Input Data Analysis Tool

Credit Suisse

Prowadzący: Dr. Inż. Jan Nikodem

Autorzy: Jędrzej Czykier, Michał Puciłowski, Wiktor Rzońca, Jan Zemło

Politechnika
Wrocławska

1. Opis projektu:

Jeziora danych (*ang. data lakes*) zbierają informacje z wielu różnych źródeł danych w różnych formatach, które w większości nie zawierają informacji o typach danych. Proces dostarczania danych opiera się na Umowie Dostarczania Danych (DDA) między dostawcą a odbiorcą danych. Celem tego dokumentu jest zebranie wymagań funkcjonalnych dotyczących dostarczania danych. Jednym z omawianych problemów jest dostarczanie danych w uzgodnionym schemacie i typie. Celem tego projektu jest napisanie skryptu w języku Python, który sprawdzi, czy dane otrzymane przez Jezioro Danych są zgodne z deklarowanym typem danych w UDD. W przypadku niepowodzenia, oczekiwanym zachowaniem jest powiadomienie o niezgodności typu lub nieprawidłowym pliku danych. Wykorzystanie ChatGPT w celu opracowania możliwych wzorców lub pomysłów, które mogą poprawić jakość danych jest dozwolone. Zespół projektowy musiałby wyjaśnić, w jaki sposób skorzystali z ChatGPT i w jaki sposób narzędzie może pomóc w zadaniach tego typu.

2. Funkcje członków grupy:

Wiktor Rzońca – kontakt z firmą/prowadzącym , pisanie kodu, zarządzanie wersją (GitHub)

Jędrzej Czykier – pisanie kodu, dokumentacja, koordynacja

Michał Puciłowski – pisanie kodu, dokumentacja

Jan Zemło – pisanie kodu, interakcja z ChatGPT, testowanie skryptu

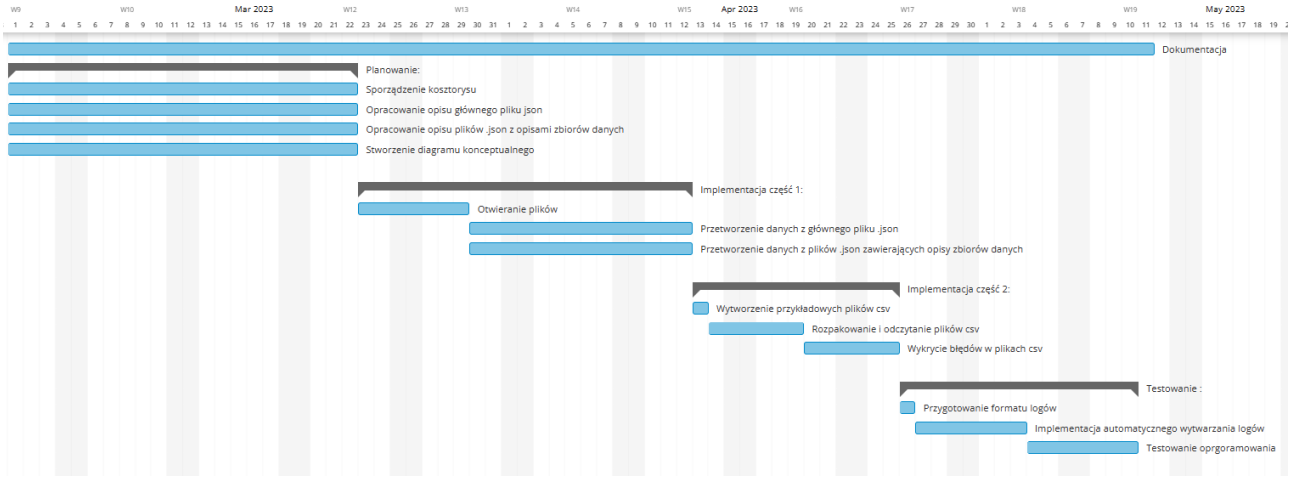
3. Wymagania:

- Program zostanie napisany w języku Python i będzie uruchamiany z linii komend.
- Program musi poprosić użytkownika o lokalizację pliku wejściowego.
- Program przeszukuje pliki w poszukiwaniu głównego pliku DDA (.json), następnie pobiera nazwy oczekiwanych zestawów danych.
- Program będzie szukał poszczególnych plików (.json) z zestawami danych, aby uzyskać oczekiwany schemat, po czym rozpakuje folder (.tar) i przejrzy każdy plik (.csv) zestawu danych i wykryje ewentualne błędy.
- Wynikiem działania programu będzie log z ostrzeżeniami o brakujących zestawach danych i listą ERROR/NO ERROR dla każdego zestawu danych.

