika nebo název obrazového dokumentu (která pole modul využívá se lze dozvědět v kapitole 5.5).

Samotný test ověří správný průběh funkce *run_ocr* nad dvěma obrazovými dokumenty. Funkce je v testovaném modulu volána pro každý zpracovávaný obrazový dokument. Výsledkem vykonání této funkce je nově vytvořený obsah, obsahující text ve správném poli. V případě, že obsah nebyl vytvořen, nebo obsah neobsahuje text, test ohlásí chybu.

7 Úprava modulů pro Drupal 8

Redakční systém Drupal ve verzi 8 přinesl v oblastech vývoje nových modulů změny, které neumožňují použití modulů ze starších verzí bez toho, aby byly pro provoz v nové verzi upraveny. Tato kapitola se věnuje těmto změnám a obsahuje analýzu částí vytvořených modulů, které je nutné pro provoz ve verzi 8 změnit [21]. V následujícím textu bude pro příklady využíván modul *custom_ocr*, ačkoliv se změny týkají obou vytvořených modulů. Výsledkem úprav by měl být správně upravený kód a odstranění, nebo přesunutí kódu následujících háčků do jiných souborů:

- *custom_ocr_menu*
- *custom_ocr_settings_form*
- *custom_ocr_validate*

7.1 Konfigurace modulu

První změnou je úprava ukládání a získávání údajů z konfigurace modulu. V původní verzi byly pro ukládání konfigurace používány globální proměnné redakčního systému a k nim patřící funkce *variable_set* a *variable_get*. Ve verzi 8 se konfigurace neukládá do obecných proměnných, ale samostatně a přistupuje se k ní pomocí objektu. Pro získání objektu nastavení lze využít funkci *Drupal::config('custom_ocr.settings')*. Vrácený objekt umožňuje pomocí funkce *get* získat hodnotu z nastavení a pomocí funkce *set* hodnotu nastavit. Ukázka práce s hodnotami konfigurace:

```
// Drupal 7
variable_get('custom_ocr_address', 'vychozi_hodnota');

// Drupal 8
$config->get('custom_ocr.address');
```

V ukázce lze postřehnout nahrazení posledního podtržítka v názvu proměnné za tečku. Původní řešení sdílelo proměnné v celém systému a v případě, že by dva moduly používaly stejný název proměnné, mohlo dojít k nechtěnému přístupu ke špatné proměnné. Nyní je ovšem konfigurace vázána

na konkrétní modul, a proto se název skládá z názvu modulu a názvu konkrétní hodnoty, které jsou oddělené tečkou. Aby modul korektně fungoval je tedy důležité nahradit použití těchto proměnných.

7.2 Informace o modulu

Jako další byla upravena práce se získáváním informací o jednotlivých modulech. V nové verzi redakčního systému používají *.info* soubory novou příponu *.yml*. Z původního názvu *custom_ocr.info*, musí být soubor přejmenován na *custom_ocr.info.yml*. Po přejmenování je nutné upravit informace uvnitř souboru. Nyní se textové informace definované v souboru zapisují do jednoduchých uvozovek. Také se namísto znaku „rovná se“(=) používá znak „dvojtečka“(:). Ukázka souboru z Drupalu verze 7:

```
name = My D7 Module
core = 7.x
```

A ukázka souboru pro Drupal 8:

```
name: 'My D8 Module'
core: 8.x
```

7.3 Formuláře

Další upravenou oblastí je zobrazování vlastních formulářů. Původní verze redakčního systému využívala háček *hook_menu*. Implementací tohoto háčku bylo možné přidat na konkrétní adresu vlastní formulář. V nové verzi se používá nový systém, ve kterém je k mapování formulářů na adresy využito nových souborů. Musí vzniknout nový soubor *custom_ocr.routing.yml*. Do tohoto souboru se přenesou všechny informace dříve přidávané zmíněným háčkem. Ukázka nového souboru:

```
custom_ocr.config:
  path: 'admin/config/search/custom_ocr'
  defaults:
    _form: '/Drupal/custom_ocr/Form/ConfigForm'
    _title: 'Simple Configuration'
  requirements:
    _permission: 'administer site configuration'
```

V ukázce je zobrazen obsah souboru v případě, že má být zobrazen konfigurační formulář. Formulář bude zobrazen na adrese určené vlastností *path*.

