Politechnika Poznańska, WIiT
Hurtownie danych i przetwarzanie analityczne

# Wyszukiwarka taryfikatorów dla banku PKO BP
# Dokumentacja

Tomasz Jankowiak 141235
Zuzanna Juszczak 141238
Stanisław Kaczmarek 141240
Eryk Kosmala 141249

8 czerwca 2022

# Spis treści

# 1 Opis problemu, założenia projektu oraz koncepcja rozwiązania

## 1.1 Opis problemu

Taryfy Opłat i Prowizji w banku PKO BP[1] to obszerne dokumenty zawierające wszystkie potrzebne dane o obowiązujących obecnie cenach. Wyszukanie w nich interesującej dla klienta informacji jest trudne, ponieważ taryfy zawierają często pozycje, które z jego perspektywy są zbędne i przez to wydłużają czas wyszukiwania.

## 1.2 Założenia

Celem projektu jest stworzenie wyszukiwarki dla taryf, dzięki której klient mógłby wyszukać konkretną opłatę/prowizję bez konieczności szukania jej w plikach PDF. W wyszukiwarce klient wpisuje frazę, która go interesuje i jako wynik dostaje wszystkie pozycje z taryfy, zawierające daną frazę.

Na wykonanie projektu składa się więc kilka etapów, są to:

1. stworzenie parsera dokumentów PDF zawierających taryfy, którego celem jest wyekstrahowanie wszystkich istotnych danych o opłatach i prowizjach,
2. stworzenie bazy danych, przechowującej wynik parsera,
3. zaimplementowanie logiki wyszukiwania w bazie żądanej frazy, przygotowanie funkcjonalności wyszukiwania danych w zadanym okresie czasowym, aby możliwe było wyświetlanie również dokumentów historycznych oraz zaimplementowanie słownika synonimów, dzięki któremu można wyszukać daną opłatę/prowizję używając słów bliskoznacznych,
4. przygotowanie prototypu wyszukiwarki, na który składa się aplikacja backendowa i frontendowa, co pozwala na wizualne przedstawienie funkcjonalności stworzonego projektu.

## 1.3 Rozwiązanie problemu

W celu wykonania projektu najpierw należy napisać parser pozwalający na przetwarzanie dokumentów pdf w formacie dostarczonym przez bank. Powinien pozwalać on na przetworzenie tabel w respektowanym przez klienta układzie. W celu jego implementacji wykorzystany zostanie język programowania python oraz zewnętrzna biblioteka stosowana do parsowania dokumentów pdf. Mając gotowy parser konieczne będzie zaprojektowanie bazy danych przechowujące wyniki parsowania dokumentów. W celu poprawnego składowania danych należy zaprojektować bazę danych, która pozwoli zachować hierarchię poszczególnych opłat z parsowanych dokumentów. W tym celu w bazie danych będą tabele zaprojektowane tak, by przechowywały dane o swoich elementach nadrzędnych.

Mając wyniki parsera przechowywane w odpowiedni sposób w zaprojektowanej bazie danych, możliwa będzie implementacja algorytmu wyszukującego dane z bazy danych na podstawie słów kluczowych użytkownika.

Ostatecznie wynik zostanie zaprezentowany w postaci prototypu wyszukiwarki. Będzie zawierała ona wszystkie podstawowe elementy niezbędne do przeprowadzenia procesu wyszukiwania.

[1] https://www.pkobp.pl/oplaty-i-prowizje

# 2 Parser dokumentów PDF

## 2.1 Biblioteka `pdfplumber`

Pierwszym celem projektu było wydobycie danych o taryfach z dokumentów w formacie PDF. Do implementacji parsera wykorzystaliśmy bibliotekę `pdfplumber`[2] dla języka Python.

Główne dane w parsowanych przez nas dokumentach znajdują się w tabelach. Algorytm ekstrahowania tabel użyty w bibliotece oparty jest na algorytmie Nurminena opisanym w pracy naukowej „Algorithmic extraction of data in tables in PDF documents"[3]. Jego działanie można przedstawić w następujących krokach:

1. znajdź wszystkie linie,
2. scal linie nakładające się lub prawie nakładające się,
3. znajdź przecięcia tych linii,
4. znajdź komórki, których wierzchołki to te przecięcia,
5. pogrupuj znalezione komórki w tabele.

## 2.2 Wynik parsera

Wejściowym parametrem parsera może być albo plik w formacie PDF, albo jedynie ścieżka do niego w lokalnym systemie plików. Wynik parsera, który następnie jest przekazywany do bazy danych i tam utrwalany, wygląda następująco:

```
class Result:
    part: str
    section: str
    validity_date: datetime
    table_results: List[TableResult]
```

Listing 1: Wynik parsera - cały dokument.

Klasa `TableResult`, której listę obiektów zawiera w sobie wynik parsera, składa się z pól pokazanych na Listingu 2. Klasa `Tariff` zawiera dane m.in. takie jak: `content`, `references` oraz `prices`.

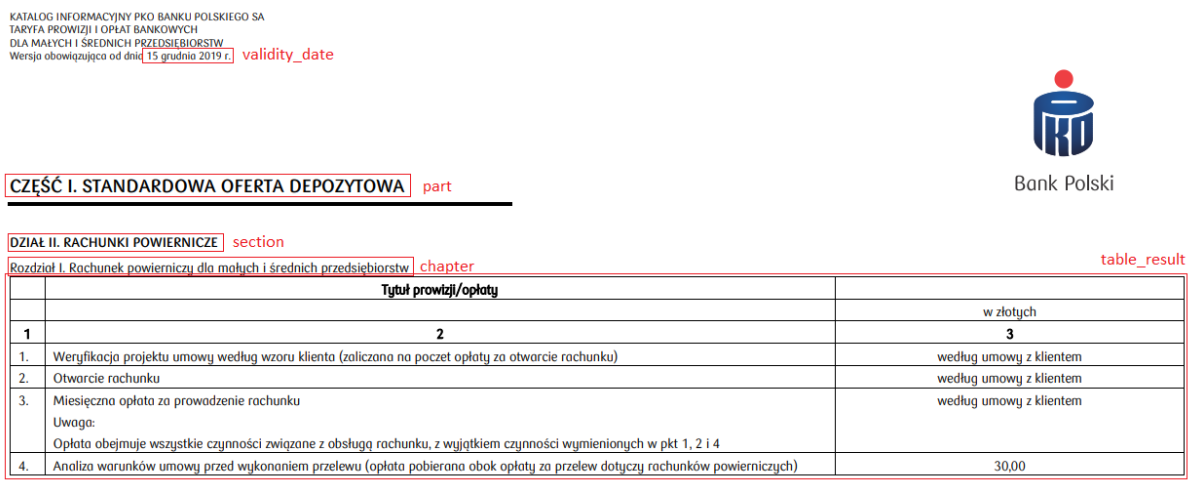
```
class TableResult:
    chapter: Optional[str]
    account_types: List[Optional[str]]
    fee_currencies: List[str]
    tariff: Tariff[TariffText]
```

Listing 2: Wynik parsera - jedna tabela.

