

**Gymnázium, Praha 6, Arabská 14**  
**Ročníková práce**



**Katalog ročníkových prací**

2020

Michail Spiridonov

# Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Podpis: \_\_\_\_\_

# Obsah

Prohlášení.....	2
Obsah .....	3
Anotace: .....	4
Abstract: .....	4
1. Úvod .....	5
2. Zadání .....	6
3. Struktura .....	7
4. Backend .....	8
4.1. Funkce.....	8
4.1.1. Komunikace s klientem a databází .....	8
4.1.2. Správa souborů .....	8
4.1.3. Správa uživatelů a sessions .....	8
4.2. Informace o práci.....	8
4.2.1. Klíčová slova .....	9
4.2.1.1. RAKE – Rapid Automatic Keyword Extraction .....	9
4.2.2. Anti plagiát .....	11
5. Frontend.....	13
5.1. Hlavní stránka a Header.....	13
5.2. Seznam prací.....	14
5.3. Zobrazení práce .....	14
5.4. Vyhledávání prací .....	15
5.5. Přidání práce .....	15
5.6. Nastavení uživatelů.....	15
6. Závěr.....	16
7. Seznam použité literatury .....	17

## **Anotace:**

Cílem práce bylo vytvořit on-line katalog ročníkových prací. V katalogu má být možnost vyhledávání, automatické vytváření klíčových slov a anti plagiát. Práce je psaná v jazyce JavaScript s použitím Node.js, express a React.

## **Abstract:**

The goal of the project was to create an on-line catalogue of term papers. The catalogue should implement search feature, automatic keyword generation from abstract and anti-plagiarism system. The project was made using JavaScript with Node.js framework, express and react.

# 1. Úvod

Tento projekt jsem vytvořil jako ročníkovou a maturitní práci z předmětu programování. Volbu tématu ovlivnil můj zájem o rozšíření svých znalostí v oblasti tvorby webových stránek, serveru s databází a také vytvoření projektu, který by mohl být využitelný studenty. Katalog umožňuje vyhledávání mezi nahranými pracemi podle různých parametrů (název, ročník, třída, autor atd.) a také podle klíčových slov, která jsou automaticky generována. Také dokáže z práce extrahovat jméno autora, název práce, třídu, ročník, předmět a jméno vedoucího práce. Aby tato funkce fungovala musí být práce napsána podle příručky „JAK PSÁT ROČNÍKOVOU PRÁCI“ pro gymnázium Arabská. Nahrávání a odstraňování prací je umožněno pouze určitým uživatelům, ale prohlížet a stahovat práce může každý. Celý projekt je napsaný v jazyce JavaScript, server je vytvořený pomocí frameworku Node.js, front-end je vytvořený pomocí Reactu (JavaScript knihovny pro tvorbu frontendu).

## 2. Zadání

Cílem této práce je vytvoření on-line katalogu ročníkových prací uspořádaný podle ročníků a oborů, s vyhledáváním podle klíčových slov získaných automaticky z anotace práce a možností porovnání podobnosti dokumentací s jinými pracemi (srovnáním textů prací a nalezení stejných vět).

# 3. Struktura

Projekt se skládá z 2 hlavních částí:

- **Backend**
  - Node.js, express – tvorba serveru
  - Generace klíčových slov
  - Anti plagiát
- **Frontend**
  - React – strukturování a render komponentů stránek
  - Vyhledávání, nahrávání prací

## 4. Backend

Server je vytvořený v Node.js pomocí knihovny express. Server komunikuje s databází (MySQL) - vytvoření nové práce, mazání, zobrazení prací (všech, anebo podle parametrů). Při přidání nové práce se automaticky z anotace získá seznam klíčových slov (frází) a zkontroluje se podobnost s ostatními pracemi.

### 4.1. Funkce

#### 4.1.1. Komunikace s klientem a databází

Komunikace s databází a klientem je nejdůležitějším úkolem backendu. Backend zpracovává požadavky z uživatelského rozhraní a posílá odpovědi. Při přidání a odstranění práce, nahrání souboru, zobrazení prací, všechny tyto požadavky přijímá backend, posílá dotaz na databázi, odpověď databáze zpracuje a pošle jí zpět klientovi. Ten tuto odpověď zobrazí v uživateli přívětivé podobě.

