Stworzenie datasetu z prognozą pogodową na obecny dzień dla każdej stolicy na świecie

Jakub Wilczak

Opis problemu

Celem zadania rekrutacyjnego jest stworzenie datasetu, który zawiera prognozę pogody dla każdej stolicy na świecie w momencie skompilowania oprogramowania. Dla realizacji tego zadania wykorzystano 2 zewnętrzne serwisy API. W tym pogodowe https://open-meteo.com, które wymaga szerokości i długości geograficznej, by zwrócić potrzebne dane. Dlatego projekt potrzebuje też usługi API, które zawiera informacje o stolicach i ich współrzędnych geograficznych https://restcountries.com. Dzięki temu najpierw projekt pobierani listę stolic z serwisu restcountries, a następnie na ich podstawie pobiera dane pogodowe.

Serwis API restcountries.com

1. Dane, które udostępnia serwis:

alpha2Code / cca2	Dwuliterowe kody krajów ISO 3166-1 alpha-2
alpha3Code / cca3	Trzyliterowe kody krajów ISO 3166-1 alpha-3
altSpellings	Alternatywne pisownie nazwy kraju
area	Powierzchnia geograficzna
borders	Kraje sąsiadujące
callingCodes / idd	Międzynarodowe numery kierunkowe
capital	Nazwy stolicy
capitalInfo > latlng	Szerokość i długość geograficzna stolicy
car > signs	Międzynarodowe znaki wyróżniające pojazdy (owal)
car > side	Strona, po której obowiązuje ruch drogowy
cioc	Kod Międzynarodowego Komitetu Olimpijskiego
coatOfArms	Linki do obrazów herbu w formacie SVG i PNG na stronie
continents	Lista kontynentów, na których leży kraj
currencies	Lista wszystkich walut
demonym	Mieszkańcy kraju
demonyms (m/f)	Mieszkańcy kraju z uwzględnieniem płci (męskiej/żeńskiej)
independent	Status niepodległości ISO 3166-1 (czy kraj jest suwerennym

fifa	Kod FIFA
flag	v2: Link do flagi SVG na Flagpedia, v3: emoji flagi
flags	Linki do flag SVG i PNG na Flagpedia
gini	Wskaźnik Giniego Banku Światowego
landlocked	Kraj bez dostępu do morza
languages	Lista języków urzędowych
latlng	Szerokość i długość geograficzna
maps	Linki do Map Google i OpenStreetMap
name	Nazwa kraju
name > official/common	Oficjalna i powszechna nazwa kraju
nativeName	Rodzima nazwa kraju
nativeName >	Oficjalna i powszechna rodzima nazwa kraju
numericCode / ccn3	Kod numeryczny ISO 3166-1 (ONZ M49)
population	Populacja kraju
postalCodes > format/regex	Format i wyrażenie regularne kodów pocztowych kraju
region	Regiony demograficzne ONZ
regionalBlocs	Bloki regionalne
startOfWeek	Dzień, od którego zaczyna się tydzień (niedziela/poniedziałek/sobota)
status	Status przypisania ISO 3166-1
subregion	Subregiony demograficzne ONZ
timezones	Strefy czasowe
topLevelDomain / tld	Internetowe domeny najwyższego poziomu
translations	Lista tłumaczeń nazwy kraju
unMember	Status członkostwa w ONZ

2. Dane potrzebne do realizacji:

Name	Nazwa kraju
Capital	Nazwa stolicy
CapitalInfo	Szerokość i długość geograficzna stolicy

Głównym celem tego serwisu API jest pobranie danych, które będą wykorzystane w celu wygenerowania zapytań API do usługi Open-Meteo, a następnie zostaną uwzględnione w kompletnym datasecie. Dlatego większość danych jest niepotrzebna.

