**DOKUMENTACJA**

TYTUŁ PROJEKTU: SYSTEMY KOLEJKOWE

PRZEDMIOT: PODSTAWY INFORMATYKI I PROGRAMOWANIA

Spis treści:

1. Założenia projektowe
2. Opis implementacji
3. Podsumowanie pracy nad projektem
4. Zaimportowane biblioteki

ZAŁOŻENIA PROJEKTOWE

Celem tego projektu była staranna i przejrzysta prezentacja danych ze strony (link: [https://api.um.warszawa.pl/#](https://api.um.warszawa.pl/)) w spójny sposób. W celu osiągnięcia tego utworzono interfejs użytkownika, który umożliwia śledzenie bieżącego stanu kolejki w wybranej instytucji.

OPIS IMPLEMETACJI

Program składa się z pięciu plików. Każdy plik odpowiada za osobne działanie podczas pracy programu. Taki podział jest czytelny i pozwala szybko zlokalizować ewentualne błędy.

Program powinien zostać uruchomiony z pliku *gui.py*. W tym pliku tworzy się interfejs użytkownika, w którym użytkownik wybiera instytucję, a następnie naciskając przycisk „**Zatwierdź**” go zatwierdza. Program przełącza się na plik *queue\_system.py* i wysyła żądania do API, aby uzyskać listę wszystkich spraw urzędowych możliwych do załatwienia w wybranej jednostce. Następnie powraca do pliku *gui.py* i czeka, aż użytkownik wybierze sprawę. Po podjęciu decyzji przez użytkownika przechodzi do pliku *queue\_system.py*, aby uzyskać informacje o bieżącej kolejce. Używkownik po wybraniu sprawy może wyświetlić wykres demonstrujący ilość osób oczekujących w kolejce do monitorowanej na interfejsie sprawy. Jeżeli użytkownik zdecyduje się na wyświetlenie program przechodzi do pliku *queue\_plot.py* i wywołuje metodę tworzącą wykres. Powyższe operacje są przechodzone ponownie, jeśli w trakcie działania programu zmienił się wybór użytkownika.

W pliku *authorities\_dict.py* znajduje się słownik, w którym każda instytucja ma przypisany identyfikator. Po dodaniu do podstawowego adresu URL identyfikatora ze słownika *authorities* program może pobrać prawidłowe dane dla danego urzędu poprzez API.

Plik *queue\_system.py* zawiera klasę *Queue*, która ma metody takie jak: *get\_id ()*, *list\_of\_institutions ()*, *actual\_queue ()*. Wywołując te metody program jest w stanie sformułować dokładny adres URL w celu otrzymania danych z API. Ponadto klasa *Queue* zawiera metody, które zapisują dane, które zostają wyświetlane na graficznym interfejsie użytkownika do list. Ten plik importuje bibliotekę *requests* i słownik *authorities* z *authorities\_dict.py*.

Plik *queue\_plot.py* zawiera klasę *QueuePlot*, która tworzy wykres przedstawiający bieżącą liczbę osób w kolejce dla wybranej sprawy w wybranej instytucji. Ta klasa ma kilka atrybutów, na przykład: *institution*, *case* i *number*, które są przesyłane z interfejsu użytkownika, a także *number\_of\_people* i *x\_data*, które są listami i przechowują dane w celu utworzenia wykresu. Każde wywołanie metody *get\_chosen\_institution* powoduje operację, która dopisuje do atrybutu *number\_of\_ people* bieżącą liczbę osób w kolejce i do atrybutu *x\_data* bieżący czas, sprawdzając parametry funkcji *get\_chosen\_institution*, czy w trakcie tej operacji pozostały niezmienione. W przeciwnym razie zawartość atrybutów *number\_of\_people* i *dane\_x* jest usuwana. Klasa *QueuePlot* zawiera również metodę, która umożliwia wyświetlanie wykresu i jego legendy. Ten plik importuje bibliotekę *matplotlib.pyplot* i bibliotekę *datetime*.