#### 4.1.2. Správa souborů

Všechny nahrané práce se ukládají do složky “files”, při nahrání je soubor práce přejmenován (malá písmena a mezery jsou nahrazeny pomlčkami). Toto zajišťuje minimální konflikty s různými souborovými systémy. Při stahování je soubor znovu přejmenován, tentokrát podle názvu práce.

#### 4.1.3. Správa uživatelů a sessions

Při přihlášení uživatele se vytvoří tzv. “session” - na počítač uživatele se uloží soubor (cookie), díky kterému uživatel získá přístup k nastavení stránky, mazání a nahrávání prací a správě uživatelů. 1 uživatel je hlavní – “admin”. Tohoto uživatele nelze odstranit, je to jediný uživatel, který může spravovat ostatní uživatele – přidávat, odebírat, měnit jejich hesla atp. Uživatelé se ukládají v databázi, kde je uloženo jejich uživatelské jméno (pomocí kterého se přihlašují), heslo a jméno. Heslo je skryté pomocí hashovacího algoritmu na základě enkrypce “blowfish”, což zamezuje získání uživatelských hesel při neoprávněném přístupu do databáze.

### 4.2. Informace o práci

Poté co je nahrán soubor s prací, je převedený na text, aby se z práce dala extrahovat další data. Pokud je práce psaná podle příručky „JAK PSÁT ROČNÍKOVOU PRÁCI“ pro gymnázium Arabská, dají se z textu získat údaje o práci jako jméno autora, ročník, třída atp., díky čemuž se nemusí při přidávání práce vyplňovat manuálně.



### 4.2.1. Klíčová slova

Klíčová slova jsou získávána z textu anotace práce. Pro získání klíčových slov jsem se rozhodl mezi třemi algoritmy: RAKE, TF-IDF a TEXTRANK. Rozhodl jsem se implementovat algoritmus RAKE, jelikož je rychlý a dokáže získat klíčové fráze, což může usnadnit vyhledávání oproti pouhým slovům.

#### 4.2.1.1. RAKE – Rapid Automatic Keyword Extraction

Tento algoritmus pro začátek potřebuje seznam separátorů frází a separátor slov. Jako separátor slov jsem použil mezeru a vytvořil seznam slov používaných pro separování frází. Algoritmus nejdříve rozdělí text do pole slov, ze kterých se poté vytvoří pole frází.

Například z věty: “Tento algoritmus se používá k automatické extrakci klíčových slov.” Vzniknou tyto fráze:

- “Tento algoritmus”
- “používá”
- “automatické extrakci klíčových slov”

Poté se vytvoří matice počtu slov ve frázích:

	tento	algoritmus	používá	automatické	extrakci	klíčových	slov
Tento	1	1	0	0	0	0	0
Algoritmus	1	1	0	0	0	0	0
Používá	0	0	1	0	0	0	0
Automatické	0	0	0	1	1	1	1
Extrakci	0	0	0	1	1	1	1
Klíčových	0	0	0	1	1	1	1
Slov	0	0	0	1	1	1	1

Tabulka 1, Matice počtu slov ve frázích

Pomocí této matice (*Tabulka 1*) se vypočítá hodnota slova vydělením stupně slova, frekvencí slova. Stupeň (*deg*) slova určuje počet výskytu slova s jinými slovy ve frázi, spočítá se sečtením všech hodnot v sloupci matice. Frekvence (*freq*) slova je počet výskytů tohoto slova v textu. Hodnoty slova (*Tabulka 2*) se uloží do pole.

	tento	algoritmus	používá	automatické	extrakci	klíčových	slov
<i>Deg</i>	2	2	1	4	4	4	4
<i>Freq</i>	1	1	1	1	1	1	1
Hodnota slova ( <i>deg/freq</i> )	2	2	1	4	4	4	4

Tabulka 2, Hodnoty slov

V tomto příkladu se žádné slovo není ve větě 2x, proto frekvence jsou všechny rovné 1.

