

**Mateusz Baczek**

Numer albumu: 101252

**Klaudiusz Florek**

Numer albumu: 101335

Informatyka, studia niestacjonarne I stopnia, semestr 4

Bazy danych pt.:

**PORTAL OGŁOSZENIOWY**



Projekt wykonywany w ramach przedmiotu: Bazy danych

Rzeszów, 2018/2019

**Spis treści**

**Założenia projektu**

Portal „myAdvert” ma być odpowiednikiem takich serwisów jak Allegro, olx, gdzie użytkownicy mają możliwość umieszczania swoich ofert sprzedaży. Użytkownicy podają jedynie podstawowe informacje bez potrzeby podawania dokładnych informacji takich jak: dokładny adres zamieszkania, dane osobowe imię, nazwisko, data urodzenia. Użytkownicy mogą dodawać swoje ogłoszenia i zarządzać z nimi z poziomu Panelu klienta. Same ogłoszenia są natomiast podzielone na kategorie, a każda kategoria jest osobnym schematem w bazie danych. Pozwala to na łatwe zarządzanie ofertami, większą skalowalność, przez możliwość rozszerzania serwisu o dodatkowe kategorie bez potrzeby wcześniejszego przeanalizowania ilości atrybutów mogących pojawiać się w ogłoszeniach. Użytkownik podczas wyszukiwania interesujących go ogłoszeń nie musi martwić się o pojawianie się ofert sprzedaży z kategorii, które go nie interesują. Ułatwia to także zarządzanie ofertami oraz rozszerzanie poszczególnych kategorii o dodatkowe atrybuty. Projekt jest z założenia jest pomysłem na problematyczną czasami funkcjonalność serwisu allegro, gdzie wyszukiwanie przedmiotów wewnątrz poszczególnych kategorii nie zawsze daje satysfakcjonujące dla nas wyniki z powodu pojawiania się ofert z kategorii, które nas nie interesują. Sam interfejs ma być prosty i przyjazny w obsłudze. Projekt oparty jest o możliwości jęzka PHP, JavaScript, bazy danych PostgreSQL oraz języka plsql. Przetwarzanie danych w bazie ma się opierać na przygotowanych wcześniej funkcjach bazodanowych bez konieczności pisania bezpośrednich zapytań wewnątrz skryptów PHP.

**Wykorzystane technologie**

**** XAMPP jest najpopularniejszym pakietem serwera WWW dla PHP. Pakiet jest kompletnie darmową, łatwą w instalacji dystrybucją Apache'a, zawierającą MariaDB, PHP i Perl. Paczka XAMPP została tak zbudowana, aby była łatwa w instalacji i w użyciu. Pakiet jest wydawany na licencji GNU General Public License jako darmowy serwer WWW do obsługi dynamicznych stron. Obecnie XAMPP jest dostępny na cztery platformy: Microsoft Windows, Linux, Sun Solaris oraz OS X. Na MS Windows dostępna jest też wersja PortableApps którą można zainstalować np. na Pendrive i przenosić. Może służyć jako środowisko dla testerów, programistów, którzy chcą szybko przetestować skrypty na swoim lokalnym komputerze, bez konieczności uruchamiania na serwerze.

****  PHP to interpretowany skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym. PHP jest najczęściej stosowany do tworzenia skryptów po stronie serwera WWW, ale może być on również używany do przetwarzania danych z poziomu wiersza poleceń, a nawet do pisania programów pracujących w trybie graficznym (np. za pomocą biblioteki GTK+, używając rozszerzenia PHP-GTK). Implementacja PHP wraz z serwerem WWW Apache oraz serwerem baz danych MySQL określana jest jako platforma AMP (w środowisku Linux – LAMP, w Windows – WAMP). Po instalacji pakietu XAMPP i uruchomieniu usługi Apache skrypty wewnątrz folderu *htdocs* są automatycznie interpretowane przez środowisko.

****PostgreSQL to jeden z trzech najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych. Początkowo opracowywany na Uniwersytecie Kalifornijskim w Berkeley i opublikowany pod nazwą Ingres. W miarę rozwoju i zwiększania funkcjonalności, baza danych otrzymała nazwy Postgres95 i ostatecznie PostgreSQL.