Je zde také definován titulek stránky a role, které mají oprávnění formulář zobrazit. Ve vlastnosti *_form* je určen soubor, který bude obsahovat definici formuláře. V nové verzi redakčního systému se zobrazení formulářů neprovádí za pomoci funkce *drupal_get_form*, ale každý formulář je nadefinovaný ve vlastním souboru. Formulář určený nastavením z ukázky by se nacházel v podadresáři *src/Form* v adresáři modulu a byl by pojmenován *ConfigForm.php*.

Soubor s formulářem je vytvářen ve stejných konvencích jako ostatní soubory modulu obsahující kód v jazyce PHP. Je tedy uvozen tagem *<php* a tento tag není na konci souboru uzavřen. V souboru je definována nová třída, která je potomkem třídy *ConfigFormBase*. Ve třídě je důležité vytvořit funkce *buildForm*, *validateForm* a *submitForm*, popsané v následujících kapitolách.

7.3.1 *buildForm*

Funkce *buildForm* zajišťuje vykreslení samotného formuláře. Stejně jako dříve jsou této funkci předány argumenty *$form* a *$form_state*. Formát vytváření formuláře přidáváním definicí polí do proměnné *$form* zůstal zachován. Je důležité si pouze ohlídat nastavení výchozích hodnot pomocí nové práce s konfigurací. Navíc funkce *t()*, zajišťující překlady textů do správného jazyka, musí být volaná z předka třídy, tzn. *$this->t("Text")*. Žádné další úpravy by nemělo být potřeba provádět.

7.3.2 *validateForm*

Pro validaci je namísto háčku využita funkce *validateForm*. V následující ukázce kódu je zobrazena jednoduchá validace, uložená v této funkci:

```
$max_words = $form_state->getValue('max');

if(!is_numeric($max_words)) {
  $form_state->setErrorByName('max',
    $this->t('Vložte prosím číslo.'));
}
```

Z ukázky je patrné, že se změnila práce se získáním hodnoty z formuláře a nastavením chybové hlášky. Proměnná *form_state* je nyní namísto pole objektem, což mění způsob získávání jejích hodnot. Chybová se hláška se navíc nenastavuje samostatnou funkcí, ale funkcí *setErrorByName* z objektu *form_state*.

7.3.3 *submitForm*

Poslední důležitou funkcí je funkce pro uložení změněných hodnot. V původní verzi redakčního systému se zápis hodnot prováděl automaticky, pokud byla využita funkce *system_setting_form*. Nově je nutné uložit změny korespondujícím úsekem kódu. Ukázka uložení jedné hodnoty:

```
$config->set('custom_ocr.max',
  $form_state->getValue('max'));
$config->save();
```

Každá hodnota konfigurace musí být takto nastavena a na závěr uložena funkcí *save*.

7.4 Menu

Formulář popsaný v předchozí kapitole bude sice dostupný při otevření konkrétní adresy, nikoliv ovšem odkazem z administračního menu. Pro přidání formuláře do menu musí být vytvořen další soubor s názvem *custom_ocr.links.menu.yml*. Ukázka obsahu souboru:

```
custom_ocr.admin:
  title: 'Custom OCR'
  description: 'Settings of Custom OCR module'
  parent: system.admin_config_content
  route_name: custom_ocr.config
```

V ukázce kódu je vidět název a popis, které budou zobrazené v menu *Nastavení*. Vlastnost *parent* určuje, ve které části menu (a nastavení) se záznam zobrazí. Vlastnost *route_name* odkazuje na záznam nadefinovaný v souboru *custom_ocr.routing.yml*, vytvořeném v kapitole 7.3. V tomto souboru je vyhledána hodnota vlastnosti a po otevření je zobrazen korespondující formulář.