4. Plan realizacji projektu:

Co 2 tygodnie będą odbywać się konsultacje z firmą nadzorującą, mające na celu przedstawienie postępu i sprecyzowanie lub nakierowanie toku pracy. Z każdego spotkania tworzony będzie dokument zawierający poruszone kwestie.

Plan realizacji projektu przedstawiony za pomocą wykresu Gantta:



5. Propozycja punktów kontrolnych

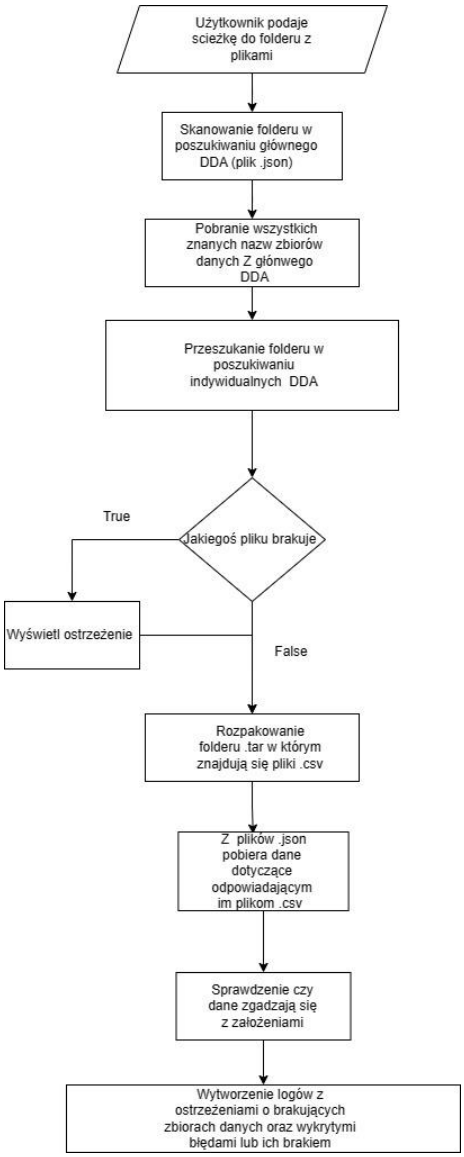
I. 22.03.2023
Do pierwszego punktu kontrolnego zdefiniujemy wymagania oraz założenia projektowe, a także rozdzielimy funkcje członków zespołu. Rozplanujemy pracę nad projektem, co zostanie zaprezentowane przy pomocy wykresu Gantta. Stworzymy diagram konceptualny na zasadzie schematu blokowego przy pomocy którego możliwe będzie proste zapoznanie się z działaniem programu. Opracujemy opis głównego pliku json oraz indywidualnych plików json wraz z opisami zawieranych informacji. Założymy repozytorium na portalu Github.com, aby móc wydajnie zarządzać wersją.

II. 12.04.2023
Do drugiego punktu kontrolnego stworzymy część programu dzięki której w linii komend możliwe będzie podanie ścieżki do pliku i zostanie ona przekazana do programu. Zaimplementujemy także możliwość otwarcia oraz pobrania informacji o zawartości wielu plików o rozszerzeniach csv oraz json.

III. 26.04.2023
Do trzeciego punktu kontrolnego zaimplementujemy część programu, która będzie porównywać dane plików zawartych w plikach z rozszerzeniem csv z danymi plików zawartych w pliku z rozszerzeniem json. Stworzymy także funkcjonalność wykrywania błędów w tych plikach, na razie wyświetlanych tylko w wersji konsolowej.

IV. 10.05.2023
Do czwartego punktu kontrolnego zaimplementujemy część programu odpowiadającą za wypisanie informacji o wyniku działania programu do pliku. Przetestujemy również poprawność działania programu w skrajnych przypadkach oraz stworzymy końcowy raport zawierający całą stworzoną dokumentacje rozszerzoną o opis przeprowadzonych testów, ograniczeń programu oraz naszej pracy z narzędziem ChatGPT.

