ZADANIE REKRUTACYJNE

HAWATEL

Łukasz Ruzicki

Aplikacja aktualizująca ceny produktów sklepu e-commerce używająca REST API do Narodowego Banku Polskiego.

Gdynia 2022

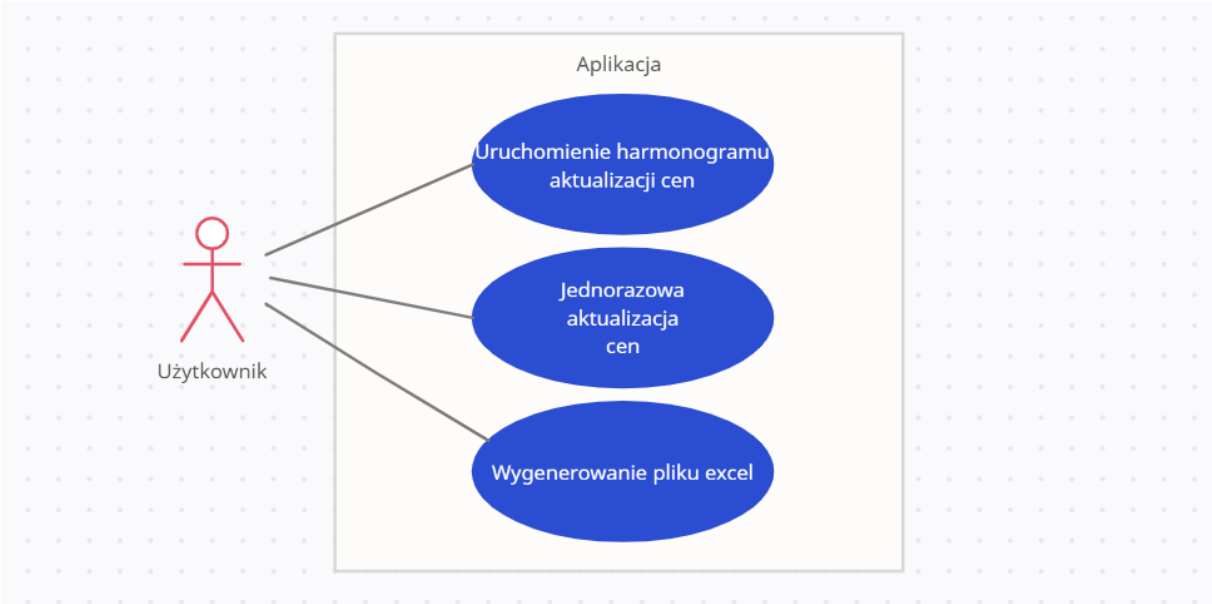
Contents

1. Opis potrzeby użytkownika końcowego

Sprzedawca posiada sklep e-commerce z różnymi produktami. Do tej pory handel był realizowany na terenie Polski. Sprzedawca chciałby wysyłać towary do krajów Unii Europejskiej oraz do Stanów Zjednoczonych. Pojawiała się zatem u niego potrzeba akceptacji płatności w dolarach amerykańskich (USD) oraz w Euro. Kupujący musi wiedzieć ile towar kosztuje w danej walucie. Sprzedawca potrzebuje rozwiązania, które cyklicznie raz dziennie lub na żądanie pobierze aktualny kurs walut z Narodowego Banku Polskiego i dokona aktualizacji cen dla produktów w bazie danych.

2. Projekt

2.1 Aktorzy i przypadki użycia



Jedynym aktorem programu jest użytkownik, który będzie miał możliwość uruchomienia codziennej aktualizacji cen, aktualizacji programu jednorazowo lub generowania pliku excel z produktami sklepu. W przypadku uruchomienia aktualizacji codziennej zaleca się uruchomienie to na serwerze.

2.2 Model danych

Dane sklepu e-commerce przechowywane są w bazie MySQL. Można edytować je bezpośrednio przez MySQL Workbench przy użciu zapytań SQL lub dokonywać zmian poprzez skrypt Pythona. W przypadku tej aplikacji wszystkie zmiany dokonywane są przez skrypt Pythona.

