Raport

Aplikacja do śledzenia wydatków i przychodów

Przedmiot: Języki Skryptowe

Imię i nazwisko autora: Adam Krzemiński

Nr indeksu: 246750

Semestr studiów: 5

Data ukończenia pracy: 19 Stycznia 2021 r.

Prowadzący laboratorium: mgr inż. Natalia Piórkowska

Spis treści

[Raport 1](#_Toc61296361)

[1. Cel projektu: 3](#_Toc61296362)

[2. Wymagania projektowe 3](#_Toc61296363)

[3. Opis architektury systemu 4](#_Toc61296364)

[4. Opis zastosowanych rozwiązań 4](#_Toc61296365)

[5. Prezentacja interfejsu 4](#_Toc61296366)

[6. Uruchomienie aplikacji 6](#_Toc61296367)

[7. Podsumowanie 6](#_Toc61296368)

[8. Literatura 7](#_Toc61296369)

[9. Aneks 7](#_Toc61296370)

## Cel projektu:

Celem przedsięwzięcia jest stworzenie projektu, który umożliwi użytkownikowi zarządzanie swoimi finansami poprzez przejrzyste śledzenie wydatków oraz przychodów.

## Wymagania projektowe

Aplikacja spełnia następujące wymagania projektowe:

**Funkcjonalne:**

1. Aplikacja pozwala użytkownikowi na łączenie się z serwerem aplikacji poprzez przeglądarkę internetową.
2. Aplikacja pozwala użytkownikowi na założenie konta poprzez rejestrację w systemie oraz późniejsze logowanie, które jest wymagane by móc korzystać z funkcjonalności aplikacji.
3. Aplikacja pozwala na modyfikację danych użytkownika – nazwy użytkownika, adresu e-mail oraz hasła, a także pozwala na zmianę wyświetlanej waluty.
4. Aplikacja umożliwia przeglądanie statystyk dla konta użytkownika – informacji odnośnie czasu posiadania konta, liczby dodanych wydatków, przychodów, budżetów itp.
5. Aplikacja umożliwia tworzenie, edytowanie oraz usuwanie własnych kategorii wydatków oraz kategorii przychodów.
6. Aplikacja pozwala użytkownikowi na tworzenie, edytowanie oraz usuwanie budżetów, do których później można dodawać wydatki oraz przychody.
7. Użytkownik, dla wybranego budżetu może tworzyć wydatki oraz przychody specyfikując ich kategorię, wartość, datę oraz opis.
8. Aplikacja na bieżąco tworzy statystyki, wykresy oraz raporty na podstawie dodanych przez użytkownika danych i wyświetla je na stronie głównej aplikacji.
9. Użytkownik ma możliwość wyświetlania oraz eksportu raportów z utworzonych przez niego danych (m.in. zestawienie tygodniowych/miesięcznych wydatków, zestawienie dziennych bilansów budżetów itp.).

**Niefunkcjonalne:**

1. Aplikacja jest aplikacją webową.
2. Aplikacja posiada interfejs graficzny, użytkownik komunikuje się z systemem poprzez przeglądarkę internetową.
3. Wprowadzane przez użytkownika dane zapisywane są do bazy danych SQLite.
4. Eksportowane pliki są zapisywane w formacie csv lub xlsx.

## Opis architektury systemu

Aplikacja została stworzona według wzorca projektowego MVC. Aplikacja składa się kontrolera umieszczonego w pliku routes.py, modeli umieszczonych w pliku models.py oraz widoków, które znajdują się w folderze templates. Kontroler pobiera dane z bazy danych, przetwarza je oraz przekazuje je do widoku, a także pobiera dane z widoku i zapisuje w bazie danych. Modele są odzwierciedleniem tabeli w bazie danych SQLite, która znajduje się również na serwerze aplikacji, w pliku sites.db. Widoki to pliki html, odpowiadające za interfejs graficzny aplikacji, które korzystają również z biblioteki Bootstrap oraz skryptów JavaScript. Arkusze stylów css oraz pliki skryptów JavaSript umieszczone są w podkatalogach folderu static. Głównym plikiem widoku jest plik layout.html, który odpowiada za części wspólne wszystkich widoków aplikacji, które po nim dziedziczą. Oprócz opisanych powyżej plików, aplikacja zawiera również plik \_\_init\_\_.py, który odpowiada za początkową konfigurację aplikacji podczas uruchamiania. Plik forms.py zawiera formularze z biblioteki WTForms, przekazywane do widoków i używane do interakcji z użytkownikiem. W folderze głównym aplikacji znajduje się również plik run.py, który uruchamia aplikację.

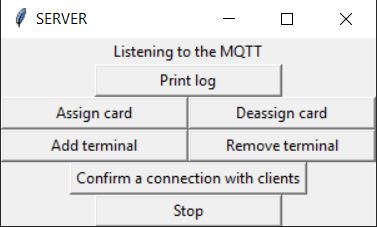
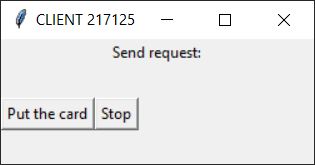
## Opis zastosowanych rozwiązań

W aplikacji zostały zastosowane następujące rozwiązania:

1. Całość aplikacji została napisana w języku python, przy użyciu biblioteki Flask.
2. Sesja użytkownika oraz funkcjonalność logowania została zaimplementowana przy użyciu biblioteki flask\_login.
3. Formularze przekazywane do widoków, z których pochodzą dane od użytkownika zostały utworzone z użyciem biblioteki WTForms.
4. Mapowanie obiektów klas Python do rekordów tabeli bazy danych SQLite w piku sites.db jest realizowane za pomocą biblioteki SQLAlchemy.

