



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ GEOLOGII, GEOFIZYKI I OCHRONY ŚRODOWISKA

KATEDRA GEOINFORMATYKI I INFORMATYKI STOSOWANEJ

Projekt dyplomowy

Aplikacja do zarządzania zbiorami danych
Dataset management application

Autor: Monika Hertel
Kierunek studiów: Inżynieria i Analiza Danych
Opiekun pracy: dr Paweł Oleksik

Kraków, 2024

Spis treści

Wstęp	3
1 Zagadnienia teoretyczne	4
1.1 Problem tworzenia streszczeń	4
1.2 Tworzenie aplikacji webowych	5
2 Implementacja	5
2.1 Projekt systemu	5
2.2 Struktura aplikacji	5
2.3 Architektura systemu	6
2.4 Scenariusze wykorzystania aplikacji	7
2.5 Testy funkcjonalne	9
2.6 TEMP Idealna aplikacja	9
3 Możliwości rozwoju i wykorzystania aplikacji	9
3.1 Przetwarzanie obrazów	9
3.2 Rozbudowanie funkcjonalności dla plików bazodanowych	9
3.3 Przejście na wersje web	9
Podsumowanie i wnioski	9

Wstep

1 Zagadnienia teoretyczne

1.1 Problem tworzenia streszczeń

(wspomnieć o tym że niektóre pdf posiadają outlines i wtedy są slay) Streszczenie w każdej pracy naukowej jest jej ważną częścią. Ma ono na celu przekazać kluczowe informacje o czytany dokumencie, aby czytelnik mógł ocenić czy dany artykuł (jest dla niego). Automatyzacja tego procesu dąży do ułatwienia autorom tworzenia prac, poprzez skrócenie czasu, którego wymaga kreacja abstraktu. Systemy ATS (ang. *Automatic Text Summarization*) są jednym z cięższych wyzwań sztucznej inteligencji, dotyczących przetwarzania języka naturalnego. Metody wytwarzania streszczeń możemy podzielić na ekstraktywne i abstrakcyjne.

Ekstraktywne a Abstrakcyjne

Podjęcie ekstraktywne polega na wybraniu najważniejszych zdań z całego dokumentu. Algorytm sam w sobie nie tworzy nowych zdań, dlatego też w często jest tak że zdania nie mają sensu w takiej kolejności

1.2 Tworzenie aplikacji webowych

W proces tworzenia aplikacji wchodzi wiele elementów. Na samym poziomie koncepcyjnym jest wiele do przemyślenia

2 Implementacja

2.1 Projekt systemu

System ma na celu ułatwienie przechowywania plików zawierających informacje. Pliki, po dostarczeniu przez użytkownika, są przenoszone do dedykowanego folderu aplikacji. Sprawdzany jest typ pliku i zgodnie z nim wykonywane są inne akcje. Dla plików o rozszerzeniu pdf pobierana jest zawartość i z niej powstaje streszczenie i słowa klucze. Potem wszystkie informacje umieszczane są w bazie danych tak, aby użytkownik mógł wyszukać z użyciem aplikacji dany plik.

2.2 Struktura aplikacji

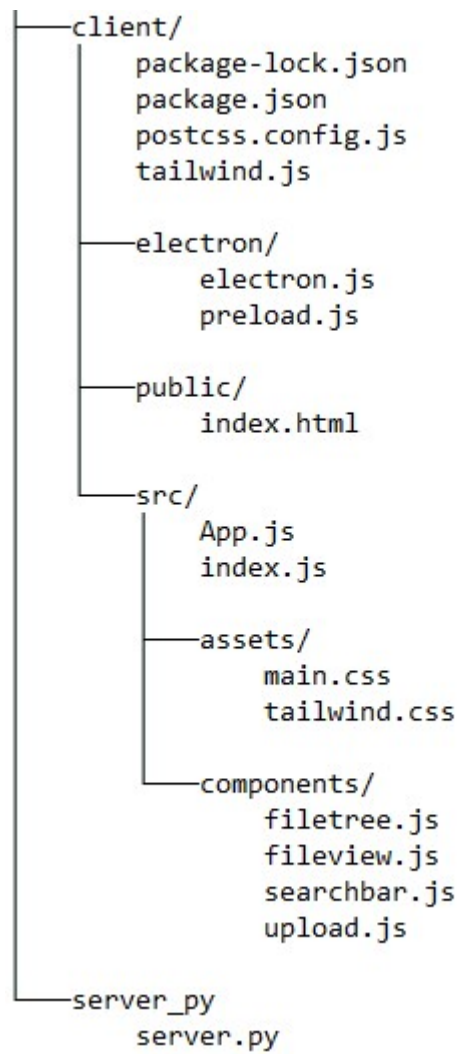
Aplikacja została zbudowana z myślą o zapewnieniu interfejsu użytkownika przy użyciu JavaScript React, serwera HTTP opartego na Flask Python, oraz bazy danych MongoDB, która jest bazą typu NoSQL. Zintegrowana całość jest uruchamiana z wykorzystaniem pakietu Electron, co umożliwia stworzenie aplikacji natywnej, zachowując przy tym funkcje przeglądarki oraz pozwalając na dostęp do zasobów systemowych.