2.3 Projekt interfejsu

W myśl Keep It Simple do uruchomienia programu użyty został terminal. Wybór dokonywany jest poprzez wprowadzenie liczby odpowiadającej interesującej opcji użytkownika. Zaoszczedza to czas przy tworzeniu GUI i w bardzo prosty sposób umozliwia dalszy rozwój interfejsu i dodanie kolejnych opcji.

3. Implementacja

3.1 Baza danych

Bza danych została zaimportowany schemat sklepu e-commerce, który jest dostępny pod adresem https://raw.githubusercontent.com/abdelatifsd/E-commerce-Database-

Project/master/3%20-%20Structure.sql przy użyciu MySQL Workbench. Następnie zostały zaimportowane testowe dane, które są dostępne pod adresem https://raw.githubusercontent.com/abdelatifsd/E-commerce-Database-

Project/master/4%20-%20Population.sql

Wszystkie zapytania wykonane zostały jedno po drugim, aby uniknąć ewentualnych problemów z wykonywania ich jednocześnie.

3.2 Najważniejsze paczki I moduły - Python

3.2.1 mysql.connector,

Umożliwia połaczenie skryptu z bazą danych MySQL. Dzięki niemu możliwe jest wykonywanie zapytań przy użyciu skryptu Python.

3.2.2 pandas, openpyxl

Dane zostają zaimportowane do Data Frame, czyli struktury danych. Następnie generowany jest plik Excel na podstawie struktury.

3.2.3 reguest

Biblioteka umożliwająca tworzenie zapytań HTTP przez Pythona.

3.2.4 logging

Tworzenie pliku z logami informującymi o działaniach programu i jego efektach.

3.2.5 schedule

Biblioteka umożliwiająca uruchamianie skryptu w określonym czasie, w tym przypadku codziennie.

3.3 Wymagania funkcjonalne

3.3.1 Dodanie kolumn UnitPriceUSD i UnitPriceEURO

```
4.  def addColumndsTomydb(self):
5.          # add coulumns to database
6.          try:
7.              self.mycursor.execute("ALTER TABLE `mydb`.`Product` ADD `UnitPriceUSD`
    DECIMAL")
8.          except:
9.              print("UnitPriceUSD already exists")
10.         try:
11.             self.mycursor.execute("ALTER TABLE `mydb`.`Product` ADD `UnitPriceEURO`
    DECIMAL")
12.         except:
13.             print("UnitPriceEURO already exists")
```

Dodane zostają kolumny do pliku przed próbą aktualizacji cen. Jeżeli istenią zostaną wyświetlone komunikaty, nie przerwie to działania programu.

3.3.1 Połączenie po REST API do NBP

```
14.     def getDataFromNBPAPI(self):
15.         response = requests.get("http://api.nbp.pl/api/exchangerates/tables/a/")
16.         if response.status_code != 200:
17.             self.logger.error("NBP API dosen't respond! Prices not updated!")
18.
19.         # take data from json file
20.         exchange_info = response.json()
21.         for i in exchange_info[0]['rates']:
22.             if i['code'] == 'EUR':
23.                 current_euro_exchange = i['mid']
24.             if i['code'] == 'USD':
25.                 current_usd_exchange = i['mid']
26.
27.         self.updatePrices(current_euro_exchange, current_usd_exchange)
```

Wysyłane jest zapytanie przez REST API do NBP. W przypadku uzyskania odpowiedzi 200, oznaczającej sukces zapytania, zostaje ono zwrócone w frmie pliku json skąd wyciągane są informacje o aktualnych kursach. Dla kluczowych danych wybierane są porządane kwoty.

Zwrócenie innego kodu zapytania niż 200, spowoduje zapis do logu.