Pomocí hodnot slov se vypočítají hodnoty frází sečtením hodnot slov ve frázi. Fráze z příkladu pak budou mít tyto hodnoty:

- “tento algoritmus” – 4
- “používá” – 1
- “automatické extrakci klíčových slov” – 16

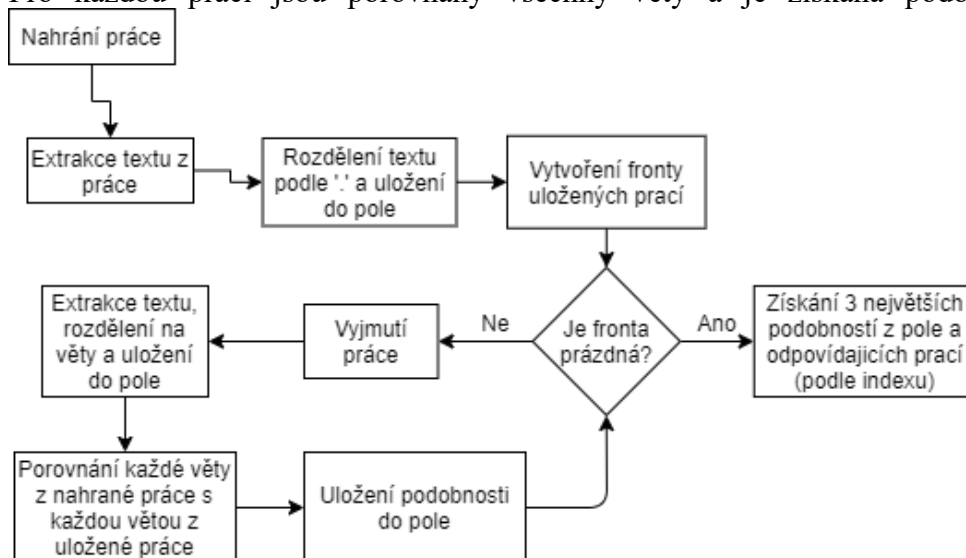
Na základě těchto hodnot jsou vybrány 3 nejlepší klíčové fráze, které jsou použity.

### 4.2.2. Anti plagiát

Hlavní myšlenkou anti plagiátu bylo porovnat všechny věty a získat “podobnost” mezi 2 pracemi pomocí vydělením počtu všech vět v práci počtem stejných vět. Tento způsob porovnávání prací není ideální, protože se dá lehce obejít a má vysokou složitost, přesto dokáže poskytnout alespoň základní údaj o podobnosti prací.

První způsob, kterým jsem se pokusil tohoto docílit, spočíval v porovnání všech vět mezi všemi pracemi:

1. Při nahrání práce se extrahuje text a rozdělí se do vět pomocí teček a velkých písmen.
2. Vytvoří se fronta ostatních prací, z nichž je stejným způsobem extrahovaný a rozdělený text.
3. Pro každou práci jsou porovnány všechny věty a je získána podobnost.



Tento způsob má bohužel pár nevýhod. Hlavní je vysoká složitost – pro tento algoritmus je  $(n^2 * m)$  kde  $n$  je počet vět a  $m$  je počet prací. Při velkém počtu prací by získávání podobnosti tímto způsobem trvalo velmi dlouho. Další nevýhodou je složitost implementace. Kvůli asynchronnosti JavaScriptu je obtížné pracovat s tak velkým objemem dat.

Druhý způsob spočíval v ukládání všech vět do tabulky (Tabulka 3), a pomocné tabulky (Tabulka 4), pro relaci  $M*N$  kde by byly údaje o četnosti.