3. Sposób pobrania danych

1. Implementacja klasy pomocniczej do przyszłego mapowania odpowiedzi z REST API na listę obiektów.

```python
class CapitalCity:  2 usages  new *
    def __init__(self, countryName, capitalName, latitude, longitude):  new *
        self._countryName = countryName
        self._capitalName = capitalName
        self._latitude = latitude
        self._longitude = longitude

    @property  new *
    def countryName(self):
        return self._countryName

    @property  4 usages  new *
    def capital_name(self):
        return self._capitalName

    @property  1 usage  new *
    def latitude(self):
        return self._latitude

    @property  1 usage  new *
    def longitude(self):
        return self._longitude
```

2. Stworzenie funkcji, która wysyła zapytanie do serwera REST API w celu pozyskania informacji o nazwie kraju, stolicy, szerokości i długości geograficznej. W przypadku problemu z zapytaniem zostanie zwrócony błąd.

```python
def getCountries():  1 usage  new *
    try:
        countries_url = 'https://restcountries.com/v3.1/all'
        response = requests.get(countries_url)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as err:
        print(err)
        logging.warning(f'RequestException City request failed: {err}')
```

4. Sposób przetwarzania danych z pierwszego serwisu.

Mapowanie odpowiedzi z REST API na listę obiektów w celu wykonania zapytań w serwisie API Open-Meteo.
1. Sprawdzenie, czy w krajach istnieją odpowiednie kolumny i czy mają one przypisane wartości.
2. Wykryte stolice z zapytania REST zostają dodane do listy obiektów na przyszłe wykorzystanie.
3. Serwis RestCountries zwraca kraj Nauru, który nie posiada stolicy, kontynent Antarktydę oraz niezależne wyspy. Dla tych danych wykonywana jest specjalna logika, która filtruje je filtruje przed zwróceniem błędu. Gdyż jest to działanie zamierzone.
4. Program zwraca błąd, jeżeli serwis RestCountries nie zwróciło danych w zapytaniu lub jeżeli pojawi się kraj, który nie ma odpowiednich danych. W dodatku jeżeli jest kraj, który nie ma stolicy to zostanie to umieszczone w logach.

```
capitalList = []
try:
    CountriesData = getCountries()
    if not CountriesData:
        raise noDataFromApiException("No data pulled from the Countries API")

    for country in CountriesData:
        try:
            if ('name' in country and country['name'] and 'common' in country['name'] and country['name']['common']
                    and 'capital' in country and country['capital'] and 'capitalInfo' in country
                    and 'latlng' in country['capitalInfo'] and len(country['capitalInfo']['latlng']) == 2):

                capitalList.append(
                    CapitalCity(country['name']['common'], country['capital'][0], country['capitalInfo']['latlng'][0],
                                country['capitalInfo']['latlng'][1]))
            else:
                if 'name' in country and country['name']:
                    if country["name"] == "Antarctic" or country["name"] == "Nauru" or country["name"] == "Antarctica"\
                            or country["name"] == "Macau" or country["name"] == "United States Minor OutlyingIslands"\
                            or country["name"] == "Heard Island and McDonald Islands" or country["name"] == "BouvetIsland":
                        pass
                    else:
                        logging.warning(f"Could not extract capital information for {country['name']['common']}")

        except KeyError as err:
            logging.warning(f'Key Error Missing key in capital data {err}')
            print(err)

except noDataFromApiException as error:
    logging.warning(f'NoDataFromApiException Missing Country Api Data{error}')