[2] https://github.com/jsvine/pdfplumber

[3] https://www.semanticscholar.org/paper/ANSSI-NURMINEN-ALGORITHMIC-EXTRACTION-OF-DATA-IN-IN-Elomaa/a9b167a86fb189bfcd366c3839f33f0404db9c10

Na rysunkach zaznaczone są wydobywane elementy dokumentu, które zostały opisane na powyższych listingach.



KATALOG INFORMACYJNY PKO BANKU POLSKIEGO SA
TARYFA PROWIZJI I OPŁAT BANKOWYCH
DLA MAŁYCH I ŚREDNICH PRZEDSIĘBIORSTW
Wersja obowiązująca od dnia 15 grudnia 2019 r. validity_date

Bank Polski

CZĘŚĆ I. STANDARDOWA OFERTA DEPOZYTOWA part

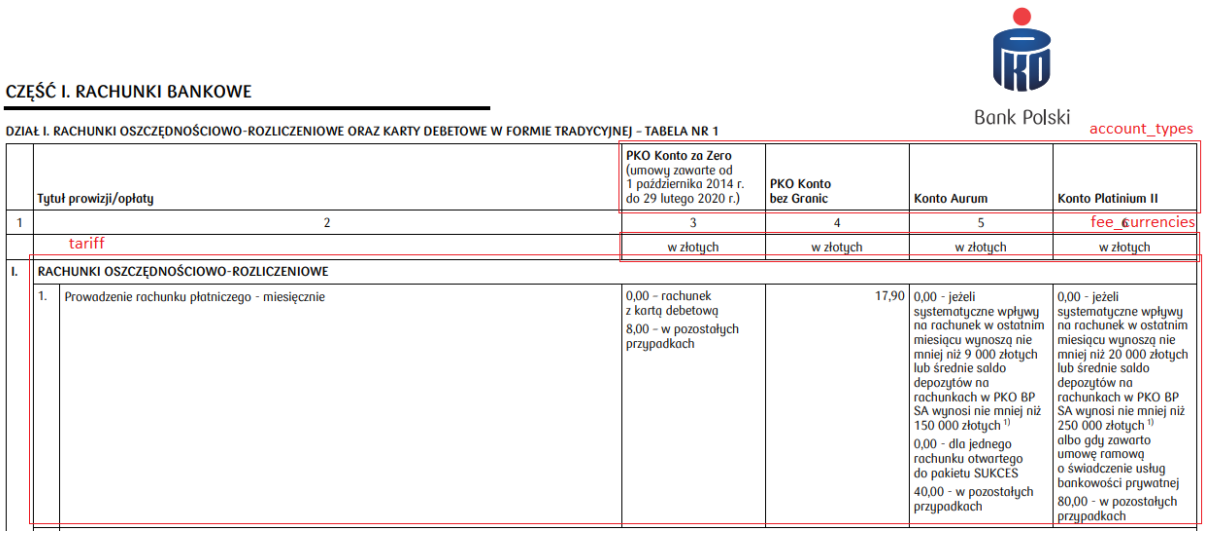
DZIAŁ II. RACHUNKI POWIERNICZE section

Rozdział I. Rachunek powierniczy dla małych i średnich przedsiębiorstw chapter

table_result

| | Tytuł prowizji/opłaty | |
|---|---|---|
| | | w złotych |
| 1 | 2 | 3 |
| 1. | Weryfikacja projektu umowy według wzoru klienta (zaliczana na poczet opłaty za otwarcie rachunku) | według umowy z klientem |
| 2. | Otwarcie rachunku | według umowy z klientem |
| 3. | Miesięczna opłata za prowadzenie rachunku<br>Uwaga:<br>Opłata obejmuje wszystkie czynności związane z obsługą rachunku, z wyjątkiem czynności wymienionych w pkt 1, 2 i 4 | według umowy z klientem |
| 4. | Analiza warunków umowy przed wykonaniem przelewu (opłata pobierana obok opłaty za przelew dotyczy rachunków powierniczych) | 30,00 |

Rysunek 1: Elementy dokumentu wydobywane przez parser.



Bank Polski

CZĘŚĆ I. RACHUNKI BANKOWE

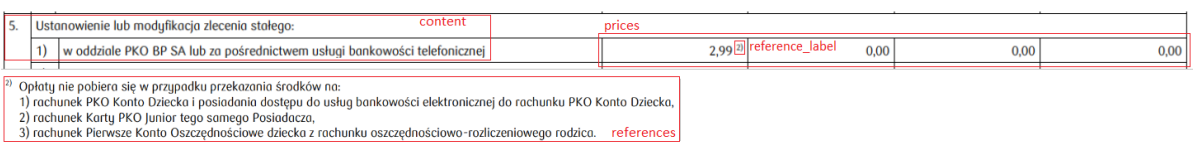
DZIAŁ I. RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE ORAZ KARTY DEBETOWE W FORMIE TRADYCYJNEJ – TABELA NR 1

account_types

| | Tytuł prowizji/opłaty | PKO Konto za Zero (umowy zawarte od 1 października 2014 r. do 29 lutego 2020 r.) | PKO Konto bez Granic | Konto Aurum | Konto Platinium II |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| | tariff | w złotych | w złotych | w złotych | w złotych |
| I. | RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE | | | | |
| 1. | Prowadzenie rachunku płatniczego - miesięcznie | 0,00 – rachunek z kartą debetową<br>8,00 – w pozostałych przypadkach | 17,90 | 0,00 - jeżeli systematyczne wpływy na rachunek w ostatnim miesiącu wynoszą nie mniej niż 9 000 złotych lub średnie saldo depozytów na rachunkach w PKO BP SA wynosi nie mniej niż 150 000 złotych [1]<br>0,00 - dla jednego rachunku otwartego do pakietu SUKCES<br>40,00 - w pozostałych przypadkach | 0,00 - jeżeli systematyczne wpływy na rachunek w ostatnim miesiącu wynoszą nie mniej niż 20 000 złotych lub średnie saldo depozytów na rachunkach w PKO BP SA wynosi nie mniej niż 250 000 złotych [1] albo gdy zawarto umowę ramową o świadczenie usług bankowości prywatnej<br>80,00 - w pozostałych przypadkach |

fee_currencies

Rysunek 2: Elementy tabeli wydobywane przez parser.