8 Závěr

Cílem této práce byla integrace nové funkcionality do redakčního sytému Drupal, používaném ve webovém portálu Porta fontium. Před zahájením práce na integraci této funkcionality bylo nutné prozkoumat samotný redakční systém Drupal ve verzi 7 a možnosti, které administrátorům a vývojářům nabízí. Po porozumění redakčnímu systému bylo důležité prozkoumat konkrétní instalaci webového portálu Porta fontium. Portál obsahuje dlouhodobě provozovaný redakční systém s vlastní konfigurací a s několika stovkami nainstalovaných modulů, které ovlivňují práci s redakčním systémem. Po analýze webového portálu byly prozkoumány dodané moduly poskytující novou funkcionalitu a byla navrhnuta vhodná integrace nových modulů redakčního systému.

Dle návrhu byly vytvořeny dva nové moduly ‚modul podobných slov a modul OCR, splňující požadavky zadavatele. Moduly poskytují provozovateli webového portálu jednodušší možnost vytváření obsahu z obrazových dokumentů a uživatelům umožňují získat relevantnější výsledky při vyhledávání.

Oba nově vytvořené a integrované moduly byly řádně otestovány a obsahují sadu automatických testů, využitelných pro testování při budoucích úpravách a rozšířeních. Aby bylo možné moduly využívat i po případné aktualizaci webového portálu na novější verzi redakčního systému Drupal, je závěr práce věnován úpravám potřebným pro provoz vytvořených modulů ve verzi 8.

Zdrojové kódy vytvořených modulů splňují konvence pro vývoj modulů redakčního systému Drupal a jsou řádně okomentovány, aby bylo ulehčeno pochopení kódu dalším vývojářům, kteří by chtěli moduly rozšiřovat o další funkcionalitu.

Práce splňuje všechny body zadání a integrované moduly splňují požadavky a očekávání zadavatele práce. Získané znalosti budou využity při dalším rozvoji vytvořených modulů a při integraci dalších modulů automatického zpracování textu.

Zkratky

- API *Application Programming Interface* - Rozhraní pro programování aplikací
- CMS *Content Manager System* - Systém pro správu obsahu
- CSV *Comma-Separated Values* - Jednoduchý formát souboru pro výměnu dat
- HTML *Hypertext Markup Language* - Značkovací jazyk používaný pro vytváření webových stránek
- OCR *Optical Character Recognition* - Převod obrazových dokumentů do strojového textu
- PHP *PHP: Hypertext Preprocessor* - Skriptovací programovací jazyk
- RSS *Rich Site Summary* - Informační zdroj upozorňující na nové změny
- Solr *Searching On Lucene w/ Replication* - Vyhledávací engine založený na platformě Apache Lucene
- URL *Uniform Resource Locator* - Odkaz na konkrétní zdroj
- XML *Extensible Markup Language* - Obecný značkovací jazyk

Seznam obrázků

Literatura

[1] *About | Drupal.org* [online]. Drupal, 2020. [cit. 20.2.2020]. Dostupné z: https://www.drupal.org/about.

[2] *Batch API overview* [online]. Drupal, C2000-2020. [cit. 2.4.2020]. Dostupné z: https://www.drupal.org/docs/7/api/batch-api/overview.

[3] *hook_block_info | block.api.inc | Drupal 7.x | Drupal API* [online]. Drupal, C2000-2020. [cit. 1.4.2020]. Dostupné z: https://api.drupal.org/api/drupal/modules!block!block.api.php/function/hook_block_info/7.x.

[4] *Usage statistics of content management systems* [online]. W3Techs, C2009-2020. [cit. 25.1.2020]. Dostupné z: https://w3techs.com/technologies/overview/content_management.

[5] *Drupal 9 release date and what it means* [online]. Drupal, C2000-2020. [cit. 15.1.2020]. Dostupné z: https://www.drupal.org/docs/9/drupal-9-release-date-and-what-it-means.

[6] *Drupal.cz | Drupal.cz* [online]. Drupal, 2020. [cit. 20.2.2020]. Dostupné z: https://www.drupal.cz/.

[7] *Our history* [online]. Drupal, 2020. [cit. 5.1.2020]. Dostupné z: https://www.drupal.org/about/history.

[8] *Localization API overview | Localization API | Drupal 7 guide on Drupal.org* [online]. Drupal, C2000-2020. [cit. 23.5.2020]. Dostupné z: https://www.drupal.org/node/322729.

