Specyfikacja implementacyjna – aplikacja Python-19 ilustrująca dane dotyczące przebiegu epidemii Sars-CoV-2

Zespół:

- Wiktor Kępiński
- Bartosz Staszyński
- Aleksander Wyżykowski

Wstęp teoretyczny - opis programu

Poniższa specyfikacja dotyczy programu Python-19, którego dotyczy dokument "Specyfikacja_funkcjonalna_Python-19.pdf". We wspomnianym pliku znajduje się specyfikacja funkcjonalna programu i przedstawienie jego celów. Program ma za zadanie interaktywne wyświetlanie danych dotyczących przebiegu epidemii koronawirusa, w wygodny dla użytkownika sposób.

Informacje ogólne

Do tworzenia oprogramowania korzystaliśmy ze środowiska PyCharm. Kod pisany był w języku Python wersja 3.9.
Program będzie opierał się przede wszystkim na korzystaniu z bibliotek PyQt5 oraz Matplotlib. Struktura programu zostanie podzielona na moduły oraz odpowiednie klasy, które będą odpowiadać za swoją część działania aplikacji.

Podział na moduły

Projekt został podzielony na 4 moduły:

- **window.py** - moduł, który będzie organizował wygląd naszego okna wraz ze wszystkimi jego elementami.
- **tabs.py** - moduł, który będzie generował dwie zakładki: "Zachorowania" i "Ozdrowienia"
- **buttons.py** - moduł, który będzie odpowiadał za przyciski i ich funkcje.
- **slider.py** - będzie zawierał klasę suwaka do wyboru zakresu dat.
- **countries_list.py** - moduł, który będzie odpowiadał za okienko obsługi listy krajów wraz z wyszukiwarką.

Wszystkie moduły zaznaczone kolorem **czerwonym** będą należały do pakietu GUI.

- **main.py** - moduł odpowiedzialny za realizację głównego scenariusza działania programu, tutaj będzie znajdywała się funkcja main().
- **data.py** - moduł zawierający dwie klasy: Parameters (parametry ustawione przez użytkownika) oraz Data (dane do wyświetlenia na wykresie)
- **updates.py** - moduł, który będzie pobierał dane z serwera i aktualizował pliki .csv z danymi oraz pobiera wybrane przez użytkownika parametry do zmiennej klasy Parameters.
- **graph.py** - moduł, który będzie pobierał odpowiednie dane do obiektu klasy Data i wyświetlał je na wykresie z wykorzystaniem biblioteki matplotlib.
- **report.py** - moduł, który będzie odpowiadał za generowanie pliku .pdf z danymi
- **exceptions.py** - moduł odpowiedzialny za obsługę błędów.

Wszystkie moduły zaznaczone kolorem **czarnym** będą należały do pakietu Backend.

Diagram klas i ich krótki opis

W celu przejrzystości kodu i zachowania zasad programowania, moduły zostały podzielone na klasy, które realizują różne zadania. Na końcu dokumentu załączony został diagram klas, a poniżej znajduje się krótki opis poszczególnych klas w każdym module:

window.py klasy:

- Window(QWidget) - klasa, która stworzy główne okno i doda do niego zakładki.

tabs.py klasy:

- TabsWidget(QWidget) - klasa, która stworzy dwie zakładki “Zakażenia” i “Ozdrowienia” i stworzy ich layout.

buttons.py klasy:

- PushButton(QPushButton) - klasa, która stworzy przycisk.
- UpdateButton(PushButton) - klasa, która stworzy przycisk do aktualizowania danych i nada mu funkcję po kliknięciu w kreatorze (realizowaną przez moduł updates.py)
- ReportButton(PushButton) - klasa, która stworzy przycisk do tworzenia raportu .pdf i nada mu funkcję po kliknięciu w kreatorze (realizowaną przez moduł report.py)
- Checkbox(QCheckBox) - klasa, która stworzy checkboxy do listy krajów.

slider.py klasy:

- Slider() – klasa, która stworzy suwak.

countries_list.py klasy:

- Searchbar(QLineEdit) - tworzy pasek wyszukiwania oraz jego funkcję do obsługi listy krajów.
- ListWidget(QScrollArea) - tworzy listę krajów razem z checkboxami i paskiem do przesuwania listy (scroll).

data.py klasy:

- Parameters() - klasa zawierająca aktualne parametry ustawione przez użytkownika, ma metody dostępwe.
- Graph_data() - klasa zawierająca dane do wyświetlenia na wykresie, posiada metody dostępowe.

updates.py klasy:

- Read_parameters() - pobiera wszystkie parametry ustawione przez użytkownika tj. wybrana zakładka, wybrane kraje, wybrany zakres czasowy i zapisuje je w przekazanym obiekcie klasy Parameters poprzez metody dostępowe typu set/change.
- Update_web_data() - pobiera dane z serwera i zapisuje je do plików .csv w lokalizacji programu.
- Update_graph_data() - pobiera wszystkie dane z plików i wybiera te, których zarządał użytkownik (na podstawie parametrów z klasy Parameters) i zapisuje je w obiekcie klasy Graph_data poprzez jej metody dostępowe.