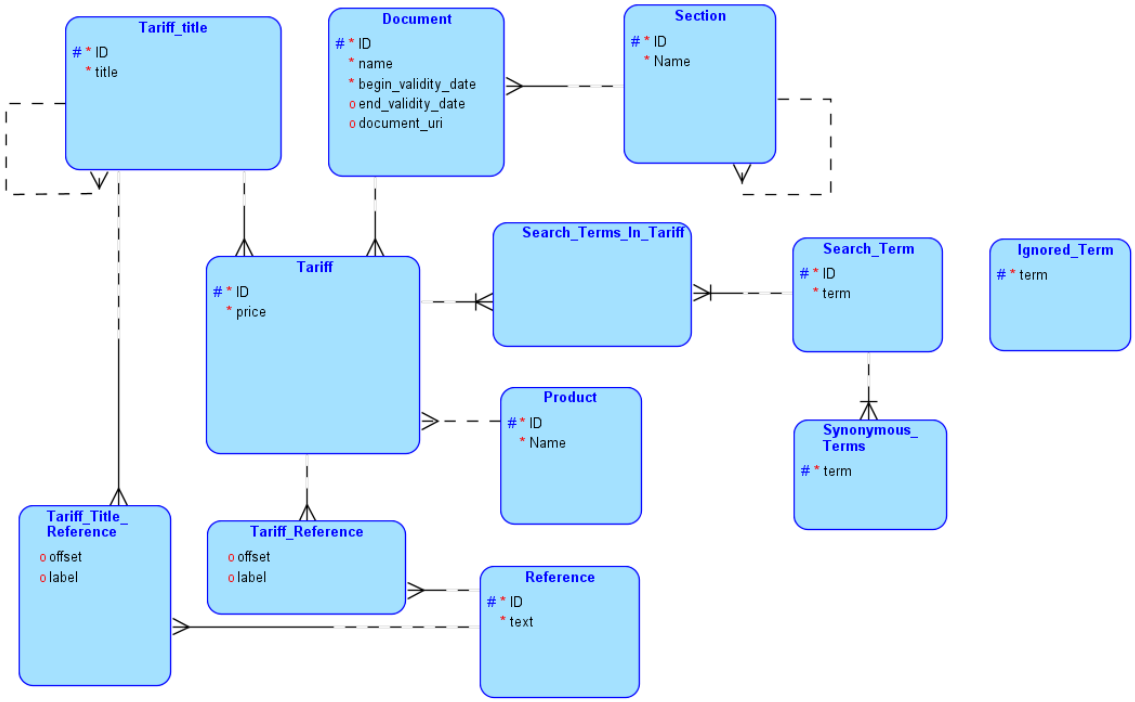
| | | | | | |
|---|---|---|---|---|---|
| 5. | Ustanowienie lub modyfikacja zlecenia stałego: | content | prices | | |
| | 1) w oddziale PKO BP SA lub za pośrednictwem usługi bankowości telefonicznej | 2,99 [2] | reference_label 0,00 | 0,00 | 0,00 |

[2] Opłaty nie pobiera się w przypadku przekazania środków na:
1) rachunek PKO Konto Dziecka i posiadania dostępu do usług bankowości elektronicznej do rachunku PKO Konto Dziecka,
2) rachunek Karty PKO Junior tego samego Posiadacza,
3) rachunek Pierwsze Konto Oszczędnościowe dziecka z rachunku oszczędnościowo-rozliczeniowego rodzica. references

Rysunek 3: Elementy taryfy wydobywane przez parser.

# 3 Baza danych

Model związków-encji bazy danych, którą zaprojektowaliśmy, został przedstawiony poniżej:




Rysunek 4: Model związków-encji bazy danych.

Główna koncepcja, którą postanowiliśmy założyć w ramach tego projektu, to zaprojektowanie takich struktur, które umożliwiałyby nam przechowywanie danych tak, żeby można było zachować hierarchię składowanych danych. Jedną z przyczyn takiego rozwiązania było umożliwienie odtwarzania struktury dokumentu na podstawie danych zawartych w tabelach bazy danych.

Dodatkowo jednym z kluczowych czynników dla optymalizacji wyszukiwania było dodanie dodatkowych (redundantnych) relacji, które nie reprezentują samego dokumentu (`Search_term`, `Synonymous_terms`, `Ignored_term`, `Search_terms_in_tariff`), ale indeksują występujące w nim słowa, dla przyszłego przeszukiwania.

## 3.1 Tabela `Section`

W tabeli `Section` przechowujemy dane o poszczególnych kategoriach i podkategoriach opłat. Przykładowo, znajdują się w niej takie dane, jak:

*„Usługi dla osób fizycznych” → „Rachunki bankowe” → „Rachunki oszczędnościowo-rozliczeniowe...”*

Kluczem głównym jest numer ID (Integer). Dodatkowe kolumny to nazwa kategorii (Text) oraz Parent_ID (Integer), który przechowuje adres ID elementu nadrzędnego, a który pozwala nam na zachowanie hierarchii poszczególnych usług, części i działów. Parent_ID służy nam za klucz obcy, który wskazuje na elementy własnej tabeli. Może mieć wartość `NULL`, gdyż główne usługi nie posiadają elementu nadrzędnego. Dzięki takiej implementacji w łatwy sposób możemy wyszukać główne elementy struktury.

## 3.2 Tabela `Document`

Tabela ta łączy dwie inne tabele – `Section` oraz `Tarrif`. Została ona użyta w celu umożliwienia implementacji obsługi historii poszczególnych opłat. Kluczem głównym jest kolumna ID (Integer). W tabeli znajdują się kolumny Begin_validity_date (Date) oraz End_validity_date (Date, Nullable), które przechowują dane o dacie obowiązywania dokumentu. Kolumna Section_ID (Integer) wskazuje na sekcję, w której znajduje się konkretny dokument. Dodatkowo tabela `Document` zawiera kolumnę Document_uri (Text, Nullable), która przechowuje `lokalne URI` danego dokumentu.

## 3.3 Tabela `Tarrif_title`

W tabeli `Tarrif_title` przechowujemy dane o poszczególnych tytułach prowizji i opłat, które pojawiają się w dokumentach PDF. Ze względu na format tych dokumentów, tabela musiała zostać tak zaprojektowana, żeby umożliwiała przechowywanie nazw taryf wraz z zachowaniem ich miejsca w hierarchii dokumentu.