```

Serwis API Open-Meteo

1. Dane, które udostępnia serwis

Weather code	Kod pogodowy
Generationtime_ms	Czas wykorzystany na wykonanie zapytania
Utc_offset_seconds	Wartość przesunięcia UTC
Timezone	Strefa Czasowa
Timezeon_abbreviation	Skrót od strefy czasowej
elevation	Wysokość nad poziomem morza
Daily_units	Lista jednostek pomiarów do zapytania
Maximum Temperature (2 m)	Maksymalna temperatura (2 m)
Minimum Temperature (2 m)	Minimalna temperatura (2 m)
Maximum Apparent Temperature (2 m)	Maksymalna temperatura odczuwalna (2 m)
Minimum Apparent Temperature (2 m)	Minimalna temperatura odczuwalna (2 m)
Sunrise	Wschód słońca
Sunset	Zachód słońca
Daylight Duration	Długość dnia
Sunshine Duration	Czas trwania nasłonecznienia
UV Index	Indeks UV
UV Index Clear Sky	Indeks UV przy czystym niebie
Rain Sum	Suma opadów deszczu
Showers Sum	Suma przelotnych opadów deszczu
Snowfall Sum	Suma opadów śniegu
Precipitation Sum	Suma opadów atmosferycznych
Precipitation Hours	Liczba godzin z opadami
Precipitation Probability Max	Maksymalne prawdopodobieństwo opadów
Maximum Wind Speed (10 m)	Maksymalna prędkość wiatru (10 m)
Maximum Wind Gusts (10 m)	Maksymalne porywy wiatru (10 m)
Dominant Wind Direction (10 m)	Dominujący kierunek wiatru (10 m)
Shortwave Radiation Sum	Suma promieniowania krótkofalowego
Reference Evapotranspiration (ET_0)	Ewapotranspiracja referencyjna (ET_0)

2. Dane potrzebne do realizacji:

Weather code	Kod pogodowy
Maximum Temperature (2 m)	Maksymalna temperatura (2 m)
Minimum Temperature (2 m)	Minimalna temperatura (2 m)

Maximum Apparent Temperature (2 m)	Maksymalna temperatura odczuwalna (2 m)
Minimum Apparent Temperature (2 m)	Minimalna temperatura odczuwalna (2 m)
Sunrise	Wschód słońca
Sunset	Zachód słońca
Daylight Duration	Długość dnia
Sunshine Duration	Czas trwania nasłonecznienia
UV Index	Indeks UV
UV Index Clear Sky	Indeks UV przy czystym niebie
Rain Sum	Suma opadów deszczu
Showers Sum	Suma przelotnych opadów deszczu
Snowfall Sum	Suma opadów śniegu
Precipitation Sum	Suma opadów atmosferycznych
Precipitation Probability Max	Maksymalne prawdopodobieństwo opadów
Maximum Wind Speed (10 m)	Maksymalna prędkość wiatru (10 m)
Maximum Wind Gusts (10 m)	Maksymalne porywy wiatru (10 m)
Dominant Wind Direction (10 m)	Dominujący kierunek wiatru (10 m)

3. Dane usunięte

Generationtime_ms	Czas potrzebny na generację zapytania
Daily_units	Lista jednostek pomiarów do zapytania
Timezeon_abbreviation	Skrót od strefy czasowej
elevation	Wysokość nad poziomem morza

3. Powód filtracji danych

Te dane zostały usunięte z dwóch powodów.

1. Dane nie wnoszące niczego do datasetu:
    - Generationtime_ms,
    - Timezone_abbreviation – redundantne dane, gdyż jest to kopia kolumny Timezone,
    - elevation – wysokość nad poziomem morza nie zmienia prognozy wykonanej przez centra meteorologiczne, więc te dane są niepotrzebne.
2. Dane w postaci listy, której nie można dodać do datasetu – Daily_units.

