Obsługa API wolnelektury.pl danych przy wykorzystaniu języka Python oraz bazy danych Oracle lub Microsoft Server

Plan pracy

MR – czyli ja, podpisałem się na wypadek jakbym zapomniał.

Spis treści

[1 Wstęp 3](#_Toc178342331)

[2 Sposób działania 3](#_Toc178342332)

[3 Technologie 4](#_Toc178342333)

[3.1 Model AI oraz NLU 4](#_Toc178342334)

[3.1.1 API ChatGPT 3.5 – podjąć decyzję 4](#_Toc178342335)

[3.2 Języki programowania 4](#_Toc178342336)

[3.3 Biblioteki programistyczne 4](#_Toc178342337)

[3.3.1 psycopg2 4](#_Toc178342338)

[3.3.2 unidecode 4](#_Toc178342339)

[3.3.3 tiktoken 4](#_Toc178342340)

[4 Przewidywane koszty 4](#_Toc178342341)

[4.1 Koszt usługi API ChatGPT 3.5 4](#_Toc178342342)

[5 Plan pracy 5](#_Toc178342343)

[5.1 Wstępne określenie potrzeb projektu 5](#_Toc178342344)

[5.2 Badanie dostępnych technologii oraz oszacowanie kosztów ich wykorzystania 5](#_Toc178342345)

[5.3 Wybór technologii które zostaną wykorzystane w projekcie 5](#_Toc178342346)

[6 Programowanie 5](#_Toc178342347)

[7 Struktura API 5](#_Toc178342348)

[7.1 Przykładowa struktura API wolnelektury.pl: 5](#_Toc178342349)

[8 Struktura bazy danych 5](#_Toc178342350)

[9 Notatki 5](#_Toc178342351)

[9.1 Wybrać przykładowy tekst do testów i do dokumentacji 5](#_Toc178342352)

[9.2 ROZRÓNIĆ PROMPTY ZALEŻNIE OD DŁUGOŚCI TEKSTU – pieprzony Mickiewicz potrzebuje innego prompta niż Pan Poeta Tuwim! 5](#_Toc178342353)

[9.3 Ważne: 5](#_Toc178342354)

[9.4 PostregreSQL ma specjalne typy danych dla JSONów i XMLów 6](#_Toc178342355)

[9.5 Utworzyć klasę odpowiadającą za tworzenie struktury projektu! Nie może to być obsługiwane w pieprzonym download\_book, jak ja w ogóle mogłem coś tak głupiego napisać? 6](#_Toc178342356)

[9.6 Zdecydować czy pobierać do bazy wszystkie dane czy można z jakichś zrezygnować 6](#_Toc178342357)

[9.7 Utworzyć resztę diagramów 6](#_Toc178342358)

[9.8 Przejrzeć funkcje i uprościć je jak tylko się da 6](#_Toc178342359)

[10 Programowanie notatnik plan działania 7](#_Toc178342360)

# Wstęp

Poniżej opisane są wymagania, rozwiązania itd. dot. projektu „Generowanie Streszczeń Tekstów z wolnelektury.pl Przy Pomocy API GPT 3.5”.

Jako przykład używana będzie moja ulubiona książka „Marsjanin”.

# Sposób działania

API GPT 3.5 pozwala na wysłanie jednorazowo zapytania wartego 4096 tokenów. Na wartość zapytania składa się input oraz output, czyli wysłane zapytanie oraz wygenerowana odpowiedź.

Dowolny tekst można podzielić na części i wydać GPT polecenie aby streścił jedynie dany fragment, a następnie połączyć wszystkie teksty w całość (ewentualnie można następnie wykonać kolejne zapytania które zintegrują tekst w spójną całość przy użyciu GPT).

Przykładowe zapytanie:  
„Streść dla mnie tekst wysłany w dokumencie. Nie dodaj żadnych własnych komentarzy ani uwag, jedynie streszczenie. Nie nazywaj tego tekstu rozdziałem, bo nim nie jest.”

# Technologie

## Model AI oraz NLU

### API ChatGPT 3.5 – podjąć decyzję

NLU - Natural Language Understanding  
GPT - Generative Pre-trained Transformer

W ramach projektu rolę sztucznej inteligencji generującej streszczenia na podstawie zapytań, będzie pełnił ChatGPT w wersji 3.5. Przy pomocy języka Python zostanie wykorzystane jego API w celu komunikacji zapytań oraz odbierania odpowiedzi, docelowo zawierających streszczenia wybranych tekstów.