Aby to zapewnić, umieściliśmy w tabeli kolumnę Parent_tariff_title_ID (Integer, `NULLABLE`), która zawiera ID (Integer) elementów nadrzędnych dla danych opłat. Dla przykładu:

| | | | |
|---|---|---|---|
| 2. | Polecenie przelewu wewnętrznego: | | |
| | 1) | pomiędzy rachunkami tego samego Posiadacza rachunku, prowadzonymi w PKO BP SA oraz przez inne spółki PKO BP SA z siedzibą na terytorium RP | |
| | 2) | na pozostałe rachunki prowadzone w PKO BP SA: | |
| | | a) | w oddziale PKO BP SA lub za pośrednictwem usługi bankowości telefonicznej |
| | | b) | za pośrednictwem usługi bankowości elektronicznej |

Rysunek 5: Fragment dokumentu PDF, który podawany jest parsowaniu.

Element „*1) pomiędzy rachunkami tego samego Posiadacza...*” w kolumnie Parent_ID zawiera ID elementu „*2. Polecenie przelewu wewnętrznego*”.

Pozwolenie na przyjmowanie wartości `NULL` przez tę kolumnę wynika z faktu, że elementy, które są najwyżej w strukturze danego dokumentu, mają przypisaną właśnie tę wartość. Pozwala nam to na proste odróżnienie głównych elementów dokumentu oraz na odtworzenie całej struktury na podstawie wartości Parent_tariff_title_ID, co jest kluczowe podczas filtrowania zapytań użytkowników. Kolumna Document_ID (Integer) zawiera klucz obcy, który odwołuje się do odpowiedniego dokumentu w tabeli `Document`, co pozwala na łatwe przechowywanie danych o dokumencie, z którego dana taryfa pochodzi.

| Tytuł prowizji/opłaty | | **PKO Konto za Zero** (umowy zawarte od 1 października 2014 r. do 29 lutego 2020 r.) |
|---|---|---|
| 2 | | 3 |
| | | w złotych |
| **RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE** | | |
| 1. | Prowadzenie rachunku płatniczego - miesięcznie | 0,00 – rachunek z kartą debetową<br>8,00 – w pozostałych przypadkach |

Rysunek 6: Elementy dokumentu przechowywane w tabeli: `Tarrif_title`

## 3.4 Tabela `Tarrif`

W tabeli `Tarrif` przechowujemy dane o konkretnych cenach dla danych produktów. Oprócz kolumny ID (Integer), która jest identyfikatorem poszczególnej opłaty, w tabeli przechowywane są dane z ceną danej oferty - Price (Text), oraz klucze obce:

- Tarrrif_title_id (Integer) - identyfikuje taryfę, do której odnosi się konkretna cena;
- Product_id (Integer) - określa produkt, którego dotyczy dany wiersz tabeli;
- Document_id (Integer) - przypisuje daną cenę do dokumentu, z którego została ona wyciągnięta.

| Tytuł prowizji/opłaty | | PKO Konto za Zero (umowy zawarte od 1 października 2014 r. do 29 lutego 2020 r.) |
|---|---|---|
| 2 | | 3 |
| | | w złotych |
| RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE | | |
| 1. | Prowadzenie rachunku płatniczego - miesięcznie | 0,00 – rachunek z kartą debetową<br>8,00 – w pozostałych przypadkach |

Rysunek 7: Elementy dokumentu przechowywane w tabeli: `Tarrif`.

### 3.5 Tabela `Product`

W tabeli `Product` znajdują się dwie kolumny: ID (Integer) do identyfikowania produktu oraz name (text), które przechowuje nazwę produktu. Tabela ta jest dodana i wykorzystywana w celu łatwiejszego zarządzania bazą danych - jeżeli będzie konieczne zmienienie nazwy produktu, jego nazwę zmienimy raz w tej tabeli, zamiast dla każdego wiersza opisującego cenę konkretnej taryfy.

| Tytuł prowizji/opłaty | | PKO Konto za Zero (umowy zawarte od 1 października 2014 r. do 29 lutego 2020 r.) |
|---|---|---|
| 2 | | 3 |
| | | w złotych |
| RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE | | |
| 1. | Prowadzenie rachunku płatniczego - miesięcznie | 0,00 – rachunek z kartą debetową<br>8,00 – w pozostałych przypadkach |

Rysunek 8: Elementy dokumentu przechowywane w tabeli: `Product`.

### 3.6 Tabela `Reference`

W tabeli `Reference` przechowujemy dane o przypisach znajdujących się w poszczególnych komórkach tabeli w dokumencie. Tabela posiada dwie kolumny ID (Integer) oraz Text (Text) przechowującą teksty poszczególnych przypisów.

### 3.7 Tabele `TariffTitleReference` oraz `TariffReference`

Obie te tabele służą do stworzenia relacji pomiędzy `Tarrif` oraz `Tariff_Title`, a tabelą `Reference`. Ich zadaniem jest umożliwienie stworzenia relacji wiele-do-wielu pomiędzy powyżej wspomnianymi tabelami, z możliwością przechowywania danych o miejscu wystąpienia przypisu w tekście - Offset (Text) oraz o etykiecie - Label (Text), którą dany przypis ma w konkretnym dokumencie.

### 3.8 Tabela `Search_Term`

W tabeli `Search_Term` znajdują się dwie kolumny: Id (Integer), która identyfikuje konkretne słowo /słowa kluczowe oraz Term (Text), w której znajduje się tekst wspomnianej frazy.

### 3.9 Tabela `Synonymous_Terms`

W tabeli `Synonymous_Terms` przechowujemy dane o synonimach, dla konkretnych wyszukiwanych słów. Dane te znajdują się w dwóch kolumnach: Term (Text) zawiera synonim, a Search_term_id (Integer) identyfikator słowa z tabeli `Search_Term`, dla którego fraza z tej tabeli jest wyrazem bliskoznacznym.

### 3.10 Tabela `Ignored_Term`

Tabela `Ignored_Term` przechowuje dane o słowach i frazach, które chcemy ignorować podczas procesu wyszukiwania. Słowa te znajdują się w pojedynczej kolumnie Term (Text). Utworzenie tej tabeli pozwala na optymalizację procesu wyszukiwania - jeżeli użytkownik poda słowo zbyt ogólne (np. jakiekolwiek spójniki) to pozwoli to nam na zmniejszenie liczby koniecznych przeszukiwań, jednocześnie gwarantując wyszukanie danej oferty na podstawie pozostałych słów kluczowych klienta.

### 3.11 Tabela `Search_Terms_In_Tarrif`

Tabela `Search_Terms_In_Tarrif` zawiera w sobie klucze obce do tabeli `Search_Term` oraz `Tarrif`, co pozwala na zawężenie poszczególnych słów kluczowych dla konkretnych ofert, co także wpływa na szybkość przeszukiwania.