JavaScript React

Jest to biblioteką pozwalającą na budowanie interaktywnych interfejsów użytkownika. Główną koncepcją React są komponenty, czyli samodzielne, hermetyczne jednostki interfejsu. Pozwala to na reakcje na zmiany wywołane przez użytkownika lub sam system i przy tym automatyczne aktualizowanie tych stron. React używa składni JSX (*Javascript XML*), który jest rozszerzeniem składni Javascript. Integruje ona kod Javascript z deklaratywnym opisem struktury interfejsu.

Python Flask

Od wybranego frameworku obsługującego serwer HTTP



Rysunek 1: Uogólniona struktura projektu

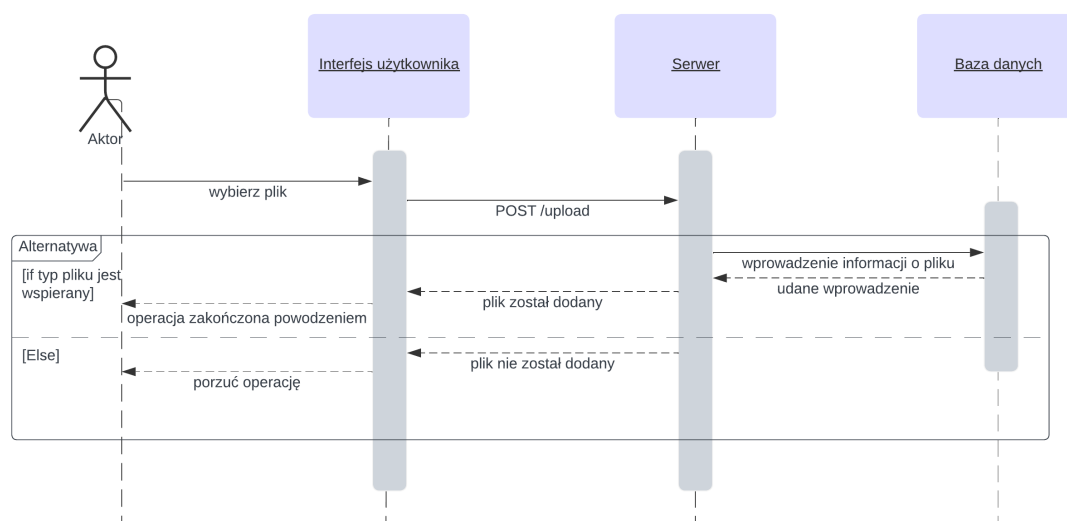
2.3 Architektura systemu

Baza danych

2.4 Scenariusze wykorzystania aplikacji

Aplikacja przyjmuje pliki o rozszerzeniu pdf, txt i csv. Zachowanie systemu przy dodaniu pierwszych dwóch rodzajów plików, zachowuje się podobnie. Skupmy się na plikach pdf.