[9] *Simpletest Testing overview (Drupal 7) | Automated testing for Drupal 7 | Drupal 7 guide on Drupal.org* [online]. Drupal, C2000-2020. [cit. 30.5.2020]. Dostupné z: https://www.drupal.org/docs/7/testing/simpletest-testing-overview-drupal-7.

[10] *Usage statistics for Drupal core* [online]. Drupal, 2020. [cit. 25.1.2020]. Dostupné z: https://www.drupal.org/project/usage/drupal.

[11] *Hooks | module.inc | Drupal 7.x | Drupal API* [online]. Drupal, C2000-2020. [cit. 28.3.2020]. Dostupné z: https://api.drupal.org/api/drupal/includes!module.inc/group/hooks/7.x.

[12] *About* [online]. IIPImage, C2020. [cit. 12.2.2020]. Dostupné z: https://iipimage.sourceforge.io.

[13] *SeleniumHQ Browser Automation* [online]. Software Freedom Conservancy, 2020. [cit. 1.6.2020]. Dostupné z: `https://www.drupal.org/docs/7/testing/simpletest-testing-overview-drupal-7`.

[14] *Tesseract Open Source OCR Engine (main repository)* [online]. Tesseract, 2020. [cit. 21.4.2020]. Dostupné z: `https://github.com/tesseract-ocr/tesseract`.

[15] *Simpletest Testing overview (Drupal 7)* [online]. Drupal, C2000-2020. [cit. 15.5.2020]. Dostupné z: `https://www.drupal.org/docs/7/testing/simpletest-testing-overview-drupal-7`.

[16] christophe. *Synonyms with Search API* [online]. colorfield, 8.3.2020. [cit. 15.4.2020]. Dostupné z: `https://colorfield.be/blog/synonyms-search-api`.

[17] JD Leonard. *Entities in Drupal 7 & the Entity API* [online]. SlideShare, 11.3.2013. [cit. 10.2.2020]. Dostupné z: `https://www.slideshare.net/jdleonard/entities-in-drupal-7-the-entity-api`.

[18] MCKINNEY, Shea Ross. *A better search for Drupal* [online]. Standford University, 26.2.2015. [cit. 15.3.2020]. Dostupné z: `https://swsblog.stanford.edu/blog/better-search-drupal`.

[19] SHREVES, Ric a DUNWOODIE, Brice. *Drupal 7 Bible*. Wiley, 2011. ISBN 978-0470530308.

[20] sillygwailo. *Writing module .info files (Drupal 7.x)* [online]. Drupal, 6.8.2009. [cit. 24.3.2020]. Dostupné z: `https://www.drupal.org/docs/7/creating-custom-modules/writing-module-info-files-drupal-7x`.

[21] SIPOS, Daniel. *Drupal 8 Module Development: Build and customize Drupal 8 modules and extensions efficiently*. Packt Publishing, 2017. ISBN 978-1782168775.

[22] VEKONY, Balasz. *Faceted Search: The Ultimated Guide for eCommerce Sites* [online]. Prefixbox, 10.3.2019. [cit. 17.3.2020]. Dostupné z: `https://www.prefixbox.com/blog/faceted-search`.

[23] VIXIE Paul. *crontab(5): tables for driving cron* [online]. die.net. [cit. 21.4.2020]. Dostupné z: `https://linux.die.net/man/5/crontab`.

Přílohy

A Administrátorská příručka

Oba vytvořené moduly obsahují vlastní konfigurační formulář dostupný administrátorovi a nabízející několik možností konfigurace.

A.1 Modul podobných slov

Obrázek A.1 zobrazuje nastavení modulu podobných slov. Popis jednotlivých voleb:

- Formuláře

  Určuje, ve kterých formulářích se budou zobrazovat podobná slova. Na každém řádku textového pole by měl být zaznamenán jeden formulář a jedno pole, pro které se podobná slova zobrazí. Zadává se ID formuláře, které lze na jednotlivých formulářích zobrazit pomocí následující volby. Pole je na formuláři nalezeno dle svého identifikátoru. Ten lze nalézt přímo při definovaní formuláře, nebo v URL adrese. Identifikátor pole se objeví jako parametr adresy v případě, že je do pole zadán výraz a je spuštěno vyhledávání.