4. Sposób pobrania danych

Program sprawdza, czy lista stolic zawiera dane. W przypadku pustej listy zgłaszany jest wyjątek noDataFromApiException. Następnie dla każdej stolicy wykonywane jest zapytanie z informacją zwrotną o prognozie pogody dla wybranej lokalizacji. Po wykonaniu zapytania do zwróconego JSON-a dodawana są 2 kolumny z informacjami o nazwie kraju i stolicy. Postać zwrotna zapytania REST API Open-Meto to dataset wraz z listą danych prognozowych dla wybranych współrzędnych geograficznych. Z tego powodu należy te dane wyodrębnić i umieścić w dedykowanych kolumnach. Ważny jest też tutaj obsługa wyjątku request.exceptions.RequestException gdyż może wystąpić błąd przez zbyt dużą ilość zapytań w krótkim okresie czasowym lub regularny problem z połączeniem lub odpowiedzią z serwera z negatywnym kodem statusu HTTP.

```
    weatherJsonList = []
    try:
        if not capitalList:
            raise noDataFromApiException("Missing Data from capitalList that was pulled from the Countries API")

        for capital in capitalList:
            time.sleep(0.15) # to avoid too many requests error
            weatherJsonList.append(getCityWeatherHistory(capital))
            print(f'added weather info for {capital.capital_name}')

        weatherJsonList = removeWeatherKeysFromJsonFile(weatherJsonList)

    except missingRequiredDataException as error:
        print(error)
        logging.warning(f'Missing Required Data Exception: {error}')
    except noDataFromApiException as error:
        print(error)
        logging.warning(f'noDataFromApiException Missing Data{error}')
```

```
def getCityWeatherHistory(city:CapitalCity):  1 usage  new *
    try:
        weather_url = f'https://api.open-meteo.com/v1/forecast?latitude={city.latitude}&longitude={city.longitude}&fo
        response = requests.get(weather_url)
        response.raise_for_status()
        json_data = response.json()
        json_data["country_name"] = city.countryName
        json_data["capital_name"] = city.capital_name

        daily_index = 0 # Even while using timezones with API, at certain hours the user receives dataset for the cur
        if len(json_data["daily"]["weather_code"])>1:
            daily_index = 1

        json_data["weather_code"]=json_data["daily"]["weather_code"][daily_index]
        json_data["temperature_2m_max"]=json_data["daily"]["temperature_2m_max"][daily_index]
        json_data["temperature_2m_min"]=json_data["daily"]["temperature_2m_min"][daily_index]
        json_data["apparent_temperature_max"]=json_data["daily"]["apparent_temperature_max"][daily_index]
        json_data["apparent_temperature_min"]=json_data["daily"]["apparent_temperature_min"][daily_index]
        json_data["sunrise"]=json_data["daily"]["sunrise"][daily_index]
        json_data["sunset"]=json_data["daily"]["sunset"][daily_index]
        json_data["daylight_duration"]=json_data["daily"]["daylight_duration"][daily_index]
        json_data["sunshine_duration"]=json_data["daily"]["sunshine_duration"][daily_index]
        json_data["uv_index_max"]=json_data["daily"]["uv_index_max"][daily_index]
        json_data["uv_index_clear_sky_max"]=json_data["daily"]["uv_index_clear_sky_max"][daily_index]
        json_data["rain_sum"]=json_data["daily"]["rain_sum"][daily_index]
        json_data["showers_sum"]=json_data["daily"]["showers_sum"][daily_index]
        json_data["snowfall_sum"]=json_data["daily"]["snowfall_sum"][daily_index]
        json_data["precipitation_sum"]=json_data["daily"]["precipitation_sum"][daily_index]
        json_data["precipitation_hours"]=json_data["daily"]["precipitation_hours"][daily_index]
        json_data["precipitation_probability_max"]=json_data["daily"]["precipitation_probability_max"][daily_index]
        json_data["wind_speed_10m_max"]=json_data["daily"]["wind_speed_10m_max"][daily_index]
        json_data["wind_gusts_10m_max"]=json_data["daily"]["wind_gusts_10m_max"][daily_index]
        json_data["wind_direction_10m_dominant"]=json_data["daily"]["wind_direction_10m_dominant"][daily_index]
        json_data["shortwave_radiation_sum"]=json_data["daily"]["shortwave_radiation_sum"][daily_index]
        json_data["et0_fao_evapotranspiration"]=json_data["daily"]["et0_fao_evapotranspiration"][daily_index]

        return json_data
    except requests.exceptions.RequestException as err:
        print(err)
        logging.warning(f'Request Exception Weather request failed: {err}')
```