# 4 Wyszukiwanie opłat

## 4.1 Przygotowanie bazy (Indeksowanie słów)

Przed rozpoczęciem dodawania dokumentów, należy na bazie stworzyć i odpowiednio zasilić słownik synonimów. Przy obecnym projekcie bazy jak i algorytmu wyszukiwania, takie ddziałanie pozwala na późniejsze indeksowanie taryf wraz ze słowami, które w nich występują, a które posiadają zdefiniowane synonimy. Jak wspomniane było wcześniej, taki zabieg pozwala na wydalniejsze wyszukiwanie interesujących danych w bazie danych.

Wartym zaznaczenia jest konieczność definicji słownika przed rozpoczęciom procesu parsowania, ze względu na fakt, że słowo występujące w tabeli `Search_term` nie może jednocześnie występować w tabeli `Synonymous_terms`. Jest to celowe ograniczenie, które gwarantuje poprawne wykonanie późniejszego wyszukiwania.

## 4.2 Algorytm wyszukiwania

Użytkownik chcąc zadać konkretne zapytanie do wyszukania ma kilka możliwości do określenia interesujących go ofert i taryf. Klient może zawęzić zakres przeszukiwania poprzez wybranie interesującego go produktu, określenie daty, dla której otrzymać chciałby wynik (przy braku wskazania konkretnej daty, zapytanie wykona się dla daty bieżącej) oraz poprzez wybór sekcji, które podane są w formie kafelków. Główną funkcją wyszukiwania jest jednak zwracanie interesujących ofert na podstawie słów kluczowych podanych przez użytkownika w wskazanym polu tekstowym.

Proces wyszukiwania zaczyna się od ograniczenia zakresu dokumentów, dla których ma być on wykonany. Następnie przekazywane są słowa kluczowe (wraz z synonimami, które znajdują się dla nich w bazie - o ile się takie znajdują), które będziemy chcieli wyszukać w bazie. Dla konkretnej taryfy znajdującej się w tabeli `Tarrif` sprawdzamy, czy w tabeli `Search_Term` znajdują się słowa lub synonimy, powiązane z danym wierszem, a które użytkownik zadał w zapytaniu.

Jeżeli sprawdzana taryfa jest powiązana ze wszystkimi słowami kluczowymi, to dodajemy ją do listy odnalezionych taryf. Po sprawdzeniu wszystkich interesujących nas taryf, dla każdej z nich wywoływana jest funkcja, która odczyta opis konkretnej taryfy. Należy tu zaznaczyć, że proces odnajdywania tytułów taryf odbywa się tylko dla elementów znajdujących się na danej, wyświetlanej stronie. Proces ten odbywa się rekurencyjnie i polega na stworzeniu listy elementów pochodzących z tabeli `Tarrif_title`. Dla tytułu taryfy, która jest bezpośrednio powiązana z daną opłatą, odszukiwany jest element nadrzędny (rodzic) w tabeli `Tarrif_title`. Proces ten trwa aż do odnalezienia elementu, który w kolumnie `Parent_tariff_title_ID` ma wartość `NULL` - taką wartość mają tylko główne elementy tej tabeli.

Mając listę tytułów jesteśmy w stanie odtworzyć całą nazwę taryfy. Mając więc jej tytuł oraz cenę zwracany jest obiekt przechowujący te dane, a proces wyszukiwania tytułu zaczyna się ponownie dla kolejnej opłaty.

## 4.3 Pseudokody funkcji głównej i pomocniczych

Główna funkcja przeszukująca:

```
def search_tariffs:
    #ograniczamy zakres wyszukiwania do
    #okreslonych dokumentow
    documents_ids = get_matching_doument_ids()

    search_terms = get_search_terms_with_their_synonyms()

    #jesli nie mozna zmapowac szukanych slow
    if all search_terms not join with tarrifs:
        #zwracana jest pusta strona
        return Page.empty()

    #pobierane sa id zindeksowanych slow
    search_terms_ids = get_ids(search_terms)

    #zwracane sa te taryfy, ktore lacza sie
    #z szukanymi slowami
    found_tariffs = find_tarrifs(search_terms_ids)

    #wywolanie dodatkowej funkcji
    search_result = tariffs_to_tariffs_results(found_tariffs)
    return Page(search_result.toDTO)
```

Listing 3: Główna funkcja przeszukująca

Pomocnicza funkcja `tariffs_to_tariffs_results(*)`:

```
def tariffs_to_tariffs_results(found_tariffs):
    #dla kazdej taryfy w znalezionych taryfach
    for tariff in found_tariffs:
        #wywolaj funkcje zwracajaca rezultat
        return tariffs_to_tariffs_result(tariff)
```

Listing 4: Funkcja pomocnicza

Pomocnicza funkcja `tariffs_to_tariffs_result(*)`:

```
def tariffs_to_tariffs_result(tariff):
    titles = []
    #do current przypisany zostaje
    #fragment tytulu oplaty bezposrednio
    #z nia powiazany
    current = tariff.tariff_title

    #dopoki tytul ten nie jest pusty
    while current is not None:
        #dodawaj kolejne fragmenty tytulow
```

```
        #z bazy w celu odbudowania pelnego
        #tytulu taryfy
        titles.append(current)

        #do current przypisany zostaje
        #tytul elementu nadrzednego z
        #tabeli Tariff_title

        #current == None jesli
        #poprzedni current nie mial elementu
        #nadrzednego, co oznaczalo odnalezienie
        #pelnego tytulu taryfy
        current = current.parent_tariff_title

    return TariffSearchResult(
        #wartosc -1 dlatego, gdyz tytul budowalismy
        #od najnizszego elementu w hierarchi

        tariff_titles=titles[::-1],
        tariff=tariff
    )
```

Listing 5: Funkcja pomocnicza

## 4.4 Wersje dokumentów

Przy wywoływaniu parsowania istnieje kilka możliwości zdefiniowania, jaki dokument tworzymy/aktualizujemy. Należy podać id sekcji (ten atrybut jest obowiązkowy, jako zabezpieczenie dla sytuacji, gdzie nie zostanie podane nic więcej). Opcjonalnie podajemy document_id (wtedy jawnie wskazywane jest, który dokument zostaje aktualizowany, reszta atrybutów nie będzie brana pod uwagę) albo document_name (wtedy jawnie definiujemy nazwę dokumentu, który stworzymy lub zaktualizujemy). Jeśli id lub nazwy nie podamy - serwis będzie szukał po nazwie pliku.