 Język wykorzystywany do tworzenia i prezentowania stron internetowych www. Jest rozwinięciem języka HTML 4 i jego XML-owej odmiany (XHTML 1), opracowywane w ramach prac grupy roboczej WHATWG (Web Hypertext Application Technology Working Group) i W3C. HTML5 poza dodaniem nowych elementów, usprawniających tworzenie serwisów i aplikacji internetowych, doprecyzowuje wiele niejasności w specyfikacji HTML 4, dotyczących przede wszystkim sposobu obsługi błędów. Niejasności co do sposobu, w jaki przeglądarki powinny obsługiwać błędy w kodzie HTML są jedną z podstawowych przyczyn, dla której wiele serwisów internetowych, napisanych z naruszeniem specyfikacji, w różnych przeglądarkach działa w inny sposób – w niektórych działając, w innych nie. Dzięki HTML-owi 5 obsługa błędów ma być ta sama we wszystkich przeglądarkach, czyli zły element będzie działać w każdej przeglądarce albo żadnej. HTML5 także stawia na semantykę. Element <div> traci na znaczeniu na rzecz *<header> <main> <article> <aside> <footer> <nav>*, a dodane zostają m.in. *<canvas> <figure> <details> <summary>*. Element *<span>* ma być mniej używany na rzecz *<mark> <output> <var> <u> <s>*.

 Kaskadowe arkusze stylów towarzyszą nam od wielu lat i można przyjąć, że stały się nieodzowną częścią każdej strony WWW, która aspiruje do miana nowoczesnej. Nie da się również ukryć, że druga specyfikacja towarzyszy nam już od blisko 10 lat i jest niemal w całości obsługiwana przez wszystkie liczące się przeglądarki. Arkusz stylów CSS to lista dyrektyw (tzw. reguł) ustalających w jaki sposób ma zostać wyświetlana przez przeglądarkę internetową zawartość wybranego elementu (lub elementów) (X)HTML lub XML. Można w ten sposób opisać wszystkie pojęcia odpowiedzialne za prezentację elementów dokumentów internetowych, takie jak rodzina czcionek, kolor tekstu, marginesy, odstęp międzywierszowy lub nawet pozycja danego elementu względem innych elementów bądź okna przeglądarki.

 Najczęściej spotykanym zastosowaniem języka JavaScript są strony internetowe. Skrypty te służą najczęściej do zapewnienia interakcji poprzez reagowanie na zdarzenia, walidacji danych wprowadzanych w formularzach lub tworzenia złożonych efektów wizualnych. Skrypty JavaScriptu uruchamiane przez strony internetowe mają znacznie ograniczony dostęp do komputera użytkownika. W języku JavaScript można także pisać pełnoprawne aplikacje stąd też duża część frameworków jest oparta właśnie na JavaScript.

 Adobe Photoshop CC to rozbudowany program graficzny przeznaczony do tworzenia i obróbki grafiki rastrowej, będący flagowym produktem firmy Adobe Systems. Program jest dostępny na platformy Windows i macOS. Projekty zapisywane są w formacie plików PSD.