Następnie należy dane przetworzyć i usunąć niepotrzebne kolumny. Jeżeli w przyjmowanych danych lista będzie pusta to wystąpi obsługa wyjątku missingRequiredDataException, a jeżeli program spróbuje usunąć nieodpowiednią kolumną to zostanie obsłużony wyjątek TypeError.

```python
def removeWeatherKeysFromJsonFile(weatherlist):  1 usage  new *
    temp = weatherlist.copy()
    weatherlist.clear()

    if not temp:
        raise missingRequiredDataException('No data found in provided weather List')

    try:
        for jsonfile in temp:
            prepared_keys_to_delete = []  # to avoid the error of deleting dictionary rows while iterating through it
            if "et0_fao_evapotranspiration" in jsonfile:
                prepared_keys_to_delete.append("et0_fao_evapotranspiration")
            if "timezone_abbreviation" in jsonfile:
                prepared_keys_to_delete.append("timezone_abbreviation")
            if "elevation" in jsonfile:
                prepared_keys_to_delete.append("elevation")
            if "daily_units" in jsonfile:
                prepared_keys_to_delete.append("daily_units")
            if "daily" in jsonfile:
                prepared_keys_to_delete.append("daily")
            for prepared_key in prepared_keys_to_delete:
                del jsonfile[prepared_key]

            weatherlist.append(jsonfile)

        return weatherlist
    except TypeError as err:
        print(err)
        logging.warning(f'Type Error Required data in weather List was not found: {err}')
        raise missingRequiredDataException("Required data in weather List was not found")
```

Zapisanie do pliku .csv

Program połączy wszystkie pliki JSON w jeden Dataframe z biblioteki pandas. Jeżeli w liście jest tylko jeden element bez podanego indexu to wystąpi obsługa wyjątku ValueError. Wyjątek IOError zostanie obsłużony jeżeli użytkownik nie będzie miał możliwości stworzenia pliku, dysk komputera będzie miał za mało pamięci lub plik będzie używany przez inny proces. Wyjątek UnicodeEnodeError wystąpi jeżeli domyślny lub jawnie określony enkoder spróbuje zakodować znaki, których nie obsługuje