ID	Věta
1	Věta 1
2	Příklad věty
3	Další příklad věty
4	Věta z úvodu

Tabulka 3, Věty ze všech prací

ID Věty	ID Práce	Četnost
1	1	4
1	2	2
3	2	5
2	4	3
3	6	4

Tabulka 4, Četnost vět v pracích

Při nahrání nové práce by tento algoritmus stejně jak v předchozím způsobu extrahoval text a rozdělil jej na věty. Každá věta by se uložila do databáze a vytvořil by se údaj o četnosti. Pokud by věta už byla uložena v databázi, údaj o četnosti by se jenom aktualizoval. Tento způsob má také poměrně vysokou složitost, ale na rozdíl od předchozího se při nahrání práce extrahuje text pouze z nahrávané práce, a místo porovnávání vět pomocí porovnání řetězců se pouze porovnávají jenom údaje o četnosti.

## 5. Frontend

Frontend aplikace je psaný v JavaScriptu pomocí Reactu. React je framework, který funguje pomocí komponentů, kde každý komponent renderuje (vykresluje) určitou část stránky. Výhodou Reactu je, že komponenty mohou ukládat data v tzv. stavu a „znovurenderovat“ komponent, pokud byl stav aktualizovaný.

Vykresluje se v podstatě jenom jedna stránka – **index.html**, kterou vykresluje **index.js**. **App.js** je hlavním komponentem, do kterého se vkládají další komponenty a **App.css** je jednotný soubor pro styly. Jelikož celý frontend je na jedné stránce, která se jenom pokaždé jinak vykreslí, je aplikace rychlejší a nemusí se načítat různé stránky.

Frontend tohoto projektu je složen z mnoha komponentů, například **Header.js** – záhlaví – na první pohled tento komponent nepotřebuje ukládat žádný stav, ale je to potřeba kvůli zobrazování jiných tlačítek přihlášeným uživatelům – před vykreslením komponentu se ověří, zda existuje „session“ (jestli je uživatel přihlášený). Další výhodou je, že se nemusí vytvářet pro každou stránku zvlášť, stačí jej vytvořit jen jednou a poté importovat do jednotlivých „stránek“.

Důležitějším komponentem je **Add.js**, který odpovídá za přidávání práce. Vykresluje formulář pro údaje, zprostředkovává nahrávání souborů a automaticky vyplňuje políčka formuláře při nahrání práce. **Delete.js** naopak zajišťuje odstranění práce a díky **Download.js** se práce dají stáhnout. **UserSettings.js** zobrazuje uživatelské nastavení, kde je se dají přidávat, editovat a upravovat uživatele.

Díky rozdělení na komponenty se snadněji frontend spravuje. Jednotlivé části jsou často na sobě nezávislé a pokud přestane fungovat jedna, nemusí ovlivnit jiné.

### 5.1. Hlavní stránka a Header

Hlavní (úvodní) stránka obsahuje pouze uvítací text a header s menu, kterým se dá navigovat na další stránky (*Obrázek 1*). Pro nepřihlášené uživatele header obsahuje pouze 4 tlačítka a to: **Home** (Úvodní stránka), **Papers** (Seznam všech prací), **Search** (Vyhledávání prací) a **Login** (přihlášení).



Obrázek 1 – Navigační menu

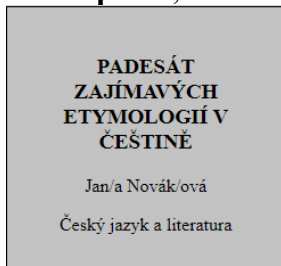
Pro přihlášené uživatele se však menu rozšíří o nové položky: **Add Paper** (Přidání práce) a **User Settings** (Uživatelské nastavení) (*Obrázek 2*).



Obrázek 2 - Navigační menu (přihlášený uživatel)