**Model biznesowy**

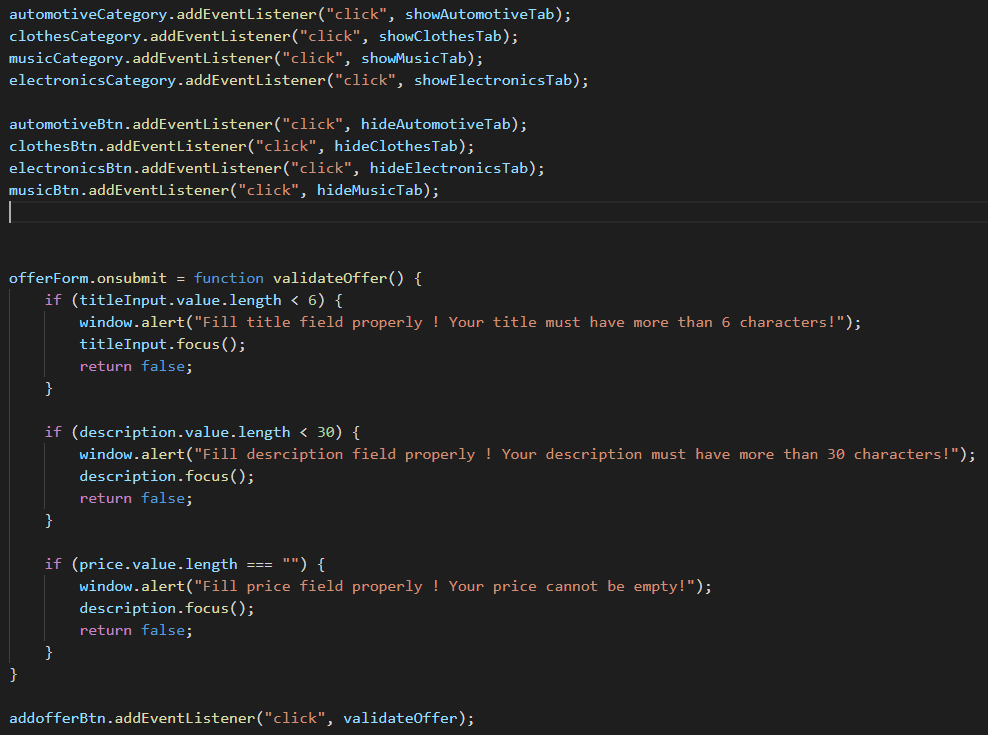
**Funkcje JavaScript**

Wykorzystane w projekcie funkcje JavaScript, odpowiadają kolejno za: funkcjonalność strony, walidację formularzy oraz sortowanie ogłoszeń. Dodatkowo w projekcie wykorzystano bibliotekę jQuery, którą wykorzystano do budowy slidera, zawierającego zdjęcia ogłoszeń.

**Przykłady i omówienie funkcji JavaScript**



Pierwsze linie kodu javascript to zazwyczaj pobranie elementów strony (model DOM), przypisanie ich do zmiennych i operowanie na nich. Funkcja toggleShowTab odpowiada za wysunięcie panelu logowania.



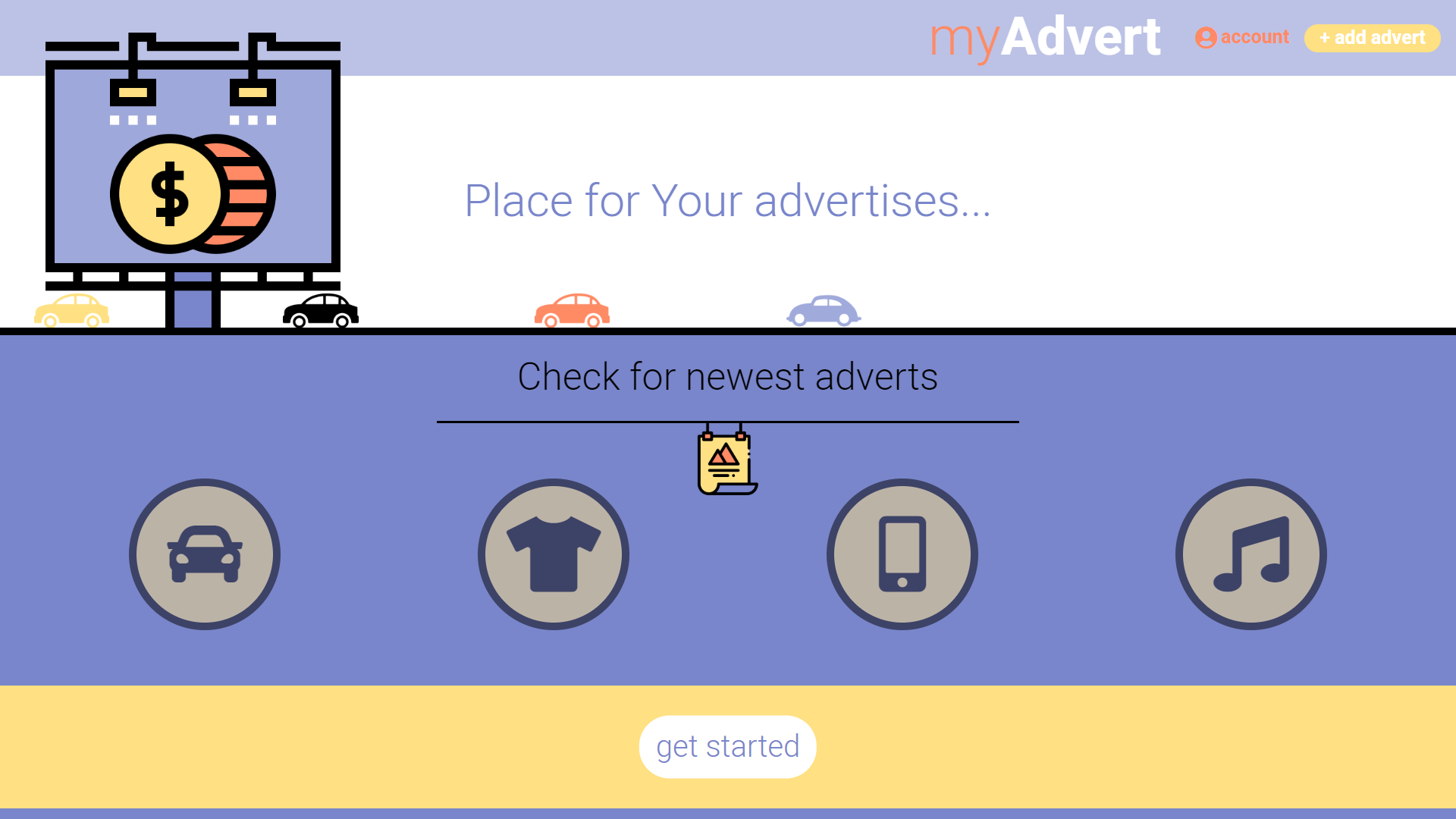
Każdemu elementowi na stronie możemy przypisać tzw. event, który będzie wywoływał konkretną funkcję, gdy użytkownik np. kliknie, tak jak w przykładzie wyżej.

