Projekt zaliczeniowy

Proces ETL

Grupa projektowa: 25

Imię	Nazwisko	Numer albumu	Grupa dziekańska	Wkład w prace nad projektem[1]	Udział procentowy
Ewa	Bąk	192207	WZISS2-1111	Stworzenie skryptu ETL, Frontend, komunikacja między podstronami, stworzenie formularza dla użytkownika, dokumentacja, stworzenie skryptu Extract	40%
Gabriela	Lenard	193670	WZISS2-1111	Stworzenie tabeli i połączenie z bazą danych, skrypt zapisujący rekordy do tabeli, skrypt wyświetlający zawartość bazy danych, operacje na tabeli, dokumentacja	25%
Karol	Skoczyk	194677	WZISS2-1111	Stworzenie skryptu ETL, Stworzenie pętli pobierającej konkretną ilość danych, stworzenie formularza dla użytkownika, dokumentacja	35%

___/70 pkt

[1] proszę wymienić konkretne zadania

Dokumentacja techniczna

Nazwy i wersje użytych technologii:

- Język programowania wykorzystany w skrypcie - Python 3.7.1
- Web framework - Flask 1.0.2
- Framework - Bootstrap 4.1
- System zarządzania bazą danych - PostgreSQL 11.1

Informacje na temat środowiska:

- Wykorzystane biblioteki:
    - bs4
    - requests
    - pandas
    - psycopg2
- Wymagania sprzętowe: procesor taktowany częstotliwością co najmniej 1 GHz, 0,5 GB pamięci na dysku twardym, 1 GB pamięci operacyjnej.

Linki do oprogramowania tworzącego środowisko:

- https://www.python.org/downloads/release/python-371/
- https://pypi.org/project/Flask/1.0.2/
- https://pypi.org/project/beautifulsoup4/
- https://pypi.org/project/pandas/
- https://pypi.org/project/requests/
- https://getbootstrap.com/
- https://www.postgresql.org/download/

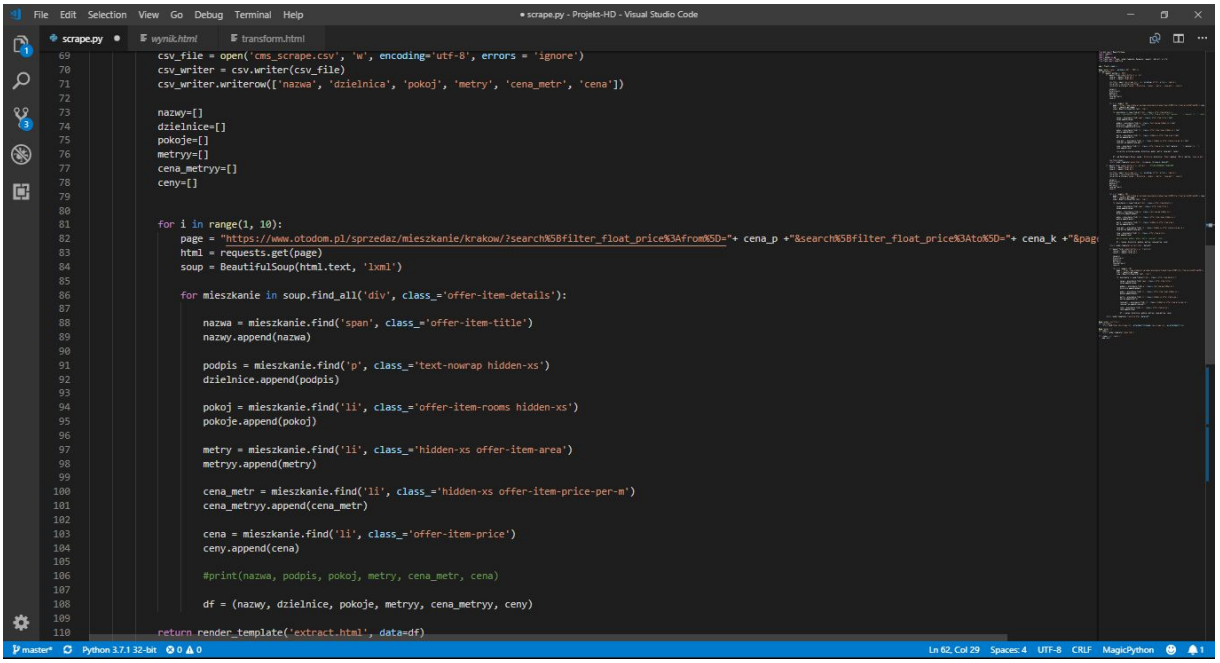
Struktura aplikacji:

- scrape.py - plik wywołujący funkcje zadane w poleceniu projektu - pełny proces ETL oraz osobne procesy E, T oraz L.
- CrTable.py - plik tworzący tabelę w bazie danych
- index.html - plik zawierający strukturę strony głównej aplikacji
- wynik.html - plik zawierający strukturę podstrony po wybraniu opcji pełnego procesu ETL
- extract.html - plik zawierający strukturę podstrony po wybraniu opcji Extract, która zawiera wyodrębnione dane w nieprzetworzonej postaci
- transform.html - plik zawierający strukturę podstrony po wybraniu opcji Transform, która zawiera dane w postaci tabeli
- clean.html - plik zawierający strukturę podstrony po wyczyszczeniu bazy danych
- style.css - plik zawierający szczegółowe opcje dotyczące wyglądu aplikacji.

```
    if request.method == 'POST':
        if request.form['submit_button'] == 'etl':
            cena_p = request.form['od']
            cena_k = request.form['do']

            csv_file = open('cms_scrape.csv', 'w', encoding='utf-8', errors = 'ignore')
            csv_writer = csv.writer(csv_file)
            csv_writer.writerow(['nazwa', 'dzielnica', 'pokoj', 'metry', 'cena_metr', 'cena'])

            nazwy=[]
            dzielnice=[]
            pokoje=[]
            metryy=[]
            cena_metryy=[]
            ceny=[]
            count = 0

            connection = psycopg2.connect(user = "postgres", password = "haslo", database = "postgres")
            cursor = connection.cursor()
            sql_add = """INSERT INTO otodom ("Tytul", "Dzielnica", "Liczba pokoi", "Metraz", "Cena za metr", "Cena") VALUES (%s,%s,%s,%s,%s,%s) ON CONFLICT DO NOTHING"""
            sql_display = """SELECT * FROM otodom"""

            for i in range(1, 10):
                page = "https://www.otodom.pl/sprzedaz/mieszkanie/krakow/?search%5Bfilter_float_price%3Afrom%5D="+ cena_p +"&search%5Bfilter_float_price%3Ato%5D="+ cena_k +"&page={}".format(i)
                html = requests.get(page)
                soup = BeautifulSoup(html.text, 'lxml')

                for mieszkanie in soup.find_all('div', class_='offer-item-details'):

                    nazwa = mieszkanie.find('span', class_='offer-item-title').text
                    nazwy.append(nazwa)

                    podpis = mieszkanie.find('p', class_='text-nowrap hidden-xs').text
                    dzielnica = podpis.split(':')[1]
                    dzielnice.append(dzielnica)

                    pokoj = mieszkanie.find('li', class_='offer-item-rooms hidden-xs').text
                    pokoje.append(pokoj)

                    metry = mieszkanie.find('li', class_='hidden-xs offer-item-area').text
                    metryy.append(metry)

                    cena_metr = mieszkanie.find('li', class_='hidden-xs offer-item-price-per-m').text
                    cena_metryy.append(cena_metr)

                    cena = mieszkanie.find('li', class_='offer-item-price').text.replace(' ','').replace('\n','')
                    ceny.append(cena)

                    to_insert = (nazwa, dzielnica, pokoj, metry, cena_metr, cena)
                    cursor.execute(sql_add, to_insert)
                    connection.commit()
```

```
                    if cursor.rowcount==1:
                        count += 1

                    csv_writer.writerow([nazwa, dzielnica, pokoj, metry, cena_metr, cena])

                df = pd.DataFrame({'Nazwa':nazwy, 'Dzielnica':dzielnice, 'Pokoj':pokoje, 'Metry':metryy, 'Cena za metr':cena_metryy, 'Cena':ceny})

            csv_file.close()

            cursor.execute(sql_display)
            datat = cursor.fetchall()
            cursor.close()
            connection.close()

            return render_template('wynik.html', c1=cena_p, c2=cena_k, data=df, rec=count, datat=datat)
```