## 5.2. Seznam prací

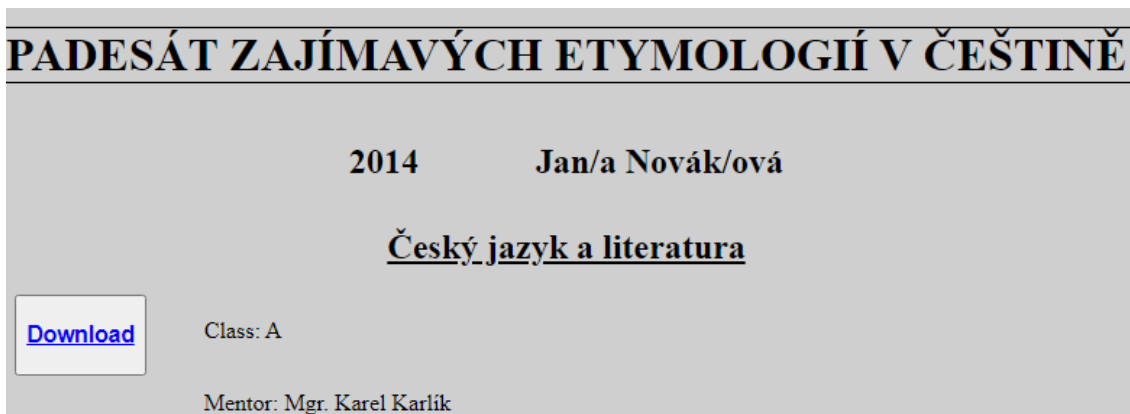
Jednotlivé práce se zobrazují jako dlaždice (Obrázek 3). Aby na dlaždici nebylo moc textu jsem vybral nejdůležitější informace o práci, proto je na každé dlaždici uveden **Autor práce, Název práce a Předmět.**



Obrázek 3 - Dlaždice

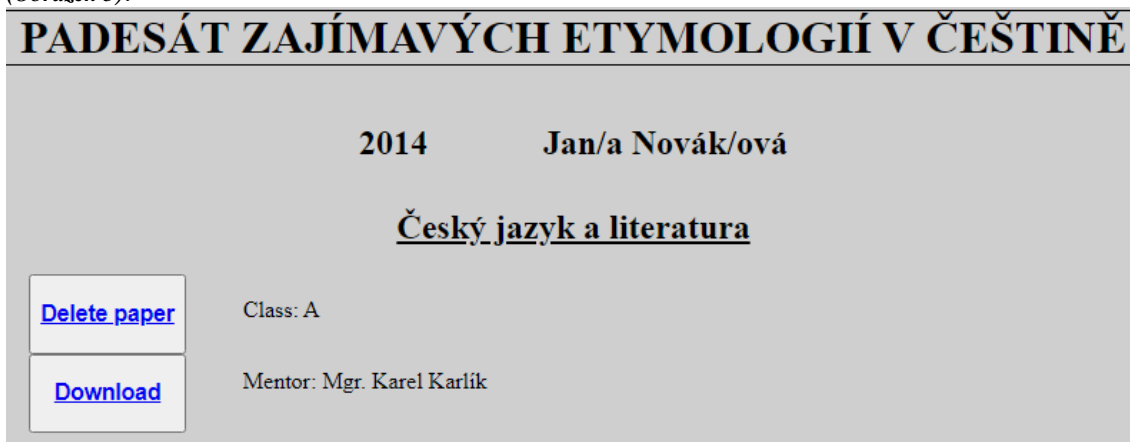
## 5.3. Zobrazení práce

Po kliknutí na dlaždici se zobrazí detaily o práci (Obrázek 4). Uživatel se tak může přesvědčit, že je to ta správná práce, kterou hledal a má možnost ji stáhnout.



Obrázek 4 – Detail práce

Pokud je uživatel přihlášený bude mít v detailu práce také možnost práci odstranit (Obrázek 5).



Obrázek 5 – Detail práce (přihlášený uživatel)

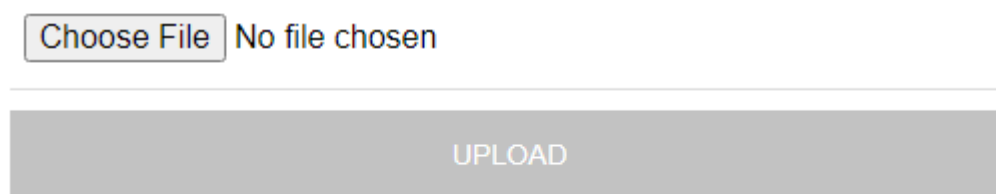
## 5.4. Vyhledávání prací

Vyhledávat se práce dají podle všech parametrů (Autor, Název, Třída, Ročník, Předmět, Vedoucí práce a Klíčová slova). Výsledky hledání se zobrazují stejně jak [seznam všech prací](#) (Obrázek 3).