Po kliknięciu w przycisk na samym końcu formularza zostaje wywołana funckja validateOffer, która sprawdza poprawność zawartych przez użytkownika danych.

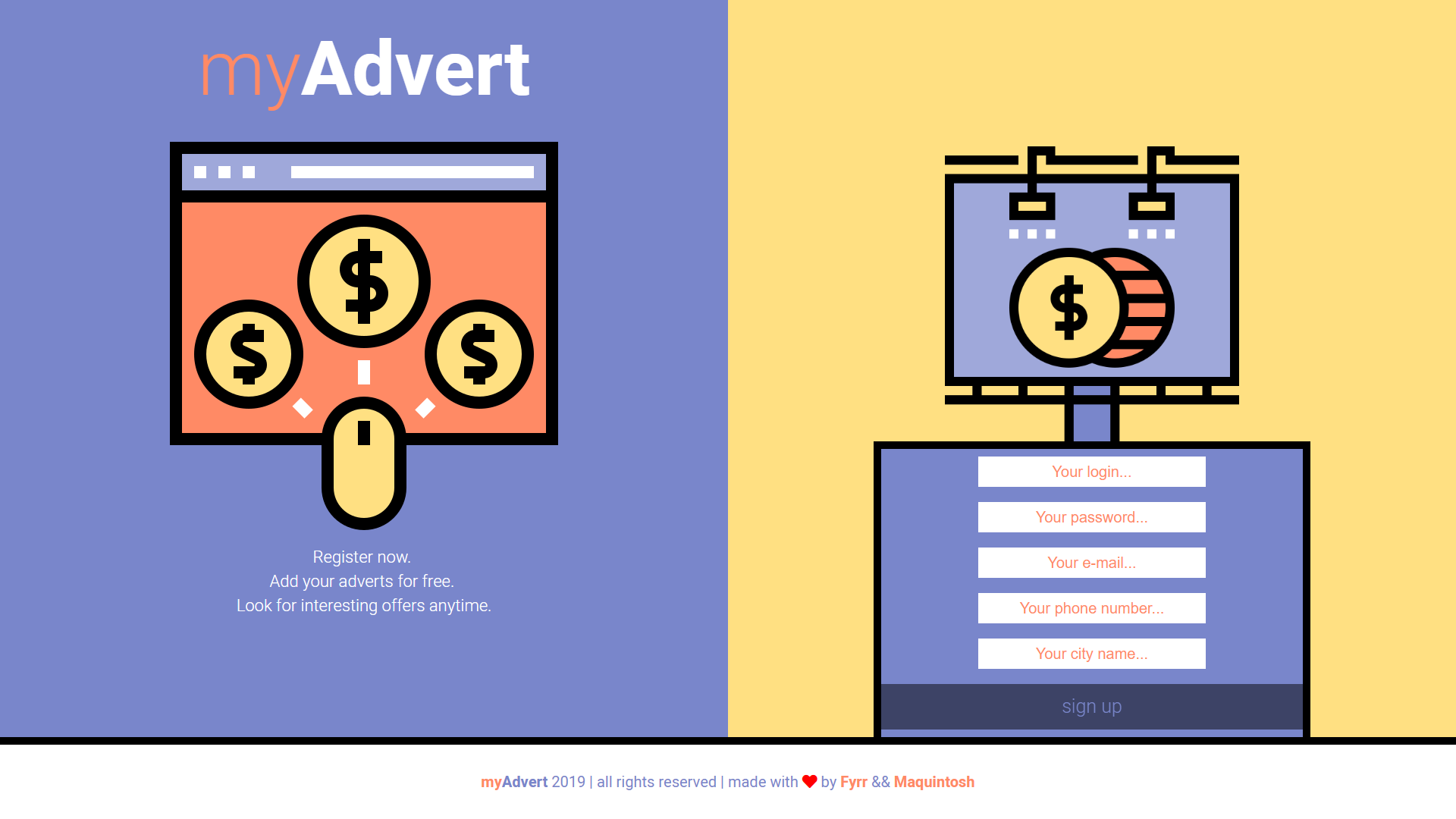


Przykład funkcji, która na podstawie aktualnie wybranej wartości znacznika html <select> zwraca posortowane ogłoszenia według wybranej kategori cenowej lub daty dodania ogłoszenia.