## Języki programowania

#### Python

Za podstawowe operacje oraz komunikację z bazą danych odpowiedzialny ma być program napisany w języku Python.

#### SQL

SQL będzie wykorzystywany w ramach konfiguracji oraz operacji na bazie danych PostgreSQL.

#### HTML/CSS

Z uwagi na niskie wymagania wobec interfejsu projektu, HTML oraz CSS będą wykorzystane aby utworzyć podstawowy interfejs użytkownika.

## Biblioteki programistyczne

Poniżej wymienione biblioteki zostaną wykorzystane w programie, w związku z czym do poprawnego działania będzie wymagana ich instalacja.

### psycopg2

Biblioteka dedykowana integracji PostgreSQL z językiem Python.

### unidecode

Bibliotek została wykorzystana w celach normalizacji danych.

### tiktoken

Biblioteka *tiktoken* jest narzędziem służącym do tokenizacji tekstu w sposób zgodny z formatem używanym przez modele językowe, takie jak GPT.

W tym projekcie biblioteka *tiktoken* zostanie wykorzystana do oszacowania kosztów zapytań wysyłanych do API oraz w celu ograniczenia kosztów pojedynczego zapytania, tak aby zabezpieczyć budżet przed np.: wygenerowaniem bardzo kosztownego zapytania na skutek błędu programu.

# Przewidywane koszty

* Przewidywany budżet projektu wynosi 500 PLN, bo resztę pieniędzy wydałem na mandaty i warunki (Policja i Szanowni Nadludzcy Szkarłatnie Połyskujący na Bordowym Tle Prowadzący – Dziękujemy za wsparcie w rozwoju, w końcu od tego jesteście!).

# Koszt usługi API ChatGPT 3.5

* Koszt 100 000 streszczeń książki wymiarów „20 000 Mil Podwodnej Żeglugi” to około 200 USD czyli około 770 PLN.
* Usługa oferuje możliwość ustawienia maksymalnych kosztów API, co pozwoli uniknąć przekroczenia budżetu przeznaczonego na API.
* Część budżetu: 250 zł
* OCZYWIŚCIE ŻE TAK – OGRANICZYĆ DO 10/20 tyś słów testy i generować pełne streszczenia tylko w ramach prezentacji!
* Koszt wygenerowania jednego streszczenia testowego to 36 groszy, załóżmy że będę generował streszczenie testowe w tak małym wymiarze jak to możliwe – dajmy na to 1 tyś słów – to daje 1/20 tej ceny, mógłbym przeprowadzać 10-ki testów i płacić tylko za pełne streszczenia

# Plan pracy

## Wstępne określenie potrzeb projektu

## Badanie dostępnych technologii oraz oszacowanie kosztów ich wykorzystania

## Wybór technologii które zostaną wykorzystane w projekcie

# Programowanie

# Struktura API

## Przykładowa struktura API wolnelektury.pl:

{

"kind": "Epika",

"full\_sort\_key": "verne jules gabriel$20 000 mil podmorskiej z~1eglugi$2848",

"title": "20 000 mil podmorskiej żeglugi",

"url": "<https://wolnelektury.pl/katalog/lektura/20-000-mil-podmorskiej-zeglugi/>",

"cover\_color": "#961060",

"author": "Jules Gabriel Verne",

"cover": "book/cover/20-000-mil-podmorskiej-zeglugi.jpg",

"epoch": "Pozytywizm",

"href": "<https://wolnelektury.pl/api/books/20-000-mil-podmorskiej-zeglugi/>",

"has\_audio": true,

"genre": "powieść przygodowa",

"simple\_thumb": "<https://wolnelektury.pl/media/book/cover_api_thumb/20-000-mil-podmorskiej-zeglugi_4ZR8kGC.jpg>",

"slug": "20-000-mil-podmorskiej-zeglugi",

"cover\_thumb": "book/cover\_thumb/20-000-mil-podmorskiej-zeglugi\_FqliSrH.jpg",

"liked": null

}

## Struktura API GPT 3.5

# GPT 3.5

## Przykładowe zapytania używane w projekcie i ich geneza

Zapytanie służące do połączenia licznych mniejszych streszczeń w spójną całość:

"Utwórz streszczenie poniższego tekstu. Odpowiedź może być kreatywna i może być rozbudowana.\nTekst:"

To zapytanie pozwala GPT na trochę kreatywności oraz swobody w związku z czym generuje swego rodzaju „luźne interpretacje”.

