Analiza obiektowa projektu: *Planowanie budżetu*

Marta Kałużna

20 czerwca 2019

1. Spis klas

a) Categories

b) Tables

c) Budget

d) Generator

e) Transaction

f) User

g) Users

2. Opis architektury systemu

Program ma na celu ułatwienie planowania budżetu. Budżet dzielimy na rzeczywisty i planowany, a transakcje na wydatki i przychody. Dzięki odpowiednim powiązaniom między rzeczywistym i planowanym budżetem, użytkownik może na bieżąco oceniać sytuację swojego portfela. Dzięki tabelom utworzonym przy pomocy biblioteki *pandas* może szybko zorientować się, czy aby na pewno wszystko jest pod kontrolą i ile pieniędzy może jeszcze wydać. Dodatkowo, interakcja stworzona w programie, pozwala utworzyć więcej niż jeden profil użytkownika. Informacje dotyczące poszczególnych użytkowników i ich budżetów, przechowywane są w osobnych plikach. Zapisują się również tabele i wykresy. Przy każdym odtwarzaniu programu, pliki wczytują się, dzięki czemu, po ponownym zalogowaniu, użytkownik ma możliwość dalszego edytowania swojego budżetu.

3. Opis klas

a) Klasa **Categories** odpowiada za kategorie wydatków i przychodów. Jej dwoma atrybutami są listy: exp_categories i inc_categories, zawierające odpowiednio kategorie wydatków i przychodów. Na starcie w jednej z list znajdują się wydatki typu: 'Mieszkanie', 'Jedzenie', 'Transport', 'Długi', 'Opieka zdrowotna', 'Dzieci', 'Rozrywka', 'Oszczędności', 'Inne', a w drugiej przychody: 'Wynagrodzenie', 'Premia', 'Inne'. Metody, które zawiera klasa Categories, to:

- *add_exp_category* dodaje nową kategorię wydatków (wyrzuca błąd, gdy podana kategoria już istnieje),
- *add_inc_category* dodaje nową kategorię przychodów (podobnie jak wyżej: wyrzuca błąd, gdy podana kategoria już istnieje),
- *show_exp_categories* wyświetla listę wszystkich kategorii wydatków,
- *show_inc_categories* wyświetla listę wszystkich kategorii przychodów,
- str - wyświetla 'opisową' wersję aktualnych kategorii.
  Wszystkie atrybuty i operacje tej klasy są publiczne.

b) Klasa **Transaction** odpowiada za pojedynczą transakcję, czyli za wydatek lub przychód. Definicja klasy Transaction "używa" klasy Categories. Argumentami klasy Transaction są: category (kategoria transakcji), t (typ budżetu: planowany/rzeczywisty), amount (kwota), day (dzień transakcji). Metody tej klasy, to:

- *check_amount* wyświetla kwotę transakcji,
- *check_day* wyświetla dzień transakcji,
- *check_type* wyświetla typ budżetu transakcji,
- *check_category* wyświetla kategorię transakcji.
- str wyświetla opis transakcji

c) Klasa **Tables** odpowiada za tabele danego budżetu: przypływów, wydatków i wydatków dzień po dniu (to właśnie są atrybuty tej klasy, dodatkowym atrybutem jest obiekt klasy Categories, dzięki któremu zapamiętujemy i dodajemy kategorie do budżetu). Ramki danych zostały wygenerowane przy użyciu biblioteki pandas. Metody tej klasy to:

- *add_inc_cat* dodaje nową kategorię przychodów do tabeli przychodów, wykorzystuje przy tym metodę add_inc_category klasy Categories,
- *add_exp_cat* dodaje nową kategorię wydatków do tabeli wydatków i tabeli dzień po dniu, wykorzystuje przy tym metodę add_exp_category klasy Categories,
- *show_df_exp* wywołuje ramkę danych wydatków,
- *show_df_inc* wywołuje ramkę danych przychodów,
- *show_df_details* wywołuje ramkę danych wydatków dzień po dniu,
- *show_pieplot* tworzy wykres kołowy wydatków,
- *show_barplot* tworzy wykres słupkowy budżetu,
- *show_selected_days* wywołuje ramkę danych wydatków od dnia a do dnia b.

d) Klasa **Budget** jest główną klasą budżetu. Dziedziczy atrybuty i metody klasy Tables. Dodatkowymi atrybutami są 4 listy: planowane wydatki, planowane przychody, rzeczywiste wydatki, rzeczywiste przychody (elementy tych list są obiektami klasy Transaction). Metody tej klasy:

- *add_income* dodaje przychód do budżetu, czyli obiekt klasy Transaction. W zależności od tego czy wybrano typ budżetu 'p' czy 'r', trafia on odpowiednio do listy planowanych lub rzeczywistych przychodów. Rzuca wyjątek, jeśli category nie znajduje się na liście kategorii przychodów danego budżetu. Aktualizuje również wygląd tabeli, uwzględniając nowe dane.
- *add_expense* analogicznie jak wyżej, dodaje wydatek do budżetu, czyli obiekt klasy Transaction, itd.
- *show_transactions* wyświetla opis wszystkich transakcji z podziałem na rzeczywiste/planowane i przychody/wydatki
- *sum_of_plan_incomes* zwraca sumę planowanych przychodów
- *sum_of_real_incomes_* zwraca sumę rzeczywistych przychodów
- *sum_of_plan_expenses* zwraca sumę planowanych wydatków
- *sum_of_real_expenses* zwraca sumę rzeczywistych wydatków
- *plan_budget* zwraca opis planowanego budżetu
- *real_budget* zwraca opis rzeczywistego budżetu
- *warnings* zwraca ostrzeżenia na jakie kategorie użytkownik powinien zwrócić uwagę (gdzie wydatki rzeczywiste przekraczają planowane)

e) Klasa **Generator** służy do symulacji przykładowego budżetu. Jej atrybutami są: n, czyli liczba wszystkich transakcji i budzet, czyli obiekt klasy Budget. Metoda tej klasy to:

- *simulation* wykonuje symulację budżetu

f) Klasa **User** to klasa, która tworzy użytkownika. Użytkownik ma przypisany login i hasło, a także swój własny budżet (obiekt klasy Budget). Metody tej klasy to:

- *change_password* zmienia hasło użytkownika, jeśli nowe hasło jest takie same jak dotychczasowe - zwraca błąd
- *show_budget* zwraca informacje na temat budżetu użytkownika
- *clear_budget* czyści budżet użytkownika (wszystkie dane)

Metody zaczerpnięte z klasy Budget (opisane wyżej), które powstały w celu ułatwienia późniejszej obsługi:

- *show_trans*
- *add_inc*

- *add_exp*
- *show_warnings*

g) Klasa **Users** odpowiada za "bazę" użytkowników - gromadzi informacje na ich temat: login, hasło, budżet. Jej atrybutami są: users (lista obiektów klasy User), krotki (lista krotek postaci: (login, hasło)), logins (lista loginów, dodatkowo wyodrębniona w celu uproszczenia działań w metodach). Metody tej klasy to:

- *add_user* dodaje nowego użytkownika
- *delete_user* usuwa wybranego użytkownika
- *show_users* zwraca listę loginów użytkowników
- *select_user* zwraca wybranego użytkownika (obiekt klasy User)
- *change_pass* zmienia hasło wybranego użytkownika

Dodatkową funkcją jest **interaction**, która po połączeniu powyższych klas, umożliwia interakcję z użytkownikiem, a także zapisuje pliki.