- Zobrazit ID formuláře ve webových formulářích

  Zaškrtnutím této volby se ve všech formulářích redakčního systému začne zobrazovat jejich ID. Tato volba by měla být použita pouze pro zjištění ID konkrétního formuláře. Po jeho zjištění je doporučeno volbu vypnout.

- Maximální počet podobných slov

  Určuje, kolik podobných slov bude uživateli nabídnuto ve formuláři vyhledávání. Je možné zadat celočíselné hodnoty nula až deset.

- Adresa API

  Adresa externího API poskytujícího podobná slova.

- Jak se má vybrat TOP X podobných slov

Při vyhledávání víceslovných frází umí modul zobrazit podobná slova dle dvou kritérií. Prvním kritériem je zobrazení těch nejpodobnějších. V takovém případě je ze všech podobných slov vybráno N nejpodobnějších a může se stát, že pro některá ze slov fráze nebude zobrazeno žádné podobné slovo. Pokud je vybráno kritérium „Poměrem“, jsou ke každému slovu fráze vybrána podobná slova ve stejném poměru. Tedy pro dvouslovnou frázi bude uživateli zobrazeno pět podobných slov od prvního slova a pět podobných slov od druhého.

Obrázek A.1: Ukázka nastavení modulu podobných slov

A.2 Modul OCR

Obrázek A.2 zobrazuje nastavení modulu OCR. Popis jednotlivých voleb:

- Obrázky

  Adresář, ve kterém se nacházejí obrázky ke zpracování. Modul načte všechny obrázky s příponou *.jp2* včetně těch, které jsou umístěné v podadresářích.

- Jazyk

  Jazyk, ve kterém jsou zpracovávané obrázky. Volba je předána použitému OCR enginu, aby mohl použít odpovídající jazykové modely.

- Velikost dávky

  Počet obrázků zpracovaných v jedné dávce. Po spuštění OCR je zobrazena stránka informující o průběhu zpracování. Tato stránka je aktualizována po zpracování každé dávky. Je tedy vhodné zvolit takové číslo, které umožní pravidelnou aktualizaci stavu zpracování.

- Aktualizovat již existující stránky

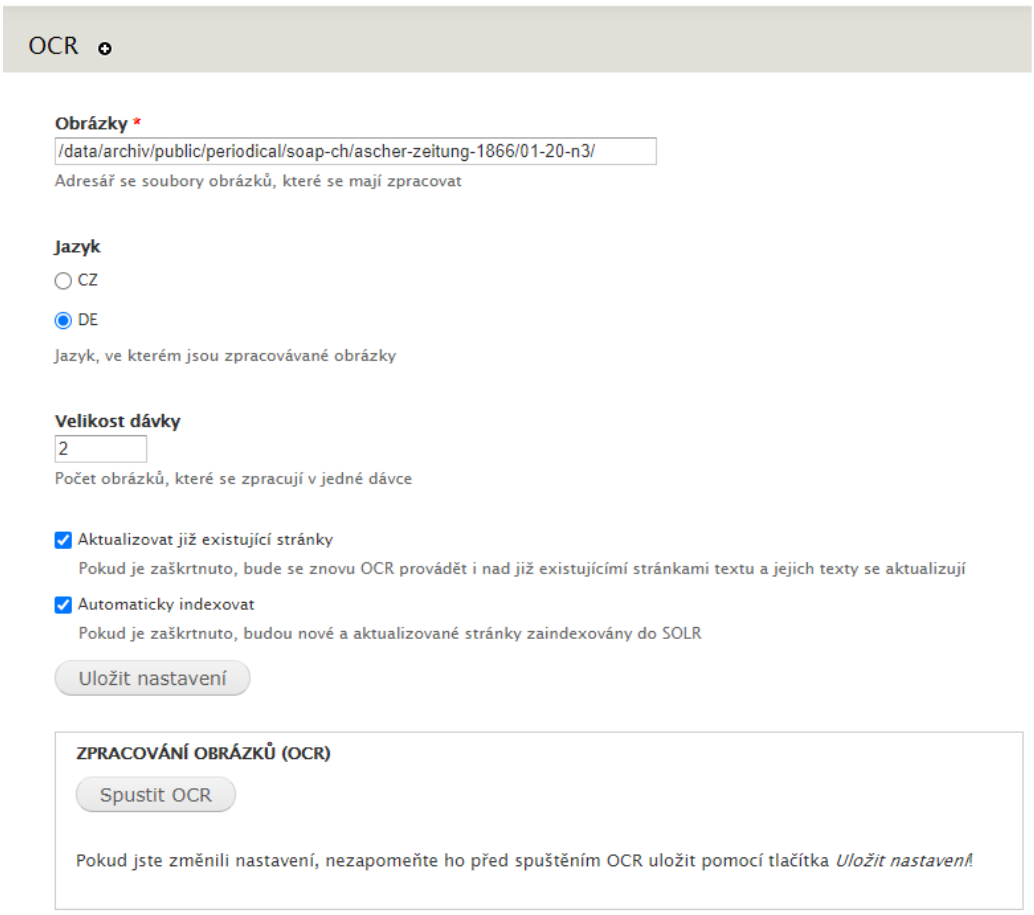
  Volba umožňující automatickou aktualizaci již vytvořeného obsahu. Může být vhodná při změně OCR enginu.