Funkcja odpowiedzialna za wykonanie pełnego procesu ETL. Zwraca tabelę zawierającą ogłoszenia zgodne z kryteriami użytkownika wpisanymi na stronie startowej aplikacji. Po wyciągnięciu danych ze strony internetowej istnieje również możliwość pobrania ich do pliku CSV. Wygląd strony końcowej warunkuje plik wynik.html.



Funkcja odpowiedzialna za wykonanie procesu Extract. Struktura kodu jest niemal identyczna jak w poprzednim przypadku z różnicą na wyświetlenie danych, które są przedstawione w formie nieprzetworzonej. Wygląd strony warunkuje plik extract.html.

```
@app.route('/csv_file')
def csv_file():
    return send_file('cms_scrape.csv', attachment_filename='cms_scrape.csv', as_attachment=True)
```

Skrypt odpowiedzialny za określenie ścieżki dostępu, która umożliwia pobranie pliku z danymi ogłoszeń w formacie CSV.

```
@app.route('/clean')
def clean():
    connection = psycopg2.connect(user = "postgres", password = "haslo", database = "postgres")
    cursor = connection.cursor()
    sql_del = """DELETE FROM otodom"""

    cursor.execute(sql_del)
    connection.commit()
    cursor.close()
    connection.close()
    return render_template('clean.html')
```

Skrypt odpowiedzialny za usunięcie wszystkich rekordów z bazy danych

Instrukcja obsługi aplikacji:

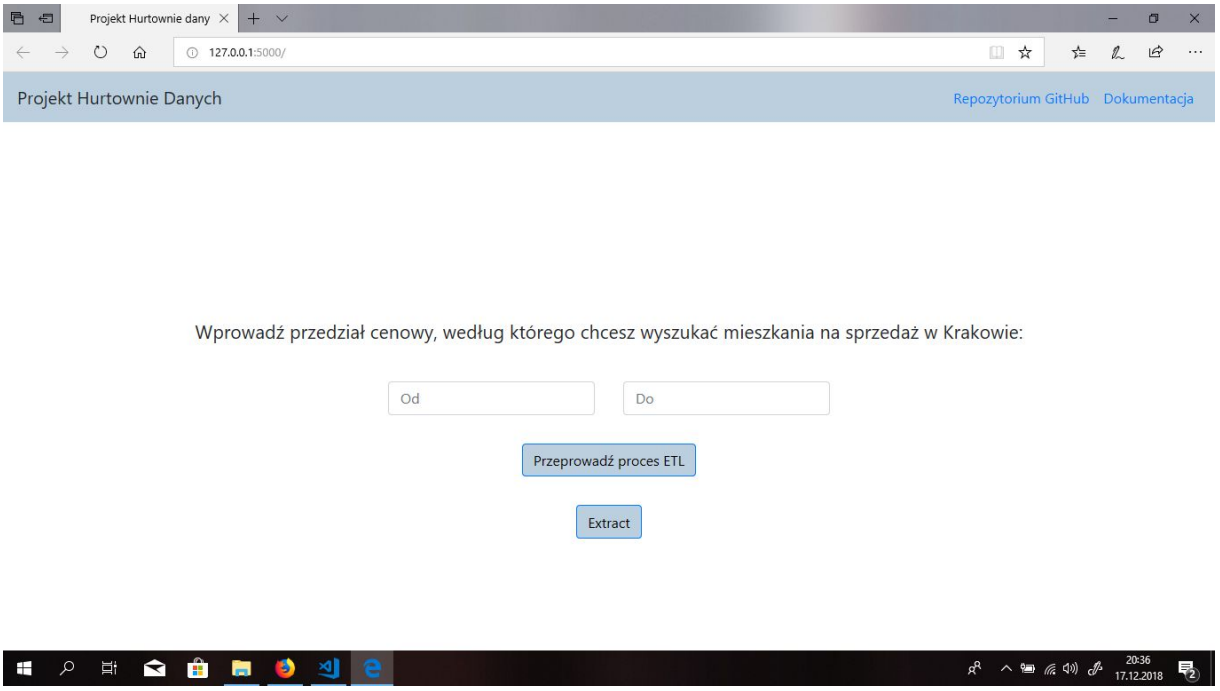
Instrukcja uruchomienia aplikacji:

1. Pobranie wszystkich elementów środowiska aplikacji
2. Pobranie skompresowanego pliku ZIP

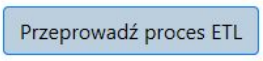
3. Wyodrębnienie plików z pliku ZIP
4. Uruchomienie aplikacji poprzez:
    4.1. Uruchomienie skryptu do tworzenia tabeli w programie PostgreSQL
    4.2. Otworzenie folderu aplikacji w edytorze kodu takim jak na przykład Visual Studio Code
        4.2.1. Uruchomienie pliku CrTable.py
        4.2.2. Uruchomienie pliku scrape.py
    4.3. Lokalizacja folderu aplikacji za pomocą Windows Command Line/ konsoli Linux
        4.3.1. Uruchomienie pliku CrTable.py
        4.3.2. Uruchomienie pliku scrape.py

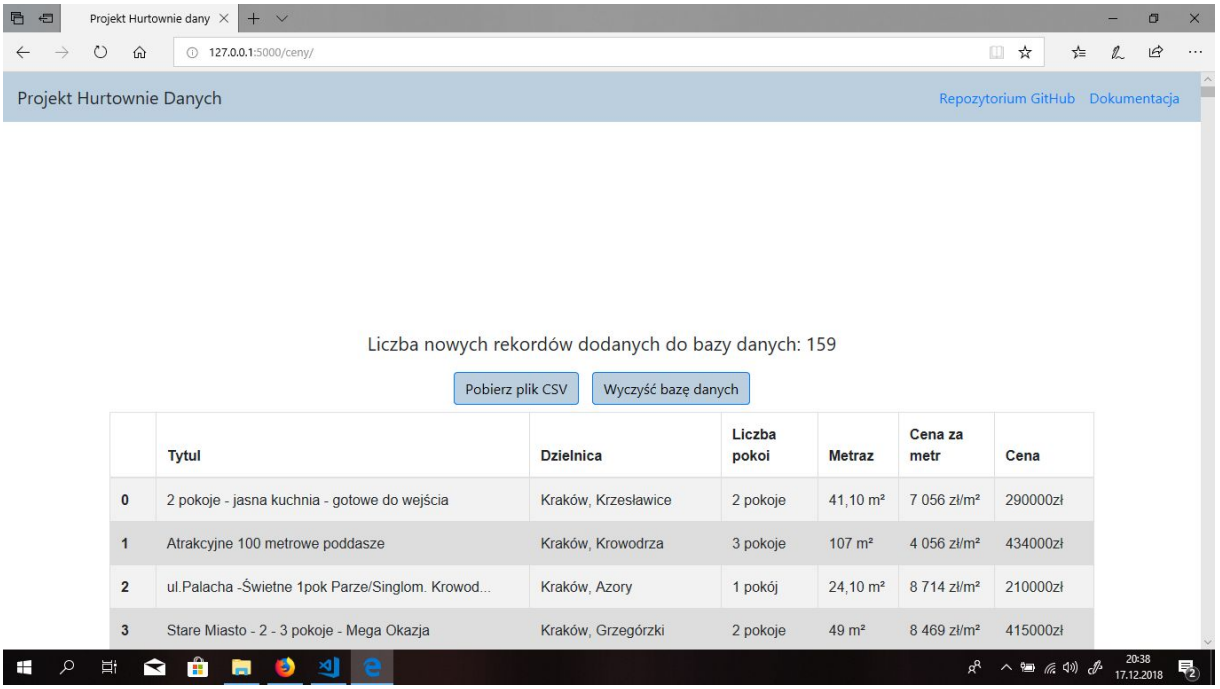
Opis funkcjonalności aplikacji:

Po wykonaniu powyższych kroków, edytor kodu lub konsola dowolnego systemu skieruje użytkownika do strony głównej aplikacji w przeglądarce internetowej.