## 5.5. Přidání práce

Po přihlášení bude uživateli dostupné menu přidání práce, kde může vyplnit údaje o práci a nahrát samotnou práci (Obrázek 6). Pokud je práce napsána podle příručky „JAK PSÁT ROČNÍKOVOU PRÁCI“ pro gymnázium Arabská, budou všechny údaje vyplněny automaticky.

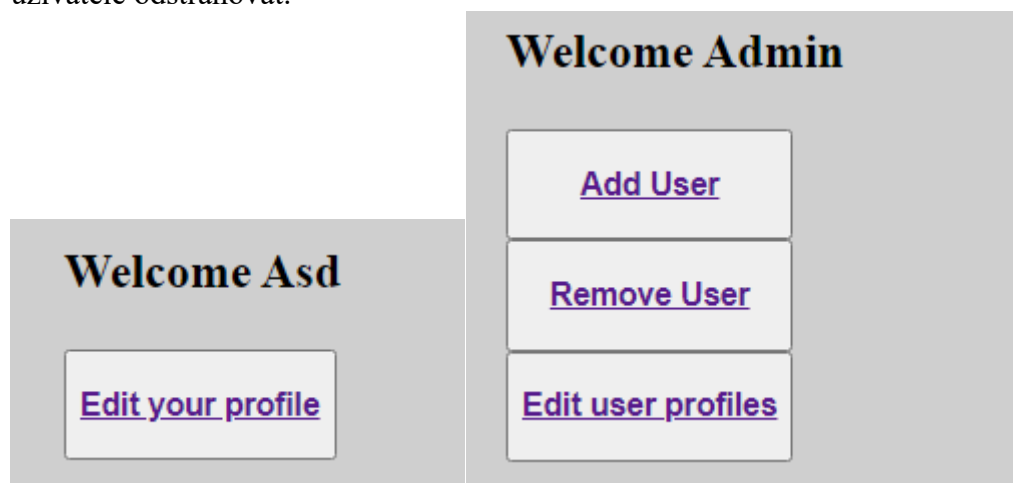
## File upload



Obrázek 6 - Nahrání práce

## 5.6. Nastavení uživatelů

Na stránce nastavení uživatelů najde běžný uživatel pouze jedno tlačítko (Obrázek 7). To jej přivede na stránku, kde si může změnit heslo a uživatelské jméno. Hlavní uživatel (admin) oproti běžnému uživateli může spravovat ostatní uživatele (Obrázek 8). ‚Admin‘ může měnit hesla jiných uživatelů, může přidávat nové uživatele a může také uživatele odstraňovat.



Obrázek 7- uživatelské nastavení (běžný uživatel)    Obrázek 8 - uživatelské nastavení (admin)

## 6. Závěr

Zadání práce jsem splnil, až na anti plagiát. Přestože systém se nezdá být těžký, složitost mých algoritmů je velmi vysoká a systém se nedá rozumně implementovat. V prvním případě kromě složitosti jsem také narazil na problém s asynchronností JavaScriptu, kvůli které se musí jednotlivé funkce volat pomocí tzv. „callback“ funkce, která je vložena do původní funkce. Takto vznikl velmi nepřehledný kód, ve kterém se velmi těžce orientuje. Kromě anti plagiátu není uživatelské rozhraní úplně nastýlované a místy je nepřehledné, přesto je ale plně funkční. Díky tomuto projektu jsem získal cenné zkušenosti ve výrobě webových stránek, práci s databázemi a serverovými aplikacemi. Naučil jsem se pracovat v JavaScriptu, používat React a implementovat databáze v MySQL.



## 7. Seznam použité literatury

- <https://reactjs.org/docs/getting-started.html>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- <https://nodejs.org/en/docs/guides/>
- <https://expressjs.com/en/guide/routing.html>
- Rose, Stuart & Engel, Dave & Cramer, Nick & Cowley, Wendy. (2010). Automatic Keyword Extraction from Individual Documents. 10.1002/9780470689646.ch1.