Zapytanie służące do streszczania dużych ilości tekstu do jak najmniejszych plików aby później małą ilością rekurencji zdjednoczyć je do jednego pliku:

"Utwórz streszczenie poniższego tekstu. "  
 "Odpowiedź ma zawierać jedynie streszcznie, żadnych wstępów lub komentarzy."  
 "\nTekst:")

# Struktura bazy danych

# Notatki

# Wybrać przykładowy tekst do testów i do dokumentacji

## ROZRÓNIĆ PROMPTY ZALEŻNIE OD DŁUGOŚCI TEKSTU – pieprzony Mickiewicz potrzebuje innego prompta niż Pan Poeta Tuwim!

## Ważne:

* Whether your API call works at all, as total tokens must be below the model's maximum limit (4097 tokens for gpt-3.5-turbo)

To może być problematyczne, trzeba utworzyć jakiś sposób podziału na więcej podzapytań.

Dobra, jako że jestem geniuszem to wymyśliłem rozwiązanie:

Hobbit – 80 000 znaków ~ 106 000 tokenów wejściowych

Ograniczenie tokenów dla GPT 3 to 4096 tokenów.  
Możemy podzielić książkę na 25 rozdziałów

## PostregreSQL ma specjalne typy danych dla JSONów i XMLów

## Utworzyć klasę odpowiadającą za tworzenie struktury projektu! Nie może to być obsługiwane w pieprzonym download\_book, jak ja w ogóle mogłem coś tak głupiego napisać?

## Zdecydować czy pobierać do bazy wszystkie dane czy można z jakichś zrezygnować

Potrzebne dane:

* kind
* title
* url
* author
* cover
* epoch
* href
* genre
* simple\_thumb

Nie potrzebuję bazy danych żeby pracować z tyloma danymi, gorzej z zapisem dużej ilości dokumentów.

## Utworzyć resztę diagramów

## Przejrzeć funkcje i uprościć je jak tylko się da

Nowy plan – baza danych nie ma sensu. Program będzie pracował tylko na jednej książce na raz i na ani jednej więcej. Po zakończeniu procesu program będzie usuwał pliki inne niż utworzone streszczenie aby zaoszczędzić na zapytaniach do ChatuGPT.

Założenia:

* Pobranie książki to minimalne zużycie zasobów, które w połączeniu z stałą aktualizacją treści książek oraz oszczędnością przestrzeni dyskowej pozwala zmniejszyć złożoność projektu tym samym zwiększając jego stabilność oraz odporność na błędy.
* OpenAI pozwala na utworzenie limitu wydatków w przeciągu miesiąca co pomoże mi nie zbankrutować.
* Bibliotek tiptoken pozwala sprawdzić ilość tokenów jakie będą wymagane do realizacji zapytania jeszcze zanim zostanie ono wysłane do Open AI.

<https://wolnelektury.pl/api/books/> - link do api

<https://wolnelektury.pl/media/book/txt/makuszynski-o-dwoch-takich-co-ukradli-ksiezyc.txt>

link do dwóhc takich – w ramach testów

# Programowanie notatnik plan działania

Najpierw utworzyć jak najprostszą strukturę i przetestować możliwość kontynuowania rozwoju programu.

Jeśli to się uda to przygotować pełny moduł komunikacji z API – ten moduł ma powstać najpierw i ma być maksymalnie uproszczony:

- Zapytanie ma być ustawione na sztywno w programie, jeśli przyjdzie do upubliczniania zapytania użytkownikowi to będzie to robione na końcu

- Ten moduł ma zwracać jedną rzecz – ciąg znaków będący odpowiedzią na query. Jakiekolwiek serializacje, eksporty, modyfikacje i chuj wie co jeszcze mają się dziać poza tym modułem.

- Do tego modułu dostęp ma mieć tylko main i żaden inny chuj nie ma kierować!

- Po utworzeniu modułu ma on być zamknięty, odsunięty, nietykalny i chuj wie co jeszcze.

Jeśli to się uda to teraz można bawić się z danymi, ETLami, SQLami, bazami danych, relacjami, sposobami przechowywania XMLów i JSONów, książkami offline pobieranymi z dysku etc.

**Najpierw utworzyć działający moduł API – później przygotować całą strukturę – na końcu zintegrować te dwie struktury i utworzyć interfejs użytkownika.**