6. Diagram konceptualny



7. Opis plików json

Pliki DDA (.JSON): Istnieją dwa typy plików JSON:

1. Główny plik JSON:

- zawiera nazwy wszystkich zbiorów danych i ogólne parametry dotyczące tych zbiorów danych, takiej jak np. nazwa, typ, opis, cid.
- każdy zbiór danych zawiera się w {}, kolejne zbiory oddzielone są przecinkiem, wszystkie dane otoczone są cudzysłowami.
- ze zbioru potrzebne nam są nazwy. Nasz skrypt będzie musiał zapisać nazwy pojawiające się przy kluczach „name”.

Przykład – fragment pliku:


```
{
    "name": "firm_contract_type",
    "type": "INTEGER",
    "description": "Firm Contract Type",
    "cid": "D"
  },
  {
    "name": "short_name",
    "type": "STRING",
    "description": "Short Name of the Contract",
    "cid": "D"
  },
```


2. Indywidualny plik JSON jednego zbioru danych.
   - nazwa zbioru danych w nazwie pliku (firm_contract_type.json, short_name.json),
   - zawiera schemat (nazwy kolumn i typy danych)
   - schematy różnią się od siebie w zależności od pliku,
   - niezmienne pozostają zasady budowy pliku JSON które doskonale opisuje Paweł Mansfeld na stronie https://mansfeld.pl/programowanie/json-co-to-jest-czy-warto-uzywac/

8. Kosztorys

Średnie wynagrodzenie dla programistów w języku Python w roku 2023:

www.glassdoor.com

16 120 zł/msc 89 zł/godz

www.erieri.com

16 460 zł/msc 91 zł/godz

www.salaryexpert.com

18 680 zł/msc 102 zł/godz

Średnia arytmetyczna z tych trzech: 94 zł/godz

Kosztorys: 480 (godzin roboczych) x 94 (średnie zarobki na godzinę programisty w języku Python podane w zł) = 45 120 zł

II punkt kontrolny

Po przygotowaniu dokumentacji wstępnej przeszliśmy do implementacji skryptu. Stworzyliśmy w tym celu 5 klas:

- Main.py
- Main_DDa_File.py
- DDAFileValidator.py
- DDA_file.py
- DDA_column.py

Main.py

Główna klasa programu, która odpowiada za uruchomienie całego programu. Łączy ona pozostałe klasy i wywołuje odpowiednie metody w odpowiedniej kolejności w celu przetworzenia danych wejściowych i wygenerowania wyników. Pobiera ona także od użytkownika ścieżkę do plików json, które mają zostać wczytane oraz nazwę głównego pliku json według którego mają one być sprawdzane.

Main_DDA_File

Klasa ma na celu znalezienie głównego pliku DDA, a następnie zwrócenie listy plików DDA znajdujących się w tym samym folderze. Funkcja ***find_dda_file*** szuka głównego pliku DDA, który znajduje się w podanym folderze. Funkcja ***get_individual_dda_files*** odczytuje plik DDA i zwraca listę plików DDA znajdujących się w tym samym folderze co plik główny.

DDA_file

Klasa ma na celu utworzenie słownika kolumn dla każdego pliku DDA. Funkcja ***dictionary*** odczytuje plik DDA i tworzy obiekty klasy **DDA_column**, które reprezentują każdą kolumnę w pliku. Wszystkie obiekty klasy **DDA_column** są przechowywane w słowniku columns klasy **DDA_file**, gdzie kluczem jest nazwa kolumny.

DDA_column

Klasa reprezentuje kolumnę w pliku DDA. Przechowuje informacje o nazwie kolumny, typie danych, formacie czasu, opisie i identyfikatorze kolumny. Każdy obiekt klasy **DDA_column** posiada następujące atrybuty: name (nazwa kolumny), type (typ danych w kolumnie), time_format (format czasu w kolumnie, w przypadku kolumny z datą), description (opis kolumny) oraz cid (identyfikator kolumny, o ile został podany w pliku DDA).

DDAFileValidator

Klasa służy do weryfikacji, czy pliki DDA istnieją w podanym folderze. Funkcja ***validate_files*** odczytuje plik DDA i tworzy oraz zwraca listę plików DDA, gdzie każda krotka składa się z nazwy pliku i wartości logicznej, czy plik istnieje czy nie.