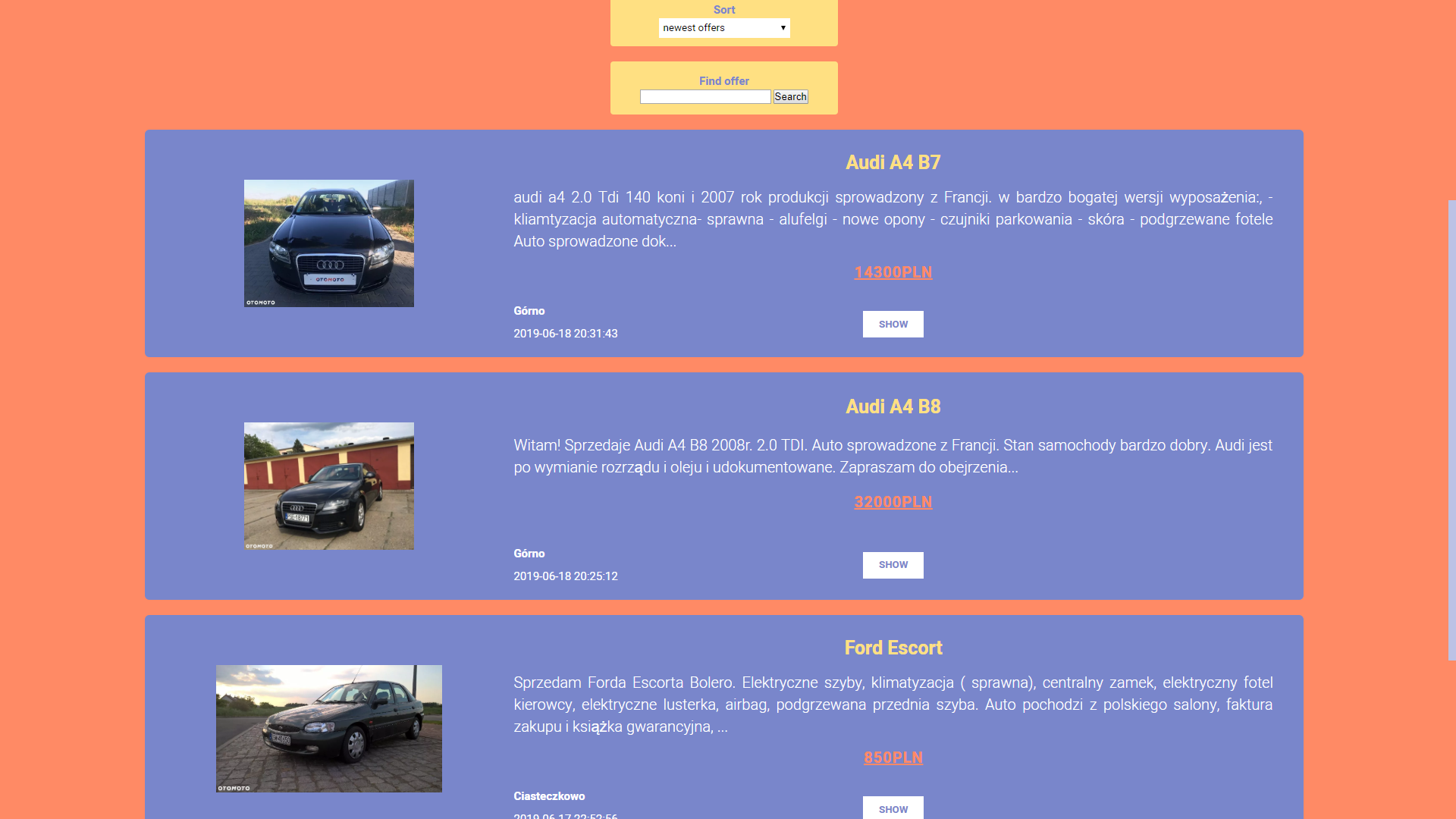
**Strona pod kątem wizulanym**



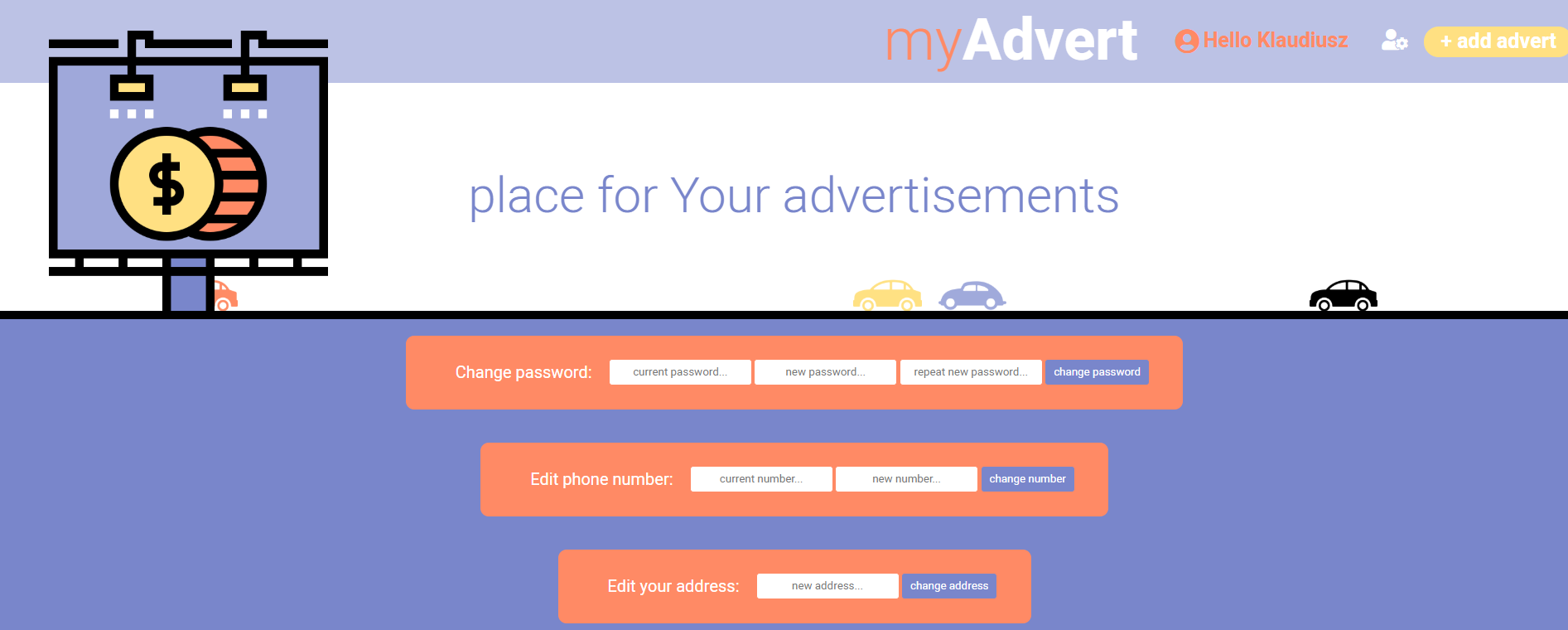
Od góry nawigacja główna wraz z panelem logowania, środek jako menu wyboru kategori ofert, którymi użytkownik jest zainteresowany, od dołu przycisk rejestracji nowego użytkownika.