- Automaticky indexovat

  Hodnota parametru rozhodne o tom, zda se má nově vytvořený obsah automaticky indexovat do Solr.

- Spustit OCR

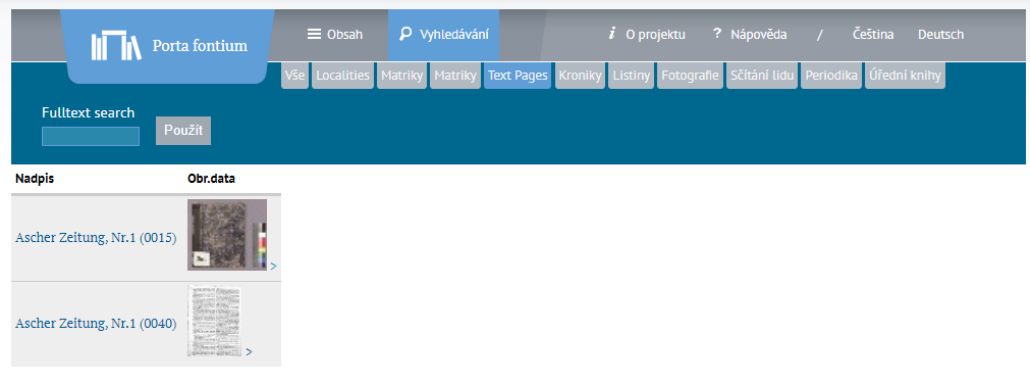
  Tlačítko, které spustí samotný proces OCR. Po jeho stisknutí se začnou zpracovávat obrázky z výše zvoleného adresáře.



Obrázek A.2: Ukázka nastavení modulu OCR

B Uživatelská příručka

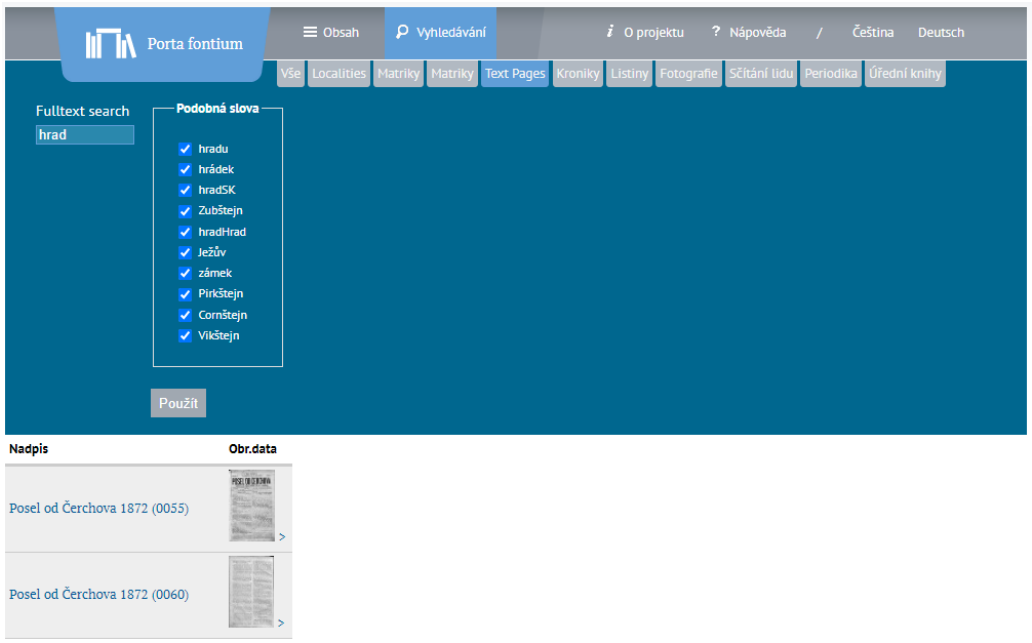
Modul podobných slov poskytuje koncovému uživateli novou funkcionalitu popsanou dále. Po otevření vyhledávacího formuláře není podpora podobných slov vidět. Formulář nabízí pouze pole k vyhledání a podobná slova nejsou nikde zobrazena (obrázek B.3).