Podczas parsowania dokumentu są 4 możliwe sytuacje, które mogą wystąpić:

- Dodany został dokument odnoszący się do sekcji, która nie posiadała jeszcze żadnego zdefiniowanego dokumentu. W takim wypadku w tabeli `Document` jako datę startową ustawiana jest wartość odczytana z dokumentu, a jako datę końcową wartość `NULL`.
- Dodany został dokument odnoszący się do sekcji, która posiadała już dokument, który obowiązywał przed obecnie dodawanym dokumentem. W takim wypadku dla nowego dokumentu postępowanie wygląda tak samo, jak w poprzedniej sytuacji, natomiast dla starego dokumentu data zakończenia ustawiana jest na datę o dzień wcześniejszą niż data startowa dodawanego dokumentu.
- Dodany został dokument odnoszący się do sekcji, która posiada już dokument, który zawierał nowsze taryfy od tych w parsowanym dokumencie. W tej sytuacji w tabeli `Document` jako datę startową ustawiana jest wartość odczytana z dokumentu, a data zakończenia ustawiana jest na datę o dzień wcześniejszą niż data startowa dokumentu, który znajdował się już w bazie.
- Dodany został dokument odnoszący się do sekcji, która posiadała już dokumenty. W tej sytuacji wynik parsowania zawiera dane o taryfach, których daty obowiązywania zawierają się w przedziale czasowym, który w bazie danych był pokryty już przez 2 inne dokumenty (dla łatwiejszego zrozumienia: pierwszy dokument w bazie zawierał taryfy obowiązujące od 2018 do 2020 roku, drugi od 2020 roku do teraźniejszości, a dodany dokument pokrywał zakres czasu od 2019 do 2020 roku).

# 5 API

Poniżej zostały opisane wszystkie endpointy, udostępniane przez kontroler.

## 5.1 Dokumenty

`GET /documents`

- wyszukiwanie dokumentów znajdujących się w danej sekcji
- parametry: `section`

`POST /documents/create`

- tworzenie dokumentu
- body: JSON zawierający dane o dokumencie, przykład:

```
{
    "name": "Rachunki bankowe",
    "section_id": 2,
    "begin_validity_date": "2019-12-01",
    "end_validity_date": "2020-02-28"
}
```

## 5.2 Sekcje

`GET /sections`

- wyszukiwanie sekcji, których nadrzędnych sekcją jest `parent`
- parametry: `parent`

`POST /sections/create`

- tworzenie sekcji
- body: JSON zawierający dane o sekcji, przykład

```
{
    "name": "Przelewy zagraniczne",
    "parent_section": 2
}
```

## 5.3 Produkty

`GET /products`

- wyszukiwanie wszystkich produktów

## 5.4 Taryfy

`POST /tariffs/parse/upload`

- parsowanie dokumentu
- `Content-Type` formularza to `multipart/form-data`, zawiera on plik `file`, natomiast parametry formularza HTML to `section_id`, `document_id`, `document_name` (należy podać albo `document_id`, albo `document_name`)

`GET /tariffs/search`

- wyszukiwanie taryf po frazie, sekcji, produkcie i dacie, wykorzystywana jest paginacja
- parametry: `text`, `section`, `products`, `date`, `page`, `items`

## 5.5 Słownik synonimów

`POST /thesaurus`

- aktualizowanie słownika synonimów
- body: JSON zawierający słowa bliskoznaczne oraz słowa ignorowane, przykład

```
{
    "ignored_terms": ["o", "i", "w", "a"],
    "thesaurus": [
        "rachunek": ["rachunku", "konto", "konta"],
        "pko": ["pkobp", "bank", "banku"]
    ]
}
```

# 6 Struktura i uruchomienie projektu

## 6.1 Struktura

Struktura projektu prezentuje się następująco:

- katalog `tariff_parser` - zawiera implementację parsera dokumentów PDF. Jest on napisany w Pythonie przy użyciu biblioteki `pdfplumber`.
- katalog `tariff_dao` - zawiera metody pozwalające na pobieranie, dodawanie, usuwanie i modyfikowanie danych w bazie danych. Jest zaimplementowany w Pythonie z wykorzystaniem biblioteki `SQLAlchemy`. Katalog ten zawiera podkatalog `Entity`, w którym zdefiniowane są wszystkie encje przechowywane w bazie.
- katalog `tariff_service` - zawiera implementację algorytmu wyszukiwania żądanej frazy. Jest on napisany w języku Python.
- katalog `controller` - zawiera kontrolery REST-owe, które wystawiają endpointy do pobierania danych z bazy danych oraz zarządzania nimi. Jest on zaimplementowany we Flasku - frameworku aplikacji webowych napisanym w języku Python.
- katalog `tariff_ui` - zawiera implementację aplikacji frontendowej napisanej w Angularze.
- katalog `model` - zawiera wszystkie zdefiniowane klasy, które służą do komunikacji pomiędzy parserem, a serwisem do wyszukiwania frazy.

Do stworzenia bazy danych wykorzystaliśmy MySQL.

## 6.2 Uruchomienie

Do uruchomienia projektu wymagane jest zainstalowanie:

- Python 3.7
- Node.js 16.15

Uruchomienie - krok po kroku:
(Uwaga: w przypadku przeniesienia rozwiązania na chmurę wymagane będzie dostosowanie poniższej konfiguracji.)

1. znajdując się w `katalogu głównym`, należy wykonać:

```
pip install -r requirements.txt
```

2. w pliku `config.py` znajdującym się w `katalogu głównym` zawarta jest konfiguracja połączenia się z bazą danych i frontendem. `DB_CONNECTION_URL` to zmienna przechowująca URL podłączenia do bazy, `FRONTEND_ORIGIN` to adres aplikacji frontendowej.
3. z poziomu katalogu `controller` uruchomić aplikację komendą:

```
python3 app.py
```

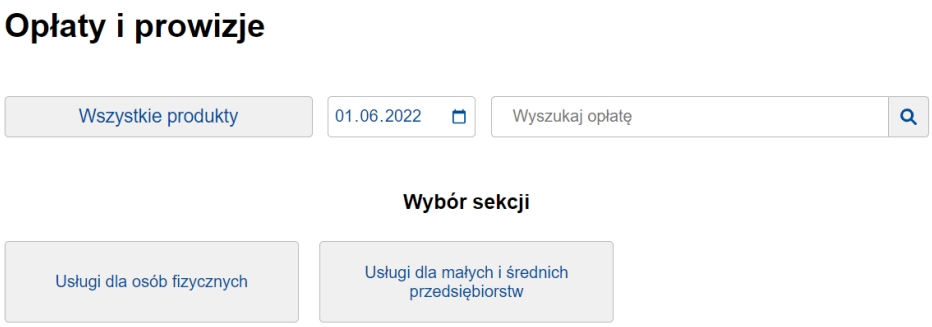
4. z poziomu katalogu `tariff_ui` należy pobrać zależności i uruchomić aplikację poleceniami:

```
npm install
npm run start
```

   W pliku `local.conf.json` w katalogu `proxy` znajduje się konfiguracja przekierowania requestów z frontendu do backendu. Jeśli np. zmieni się adres aplikacji backendowej, należy dostosować również konfigurację w tym pliku.