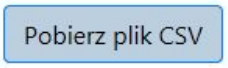


Wyświetlone są na niej pola do uzupełnienia kryteriów cenowych przy poszukiwaniu mieszkania, a także przycisk wykonania procesu ETL oraz Extract. W górnym rogu znajdują się również odnośniki do pobrania pliku dokumentacji oraz do repozytorium z kodem aplikacji.

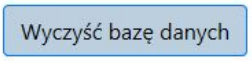


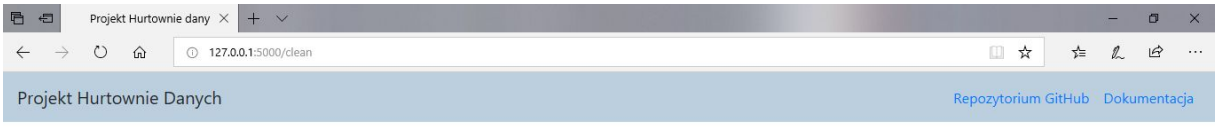


Wybór wykonania całego procesu ETL poskutkuje przejściem do podstrony, w której znajduje się tabela z wyszczególnionymi ofertami wedle (lub bez podanego) kryterium wyboru. Oprócz ceny, w tabeli widnieje nazwa ogłoszenia, dzielnica, w jakiej znajduje się oferowana nieruchomość, liczba pokoi, metraż mieszkania oraz jego cena za metr kwadratowy.



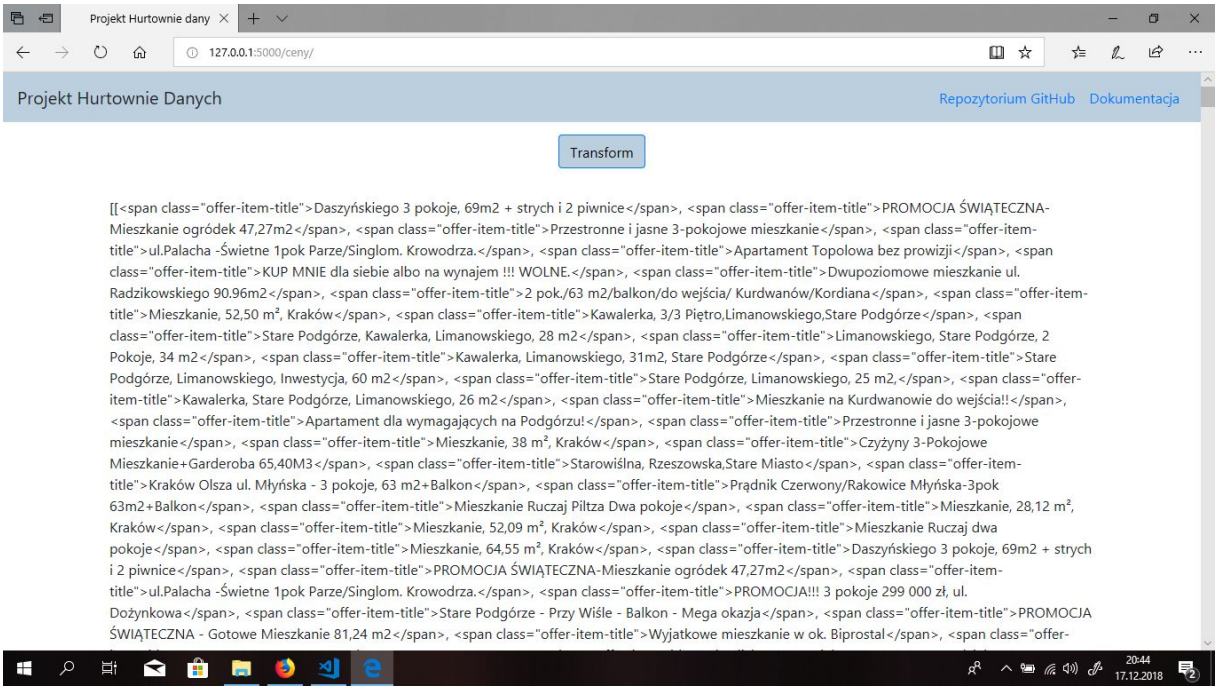
Przycisk CSV służy do pobrania zebranych danych i zapisu ich na dysk komputera. Możliwe jest przeglądanie tych danych i edytowanie według własnych dowolnych potrzeb w programie typu Excel.