Podstrona rejestracji nowego użytkownika. Od prawej formularz rejestracji.



Podstrona ogłoszeń z kategorii ‘motoryzacja’.



Panel klienta z edycją danych.

**Moduły PHP**

**Funkcje**

Projekt podzielony jest na dwa typy plików wykonawczych: pliki obsługujące widoki, plik obsługujący połączenie z bazą oraz zwracający elementy widoków odpowiednio przygotowane przez wykorzystanie możliwości języka PHP.

**Baza danych – PostgreSQL**

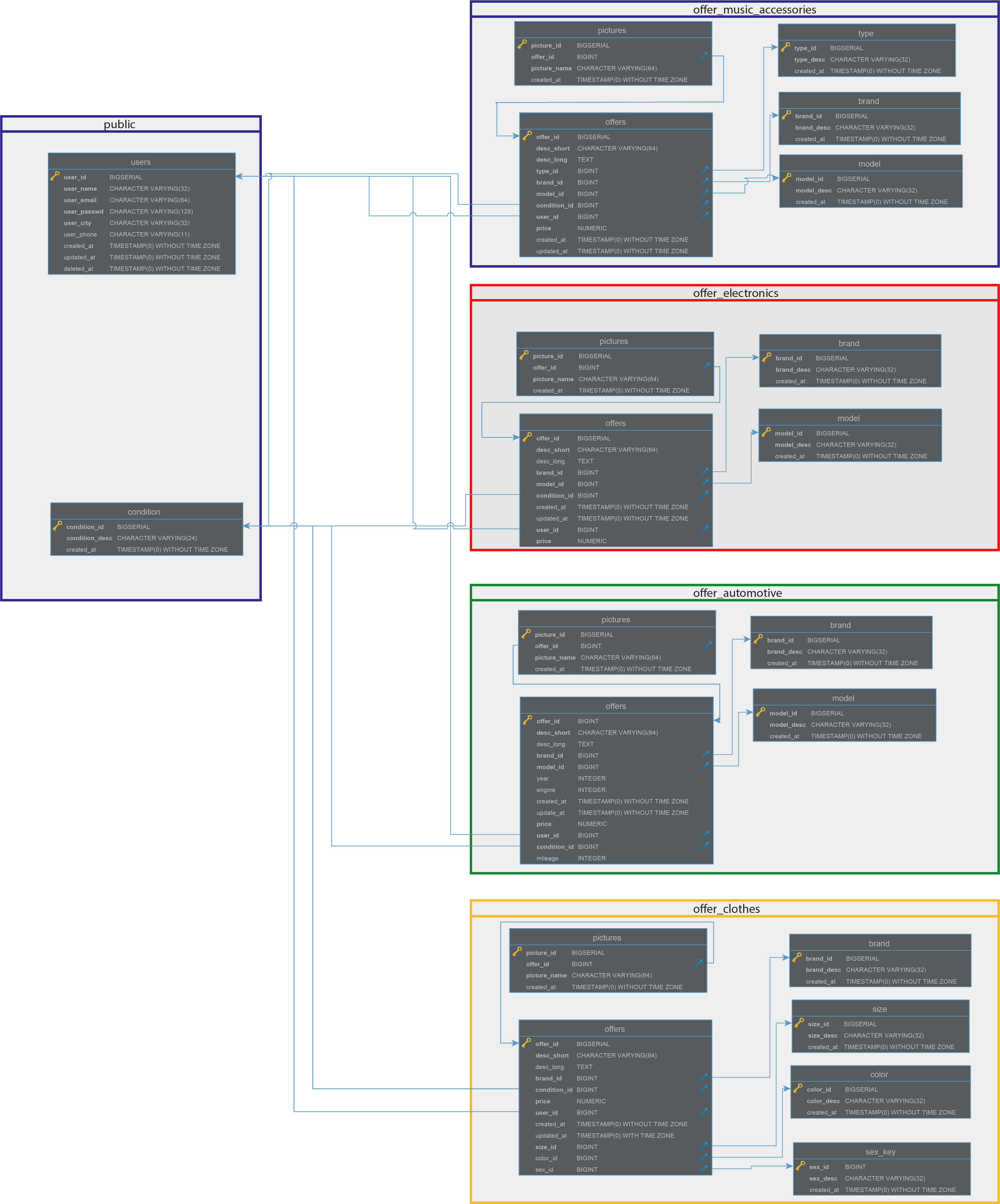
Baza danych PostgreSQL posiada obsługę języka plsql w postaci plpgsql. Wykorzystany connector wewnątrz plików PHP oraz zapytania występują w formie wywołań funkcji z pomocą instrukcji *SELECT*. Jak możemy wyczytać w dokumentacji[[1]](#footnote-1) instrukcja *SELECT* służy jako wywołanie funkcji, z pomocą której możemy dokonywać wyświetlania, dodawania, usuwania lub modyfikacji danych bez potrzeby deklarowania bezpośrednich zapytań według plików wykonawczych.