Plik *gui.py* odpowiada za stworzenie graficznego interfejsu użytkownika. Zawiera klasę *Application*, która dziedziczy po klasie *Frame* z biblioteki *tkinter*. Ten plik łączy inne wcześniej wymienione pliki, dlatego ma wiele metod, na przykład: metodę *create\_widgets*, która tworzy i projektuje interfejs użytkownika. Co więcej, klasa *Application* ma pewne metody zdarzeń, które pozwalają widżetom na interakcję z innymi. Ten plik importuje słownik *authorities* z *authorities\_dict.py*, klasę *Queue* z *queue\_system.py* i klasę *QueuePlot* z *queue\_plot.py*.

Ostatni plik to *test\_queue\_system.py* zawiera testy jednostkowe kilku funkcji z *queue\_system.py*, *queue\_plot.py*. Importuje bibliotekę *tkinter*, *authorities* z *authorities\_dict.py*, *Queue* z *queue\_system.py*, *QueuePlot* z *queue\_plot.py* i *Application* z *gui.py*.

PODSUMOWANIE PRACY NAD PROJEKTEM

Podczas tworzenia kodu spełniającego wymagania projektu napotkałam pewne trudności, na początku brak listy gromadzącej identyfikatory wszystkich instytucji oraz brak dokumentacji do bazy danych ze strony. Aby rozwiązać ten problem, utworzyłam zagnieżdżony słownik w pliku *aut\_dict.py*. Takie rozwiązanie pozwala pobierać dane poprzez API dla konkretnego urzędu.

W *queue\_system.py* stworzyłem klasę, której metody pozwalają na wybranie potrzebnych do programu działania danych ze słownika json API.

Uświadomiłam sobie, że niektóre instytucje nie mają danych, ich odpowiedź z API jest pusta. Problem ten został rozwiązany przy pomocy konstrukcji *try:… catch TypeError:…*, która wyświetla komunikat „Brak danych” w polu do wpisania wyboru, ilekroć użytkownik wybierze ten typ instytucji.

Wybrałam graficzny interfejs użytkownika do prezentacji danych i obsługi żądań użytkownika, ponieważ jest intuicyjny w obsłudze. Po kilku próbach umieszczenia wszystkich widżetów we właściwym miejscu przy pomocy funkcji: *pack*, *grid* lub *place*. Postanowiłam użyć funkcji *grid*, aby rozmieścić wszystkie widżety. Następnie przy pomocy funkcji zdarzeń połączyłam niektóre widżety z innymy, aby mój interfejs był bardziej interaktywny i co za tym idzie przyjazny użytkownikowi.

Po zaprojektowaniu graficznego interfejsu użytkownika miałam problem z aktualizacją wyświetlanych na nim danych. Okazało się, że biblioteka, która tworzy interfejs użytkownika ma wbudowaną funkcję *after ()*, która może rozwiązać problem aktualizacji danych w czasie rzeczywstym. Funkcja *after* z biblioteki *tkinter* użyta w funkcji która ma się co jakiś czas odświeżać aktualizuje ją z zadaną częstotliwością. Zaraz po użyciu funkcji *after ()* w odpowiednim miejscu, mój interfejs zaczął aktualizować dane wyświetlane na interfejsie.

To, czego nie udało mi się osiągnąć, to wyświetlanie wykresu na którym użytkownik mógłby śledzić na bieżąco ilość osób oczekujących w kolejce dla danej sprawy. Użytkownik za pomocą nasićnięcia na przycisk „**Pokaż wykres**” może wyświetlić wykres na któtym widać, ile jest osób oczekujących do danej sprawy w momencie kliknięcia na przycisk, z zastrzeżeniem, że wykres ten nie jest dynamiczny i nie aktualizuje się w czasie rzeczywistym.

IMPORTOWANE BIBLIOTEKI

Aby uruchomić program, powinny być zainstalowane biblioteki takie jak *requests*, *tkinter*, *datetime* i *matplotlib.pyplot*.