3.3.2 Aktualizacja cen

```
28. def updatePrices(self, current_usd_exchange, current_euro_exchange):
29.         # add foreign currency prices to mydb
30.         all_products = self.mycursor.fetchall()
```

```
31.         for iterate_product in self.result:
32.             foreign_currency_prices = (float(iterate_product[6]) / current_euro_exchange,
    float(iterate_product[6]) / current_usd_exchange, iterate_product[0])
33.             self.mycursor.execute("UPDATE `mydb`.`Product` SET `UnitPriceEURO` = %s,
    `UnitPriceUSD` = %s WHERE `ProductID` = %s", foreign_currency_prices)
34.
35.         self.mydb.commit()
36.         self.logger.error("UnitPriceEURO and UnitPriceUSD updated")
```

Ceny przeliczane są zgodnie z kursami. Dane są zamieniane na odpowiednie typy, aby można było wykonać operacje. Typ wyniku pasuje do UnitPrice z bazy danych. Aktualizowane są następnie dane poprzez wykonanie zapytania „UPDATE” dla odpowiednich produktów.

3.3.1 Generowanie Excela

```
4   def importToExcel(self, filename = 'products.xlsx'):
5           self.startScheduler()
6           # prepare data to import mydb to dataframe and excel file
7           self.mycursor.execute("SHOW COLUMNS FROM `mydb`.`Product`;")
8           result = self.mycursor.fetchall()
9           column_names = []
10          column_index = []
11          # set column order to match the client require order
12          for i in range(11):
13              column_index.append(i)
14              if i == 6:
15                  column_index += ([12, 13])
16          for i in column_index:
17              column_names.append(result[i][0])
18
19          # make empty dataframe with coulmn names from mydb
20          df = pd.DataFrame(columns = column_names)
21
22          product_data = []
23          self.mycursor.execute("select * FROM `mydb`.`Product`")
24          result = self.mycursor.fetchall()
25
26          # make dict with pairs column name: value in client order
27          for product in result:
28              dict = {column_names[i]: product[column_index[i]] for i in
    range(len(column_index))}
29              df = df.append(dict, ignore_index = True)
30
31          # import dataframe to excel file
32          df.to_excel(filename)
```

Excel generowany jest na podstawie Data Frame z nazwami kolumn w kolejności ustalonej w wymaganiach przez klienta. Wygenerowany plik może mieć zmienioną nazwę poprzez wprowadzenie argmunetów przy wywołaniu.

3.2.3 Wymagania niefunkcjonalne

3.2.4 OOP

Wszystko zostało napisane obiektowo.

3.2.5 Dokumentacja

Czytasz ją 😊

3.2.6 Rozwiązanie dostępne na Github

https://github.com/lukruz/Price-updater

3.2.7 Logi

Logi zapisywane są do pliku „logfile.log”.

3.2.8 Export schematu SQL

Dostępny w folderze na GitHub

3.2.9 Obsługa wyjątków

Wszystkie wymagane wyjątki są obsługiwany przy pomocy „try i except”. Dodatkowo zostało to użyte przy menu i w kilku niezbędnych miejsach.

4 Testy

4.1 Utworzenie istniejącej kolumny

Uruchomienei skryptu dwukrotnie spowoduje dodanie kolumny raz. Nie zostanie ona dodana dwukrotnie, ani nie zostanie pokazany błąd zapytania SQL.

4.2 Generowanie pliku Excel w pierwszej kolejności

Wybranie opcji samego generowania pliku Excel w menu poprzez wybranie opcji „3”.

4.3 Niepołączenie do NBP

Wpisana zostanie do logów informacja.

Mógłbym opisywać testy manualne, opisać rezultaty, jak to dokładnie testować i jak wynik byłby spodziewany, ale nie ma co się tak rozpisywać w zadaniu rekrutacyjnym.

5. Uruchomienie programu

Otworzyć plik dist -> main -> main.exe

Uruchomi się terminal, wyświetli się menu z 3 opcjami. Wprowadzenie innej opcji niż 1, 2, 3 spowoduje powtórzenie zapytania do skutku. Po wykonaniu programu, terminal nie czeka na żaden sygnał i zakańcza program. W zależności od opcji zostanie uruchominy skrypt harmonogramu, jednorazowa aktualizacja lub generowanie pliku excel.