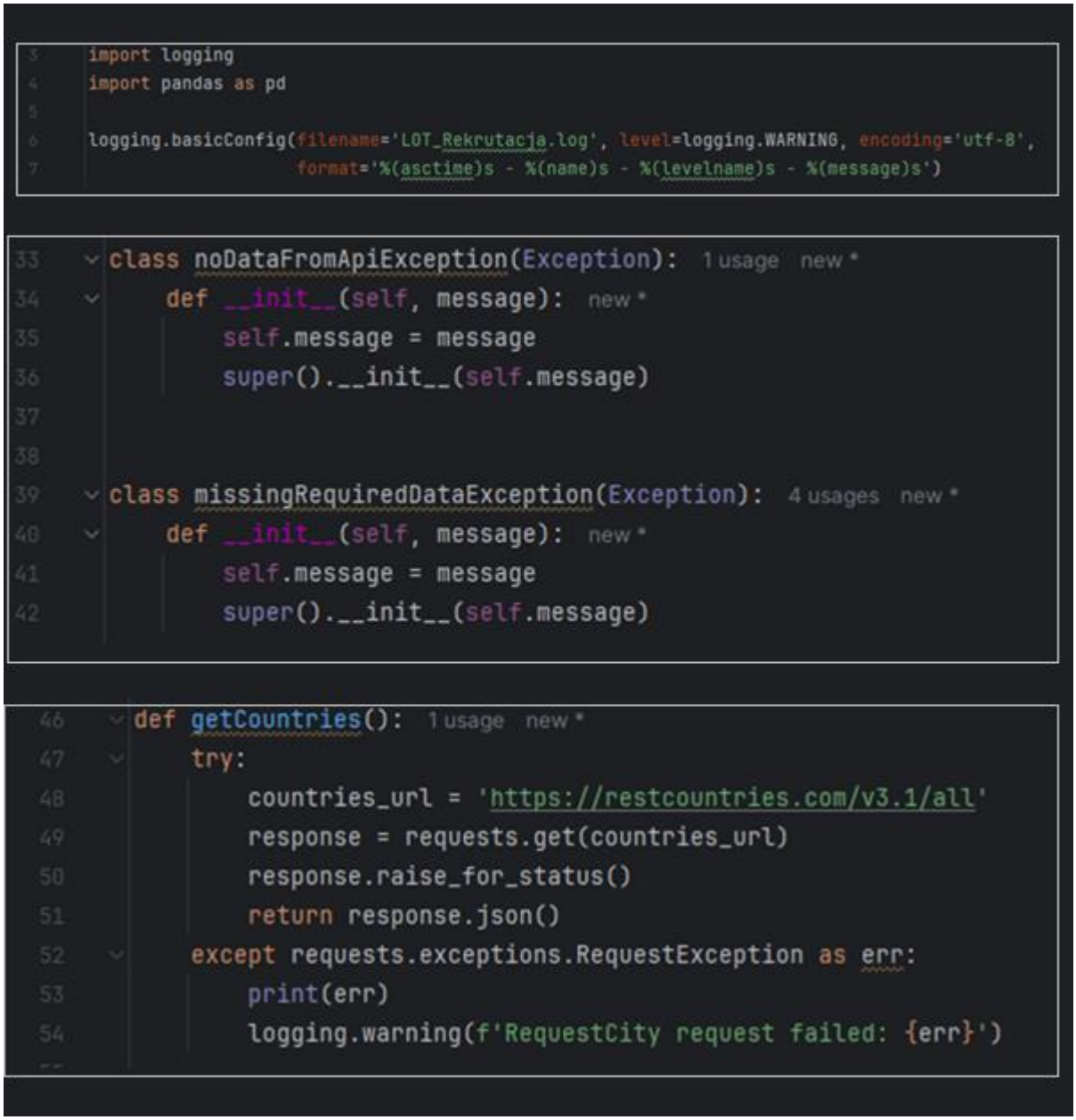
```python
# Saving to CSV
try:
    df = pd.DataFrame(weatherJsonList)
    df.to_csv( path_or_buf: 'weather.csv', index=False, encoding='utf-8')

except ValueError as valueError: #for example only one element in list without providing index while creating dataframe in pandas
    print(valueError)
    logging.warning(f'ValueError Creating DataframeError{valueError}')
except IOError as ioError: #saving to csv file.
    print(ioError)
    logging.warning(f'IOError {ioError}')
except UnicodeEncodeError as unicodeError: #if the data contains characters than cannot be encoded with default encoder
    print(unicodeError)
    logging.warning(f'UnicodeEncodeError {unicodeError}')
```

Dodatkowe Informacje

1. Sposób obsługi błędów

System posiada funkcjonalność logowania błędów w systemie poprzez wypisywanie ich do specjalnego pliku o nazwie LOT_Rekrutacja.log. W dodatku zostały stworzony dwie klasy Exception :
1.noDataFromApiException- w przypadku braku danych dostarczonych z API.
2.missingRequiredDataException -w przypadku braku danych do dalszej operacji systemu.



```
import logging
import pandas as pd

logging.basicConfig(filename='LOT_Rekrutacja.log', level=logging.WARNING, encoding='utf-8',
                    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

```
class noDataFromApiException(Exception):  1 usage  new *
    def __init__(self, message):  new *
        self.message = message
        super().__init__(self.message)


class missingRequiredDataException(Exception):  4 usages  new *
    def __init__(self, message):  new *
        self.message = message
        super().__init__(self.message)
```

```
def getCountries():  1 usage  new *
    try:
        countries_url = 'https://restcountries.com/v3.1/all'
        response = requests.get(countries_url)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as err:
        print(err)
        logging.warning(f'RequestCity request failed: {err}')
```