Baza danych została pomyślnie wyczyszczona!

Przycisk “Wyczyść bazę danych” pozwala na usunięcie z bazy danych przechowywanych w niej danych.

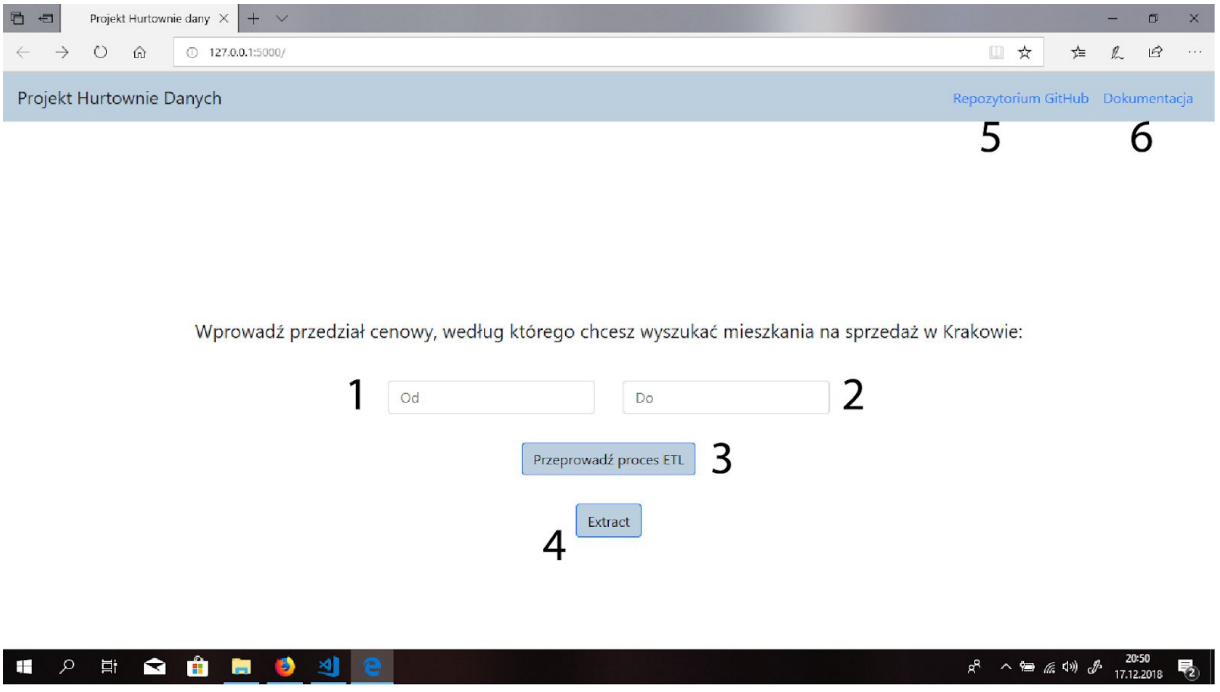


Wracając do ekranu głównego, wybór opcji Extract przeniesie użytkownika do podstrony, w której wykonany zostanie pierwszy etap procesu ETL i wyświetli się jego wynik w formie kodu Html.