1. Gdy użytkownik wybierze plik, aplikacja przesyła ścieżkę pliku do serwera,
2. System wyciąga z pliku cały tekst z pomocą biblioteki języka Python *PdfMiner*,
3. Z zawartości zostają wyciągnięte słowa kluczowe. Tworzenie tzw. tagów odbywa się z pomocą biblioteki *yake* oraz funkcji *KeywordExtractor()*. Funkcja ta przyjmuje variables dotyczące języka danego tekstu, maksymalną ilość słów w tagu, ponieważ możemy ustawić ich więcej niż 1, deduplication threshold który definiuje szansę na powtórzenie się słów w różnych tagach (im bliżej 0 tym mniejsze prawdopodobieństwo), oraz oczekiwaną liczbę słów kluczowych.
4. w tym samym czasie z pomocą biblioteki *sumy* oraz funkcji *TextRankSummarizer*, jest tworzone streszczenie metodą ekstraktywną, polegającą na (opis TextRank)
5. jeżeli plik pdf posiada outlines to metoda *.get_outlines()* ekstraktuje je i z wyniku możemy wyciągnąć tytuł dokumentu i nadać plikowi taką nazwę. a jeżeli ich nie ma to tytułem pliku zostaje pierwsze słowo kluczowe
6. ostatecznie wszystkie informacje są zbierane i przesyłane do bazy danych w poniższej formie

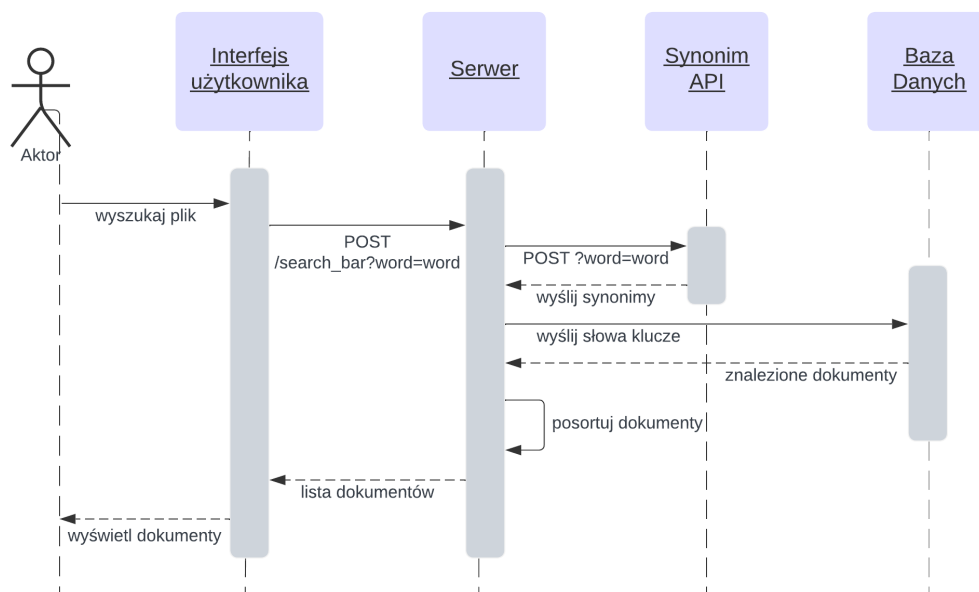


Rysunek 2: Schemat sekwencyjny dodawania pliku

Wyszukiwanie i wyświetlanie plików

Użytkownik ma możliwość wyszukiwania plików po słowach kluczach lub tytule pliku. Tutaj przydatne jest użycie API generującego synonimy dla wyszukiwanego słowa. System zachowuje się w poniżej opisany sposób.

1. aplikacja przesyła komunikat ze słowem wyszukiwanym tzn POST /search_bar,
2. serwer używając API synonimów pobiera 5 najbliższych słów do słowa szukanego przypisując im ranking,
3. serwer przesyła osobne komunikaty do tabeli *file_properties* w naszej bazie danych, zawierające osobno słowo klucz oraz synonimy
4. baza zwraca komunikaty ze znalezionymi dokumentami oraz szukanym słowem, na co server przypisuje im wagi
5. serwer zwraca listę plików użytkownikowi, posortowane zgodnie z rankingiem



Rysunek 3: Schemat sekwencyjny wyszukiwania pliku

2.5 Testy funkcjonalne

2.6 TEMP Idealna aplikacja

Dla każdego pliku aplikacja ma opcje.

3 Możliwości rozwoju i wykorzystania aplikacji

3.1 Przetwarzanie obrazów

3.2 Rozbudowanie funkcjonalności dla plików bazodanowych

3.3 Przejście na wersje web

Podsumowanie i wnioski

Spis rysunków

1	Uogólniona struktura projektu	6
2	Schemat sekwencyjny dodawania pliku	7
3	Schemat sekwencyjny wyszukiwania pliku	8