graph.py klasy:

- GraphWidget(FigureCanvas) - tworzy wykres z odpowiednimi danymi z klasy Graph_data().

report.py klasy:

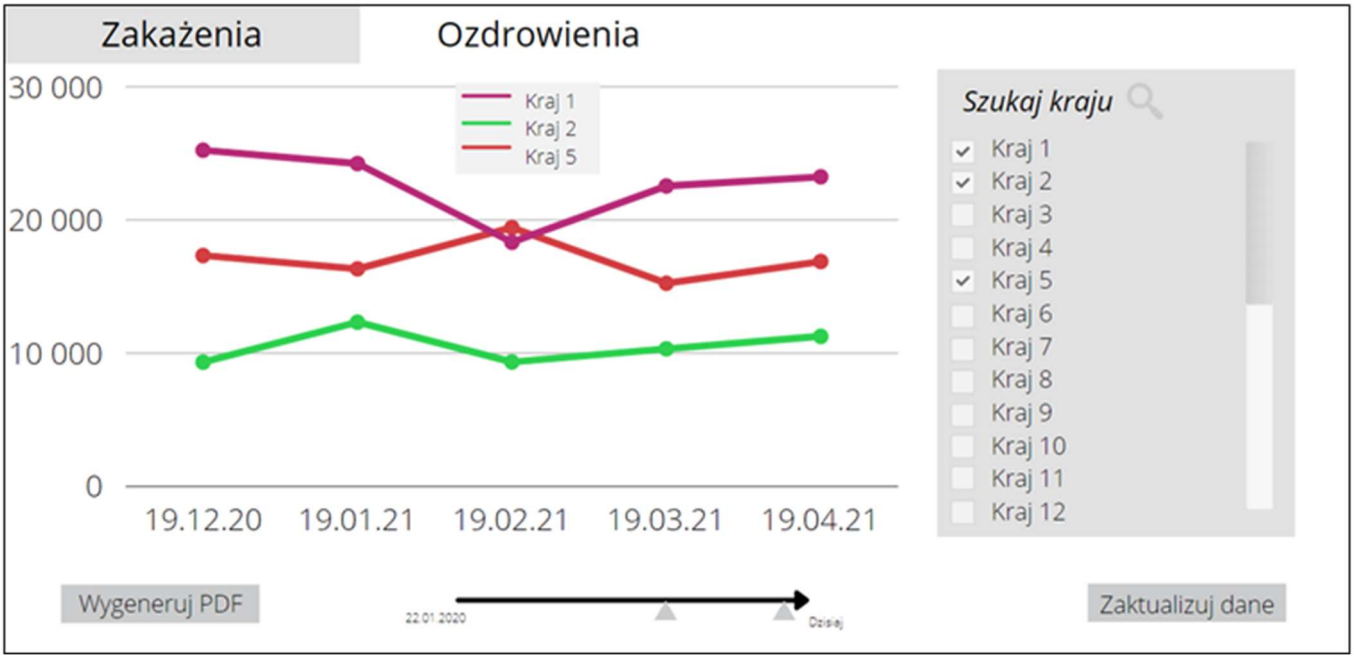
- Report() - tworzy raport do pliku .pdf z wykresem i tabelą danych z klasy Graph_data().

exceptions.py klasy:

- UnableToGenerateReportException(Exception) - klasa obsługująca błąd “Nie można wygenerować raportu, nie wybrano żadnych krajów.”
- UnableToDownloadNewestData(Exception) - klasa obsługująca błąd „Nie można pobrać najnowszych danych. Spróbuj ponownie później.”
- UnableToAddMoreCountriesToGraph(Exception) - klasa obsługująca błąd "Nie można dodać więcej krajów do wyświetlenia.”

Implementacja interfejsu graficznego

Do stworzenia interfejsu graficznego będziemy korzystać z funkcji biblioteki PyQt5 przy pomocy modułów w pakiecie GUI. Będzie on realizowany przez klasy opisane powyżej kolorem **czerwonym.**



Uwagi

Powyższa specyfikacja funkcjonalna ma na celu zarys organizacji projektu, sposobu jego realizacji. Niektóre (bardziej konkretne) kwestie dotyczące sposobu implementacji funkcji programu mogą ulec zmianie. W związku z tym będzie prowadzony raport historii zmian, który zostanie upubliczniony jako załącznik do powyższej specyfikacji przy finalnym oddaniu projektu.

Niektóre klasy zostały opisane na niższym poziomie szczegółowości, ponieważ wymagają one od nas głębszego dopracowania. Ich metody nie zostały jeszcze dostatecznie doprecyzowane, dlatego zostaną one opisane dokładniej przy finalnym oddaniu projektu.