Opis scenariuszy użycia aplikacji:

1. Użytkownik zainteresowany jest wyszukaniem mieszkań na sprzedaż w obrębie miasta Kraków
2. Uzupełnia pola “od” oraz “do” w interesujące go kwoty
3. Naciska przycisk “Przeprowadź proces ETL”, strona przekierowuje go do podstrony zawierającej spis ogłoszeń pasujących do kryteriów
4. Użytkownik ma możliwość pobrania pliku CSV z interesującymi go danymi

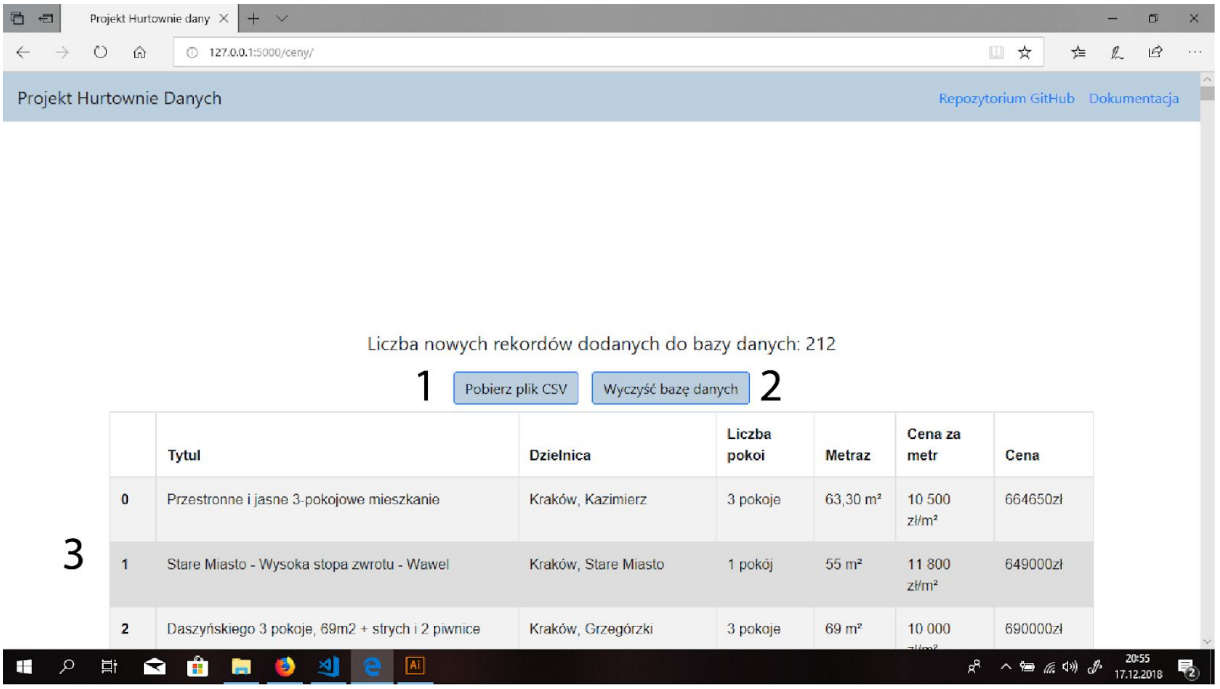
Opis widoków okna aplikacji:



Strona główna

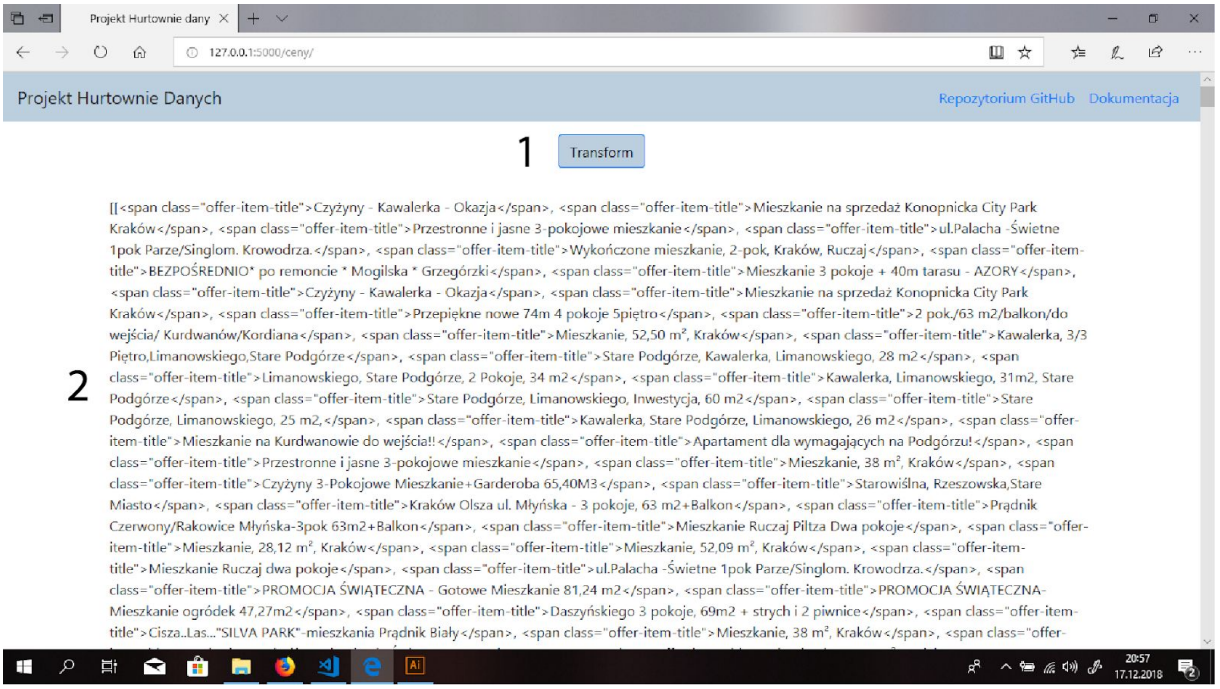
1. Pole formularza ceny początkowej

2. Pole formularza ceny końcowej
3. Przycisk przeprowadzania procesu ETL
4. Przycisk przeprowadzenia procesu Extract
5. Odnośnik do repozytorium projektu
6. Odnośnik do pobrania dokumentacji



Strona procesu ETL

1. Odnośnik do pobrania pliku CSV
2. Przycisk czyszczący bazę danych
3. Tabela zawierająca pobrane dane



Strona procesu Extract

1. Przycisk do przeprowadzenia procesu Transform
2. Pobrane dane w formie kodu Html

Model danych użytych w projekcie:

Model danych strukturalny:

Nazwa pliku	Nazwa zmiennej	Opis zmiennej
	cena_p	cena początkowa oferty, która jest pobierana od użytkownika
	cena_k	cena końcowa oferty, która jest pobierana od użytkownika
	nazwy nazwa	przechowuje nazwę/tytuł ogłoszenia
	dzielnice podpis	przechowuje dokładniejszą lokalizację miejsca z ogłoszenia
	pokoj pokoje	przechowuje ilość pokojów, które oferuje miejsce z ogłoszenia
	metry metryy	przechowuje ilość metrów kwadratowych mieszkania w ofercie

scrape.py	cena_metr cena_meryy	przechowuje cenę jednego metra kwadratowego mieszkania w ofercie
	cena ceny	przechowuje cenę mieszkania zamieszczonego na stronie
	page	przechowuje adres strony, z której pobierane są dane
	html	pobiera adres strony
	soup	definiuje podstawowy interfejs strony
	mieszkanie	zbiór wszystkich danych pojedynczego ogłoszenia
	df	wyświetla dane pobrane ze strony internetowej
	count	przechowuje numer nowo dodanych do bazy rekordów
	datat	przechowuje wiersze pobrane z tabeli
	column_names	Przechowuje nagłówki tabeli
	dt	Wyświetla tabelę zawierającą pobrane rekordy

Nazwa funkcji	Wartość zwracana	Przyjmowane parametry	Opis działania
scrape.py			
getValue()	strona wynik.html wartości c1, c2, data, rec, datat	brak	Funkcja przeprowadza proces ETL
app.route()	plik cms_scrape.csv	/csv_file	Funkcja przekierowuje do pobrania pliku cms_scrape.csv
app.route()	strona clean.html	/clean	Funkcja wykonuje skrypt usuwający wszystkie rekordy z bazy danych
app.route()	plik 25_Projekt.pdf	/dokumentacja_file	Funkcja przekierowuje do pobrania pliku 25_Projekt.pdf
app.route()	strona index.html	/	Przekierowanie podczas uruchamiania skryptu do strony głównej