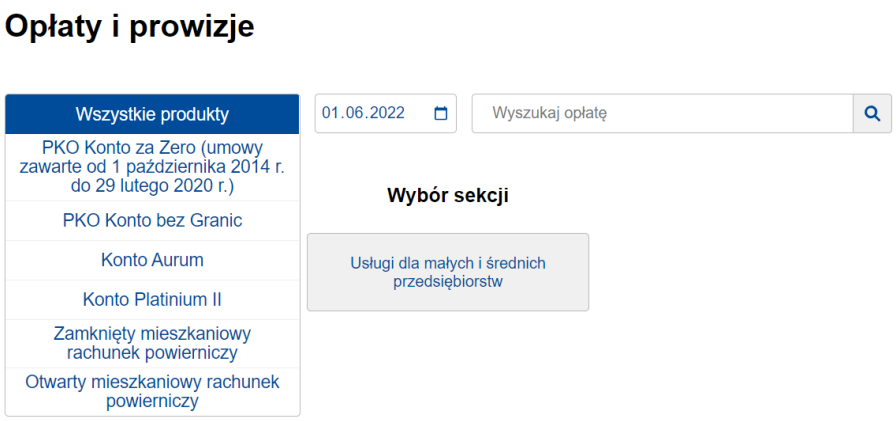
# 7 Prototyp wyszukiwarki

Finalnym etapem naszego projektu przygotowanie aplikacji frontendowej w celu wizualnego przedstawienia przygotowanego rozwiązania.



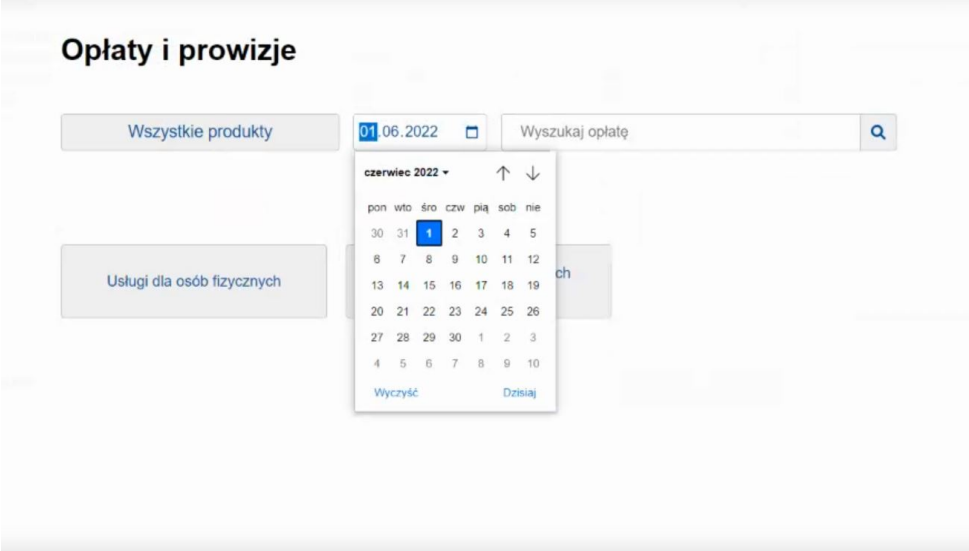

Rysunek 9: Wyszukiwarka.

W celu wyboru produktu, po którym chcemy filtrować, należy wybrać odpowiedni z listy `Wszystkie produkty`.




Rysunek 10: Wybór produktu.

Wyszukiwarka ma zaimplementowaną filtrację po dacie - wybierając konkretną datę, możemy wyszukać opłaty i prowizje z zadanego przedziału czasowego.




Rysunek 11: Wybór daty.

**Znalezione opłaty i prowizje**

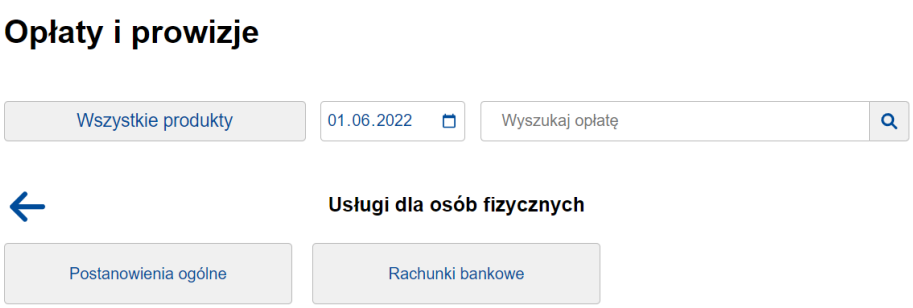
| | |
|---|---|
| **Opłata za:** | Wyciągi bankowe :<br>- odbierane w oddziale :<br>- za pierwszy wyciąg przygotowany w danym miesiącu |
| **Konto:** | Zamknięty mieszkaniowy rachunek powierniczy |
| **Obowiązuje:** | Od 2019-12-15 |
| **Opłata:** | 0,00 |

| | |
|---|---|
| **Opłata za:** | Wyciągi bankowe :<br>- odbierane w oddziale :<br>- za pierwszy wyciąg przygotowany w danym miesiącu |
| **Konto:** | Otwarty mieszkaniowy rachunek powierniczy |
| **Obowiązuje:** | Od 2019-12-15 |
| **Opłata:** | 0,00 |

| | |
|---|---|
| **Opłata za:** | Wyciągi bankowe :<br>- odbierane w oddziale :<br>- za każdy kolejny wyciąg przygotowany w danym miesiącu |
| **Konto:** | Zamknięty mieszkaniowy rachunek powierniczy |
| **Obowiązuje:** | Od 2019-12-15 |
| **Opłata:** | 10,00 |