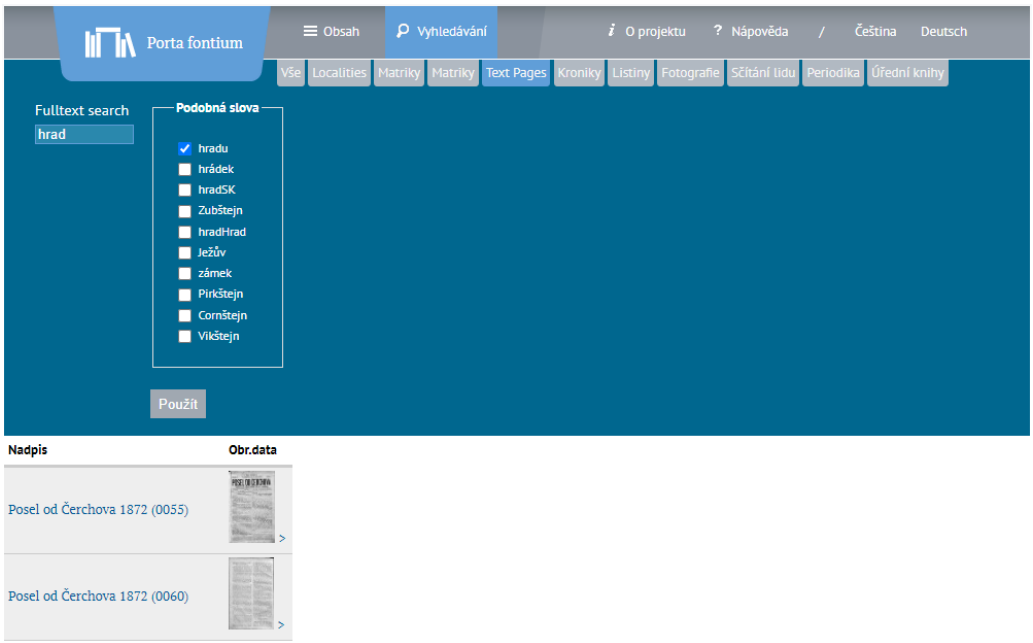
Obrázek B.3: Ukázka vyhledávacího formuláře

Poté, co uživatel zadá výraz, který chce vyhledat, je spuštěno získání podobných slov pro frázi zadanou do vyhledávacího pole. Pokud je úspěšné, je výraz zadaný uživatelem automaticky rozšířen o podobná slova a tato slova jsou ve formuláři zobrazena (obrázek B.4).

Po zobrazení slov si uživatel může zvolit, zda mu podobná slova vyhovují. Pokud některé z nich nechce do vyhledávání zahrnout, může jej v zobrazeném seznamu odškrtnout a spustit vyhledávání znovu. V tomto případě už nejsou podobná slova znovu vyhledána, jen se hledaný výraz rozšíří o vybraná podobná slova (obrázek B.5). Pokud uživatel zadá nový výraz k vyhledávání, jsou podobná slova opět vyhledána a všechna automaticky přidána k hledanému výrazu.



Obrázek B.4: Formulář po zadání výrazu a spuštění vyhledávání



Obrázek B.5: Formulář po změně výběru podobných slov