## Prezentacja interfejsu

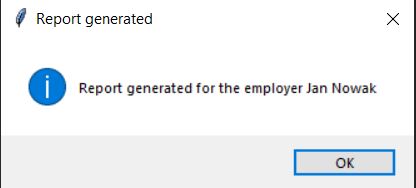
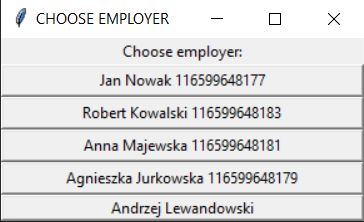
* Po uruchomieniu, aplikacje serwerowa oraz kliencka wyglądają w następujący sposób:

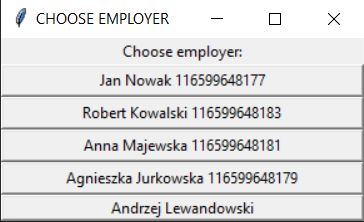
* Przykładowa symulacja przyłożenia karty wygląda w następujący sposób. Po wciśnięciu przycisku „Put the card” w aplikacji klienckiej, w konsoli aplikacji serwerowej pojawia się informacja o użyciu karty:

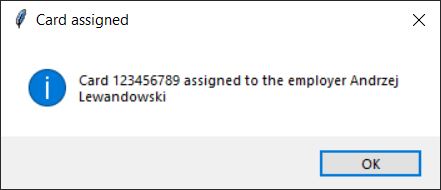


* Przykładowe wygenerowanie raportu dla pracownika, po wciśnięciu przycisku „Print log” w aplikacji serwerowej wygląda w następujący sposób:

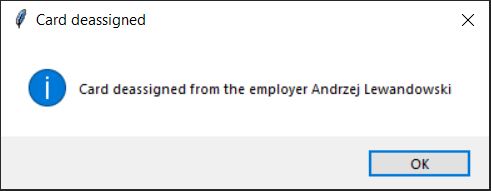
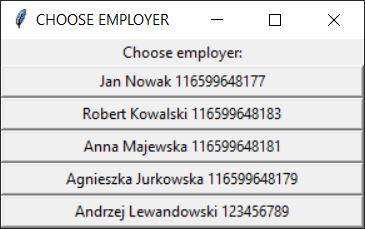


* Przykładowe przypisanie numeru karty pracownikowi (Andrzejowi Lewandowskiemu), po kliknięciu przycisku „Assign card” w głównym menu:

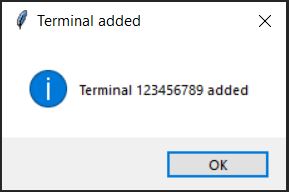
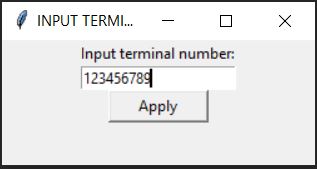




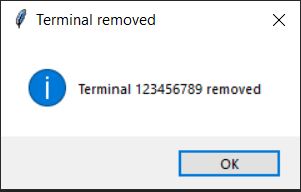
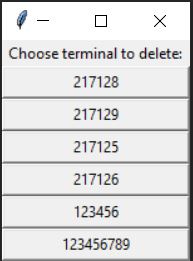
* Przykładowe usunięcie przypisania numeru karty pracownikowi (Andrzejowi Lewandowskiemu), po kliknięciu przycisku „Deassign card” w głównym menu:



* Przykładowe dodanie terminala do systemu, po kliknięciu przycisku „Add terminal” w głównym menu:



* Przykładowe usunięcie terminala z systemu, po kliknięciu przycisku „Remove terminal” w głównym menu:



## Uruchomienie aplikacji

W celu uruchomienia aplikacji w przeglądarce internetowej, można połączyć się z serwerem aplikacji, używając adresu: <expenseandincome-tracker.herokuapp.com>

W celu zainstalowania oraz uruchomienia aplikacji z kodu źródłowego na własnym komputerze, należy:

* Pobrać folder aplikacji wraz z całą zawartością.
* Zainstalować na komputerze język Python.
* Za pośrednictwem instalatora pip zainstalować wszystkie pakiety, których nazwy zamieszczone są w pliku requirements.txt.
* Uruchomić plik run.py z głównego folderu aplikacji. Aplikacja działa na adresie localhost:5000/

## Podsumowanie

Wykonany projekt zgodny jest z przedstawionymi wymaganiami funkcjonalnymi oraz niefunkcjonalnymi. Podczas implementacji projektu nie wystąpiły znaczne trudności. W przyszłości projekt powinien zostać rozbudowany o szerszą możliwość zarządzania danymi przez użytkownika, m. in. Możliwość usunięcia swojego konta. Kolejną propozycją rozszerzenia projektu jest przystosowanie interfejsu graficznego do korzystania z aplikacji na przeglądarkach internetowych urządzeń mobilnych. Istnieje również możliwość utworzenia widoków do seryjnego wprowadzania danych, co poprawi intuicyjność korzystania z aplikacji. Można również uzupełnić stronę główną użytkownika o dodatkowe statystyki i podsumowania. Inną możliwą funkcją do rozbudowy aplikacji jest dodanie obsługi wielu walut oraz języków interfejsu aplikacji.

## Literatura

1. Dokumentacja języka Python.
2. Dokumentacja biblioteki Flask.
3. Informacje na temat budowania aplikacji webowej: <https://youtu.be/MwZwr5Tvyxo>.

## Aneks

Do niniejszego raportu załączony został kod projektu w formie elektronicznej, który dostępny jest również na platformie github, w repozytorium: <https://github.com/admkrz/ExpenseTracker>.