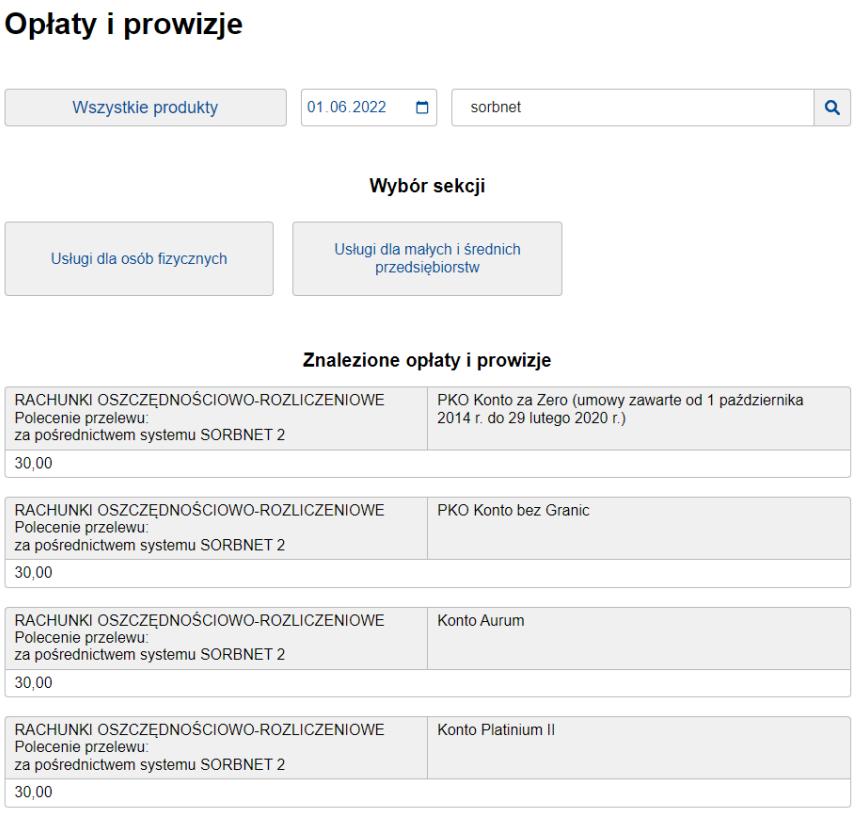
Rysunek 12: Wyniki z wybraną datą.

Oprócz tego, istnieje możliwość zawężenia obszaru wyszukiwania do opłat i prowizji dotyczących odpowiednich sekcji.



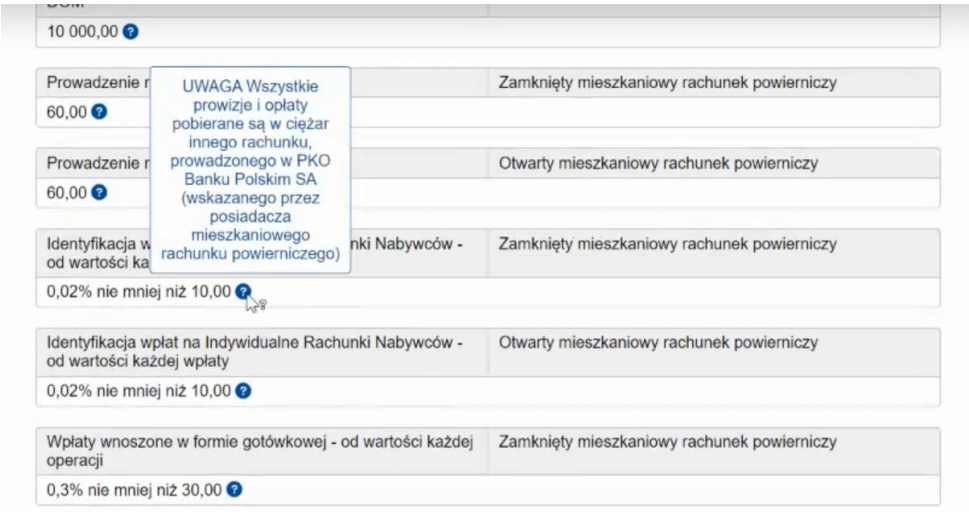

Rysunek 13: Wybór sekcji.

Najważniejszym elementem prototypu wyszukiwarki było zaimplementowanie mechanizmu pozwalającego na filtrowanie opłat i prowizji przy użyciu słów kluczowych podanych przez użytkownika. Opis działania tego mechanizmu znajduje się w rozdziale 4.2. W polu wyszukiwania wpisujemy żądaną frazę. W sekcji `Znalezione opłaty i prowizje` znajdują się wyniki wyszukiwania.




Rysunek 14: Wynik wyszukiwania.

Mogą one zawierać również odnośniki, których treść przedstawiona jest za pomocą tooltipa (chmurki z informacją).




Rysunek 15: Odnośniki.

Przy wyszukiwaniu danych wykorzystywana jest paginacja, co pozwala na zwiększenie wydajności oraz bardziej przejrzyste wyświetlanie dużej liczby wyników.

| | |
|---|---|
| **Opłata za:** | RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE :<br>- Polecenie przelewu: :<br>- standardowe: :<br>- za pośrednictwem usługi bankowości elektronicznej |
| **Konto:** | Konto Platinium II |
| **Obowiązuje:** | Od 2020-03-01 |
| **Opłata:** | 0,00 |

| | |
|---|---|
| **Opłata za:** | RACHUNKI OSZCZĘDNOŚCIOWO-ROZLICZENIOWE :<br>- Polecenie przelewu: :<br>- natychmiastowe: :<br>- w oddziale PKO BP SA lub za pośrednictwem usługi bankowości telefonicznej |
| **Konto:** | PKO Konto za Zero (umowy zawarte od 1 października 2014 r. do 29 lutego 2020 r.) |
| **Obowiązuje:** | Od 2020-03-01 |
| **Opłata:** | 11,99 |

1 2 3 4 5 6 7 8 9

Rysunek 16: Paginacja.

# 8 Możliwości dalszego rozwoju

Przygotowany projekt ma na celu przedstawienie prototypu przykładowej wyszukiwarki taryf, której celem jest zaprezentowanie możliwości takiego rozwiązania. Głównym zadaniem wyszukiwarki byłoby skrócenie czasu wyszukiwania opłat/prowizji, a przez to zwiększenie efektywności korzystania ze strony.

Istnieje wiele możliwości dalszego rozwoju wyszukiwarki. Jednym z pomysłów na ulepszenie stworzonego rozwiązania jest wykorzystanie przetwarzania języka naturalnego, co pozwoliłoby na lepsze dopasowanie wyszukiwanych przez klienta fraz. Obecnie wyszukiwarka jest ograniczona jeśli chodzi o np. odmianę słów kluczowych, natomiast użycie przetwarzania języka naturalnego rozwiązałoby ten problem.