**Baza danych**

W projekcie została utworzona baza danych *meteopi\_myadvert*, nazwa bazy została wygenerowana przez generator usługodawcy <https://linuxpl.com/> zawierający nazwę hosta oraz dobraną do tego projektu nazwę bazy danych odwołującej się do nazwy projektu „myAdvert”. Projekt z założenia miał obsługiwać różne typy ogłoszeń, w związku z tym każda z kategorii odwołuje się do osobnego schematu. Dane wykorzystywane przez każdą z kategorii przechowywane są w domyślnym – schemacie publicznym.   
Schematy:  
- public – przechowywane są dane użytkowników w tym loginy i hasła oraz tabela condition(stan)  
- offer\_automotive – ogłoszenia, tabele powiązane wyłącznie z ogłoszeniami z kategorii motoryzacja  
- offer\_clothes – ogłoszenia, tabele powiązane z kategorią moda  
- offer\_electronics – ogłoszenia, tabele powiązane z kategorią elektronika  
- offer\_music\_accessories – ogłoszenia, tabele powiązane z kategorią akcesoria muzyczne/muzyka

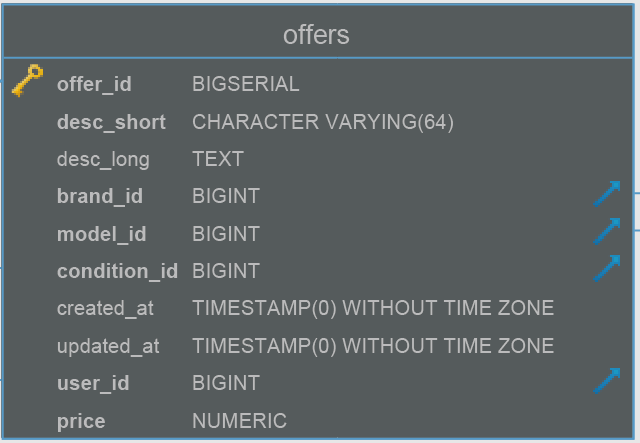
**Schemat ERD**

Poniższy schemat przedstawia powiązania z poszczególnymi tabelami oraz schematami, które zostały zastosowane w celu uporządkowania ogłoszeń. Każde z ogłoszeń posiada odpowiednią listę atrybutów. Przykładowo ogłoszenia z modą nie posiadają pojemności silnika, elektronika nie posiada rozmiaru.



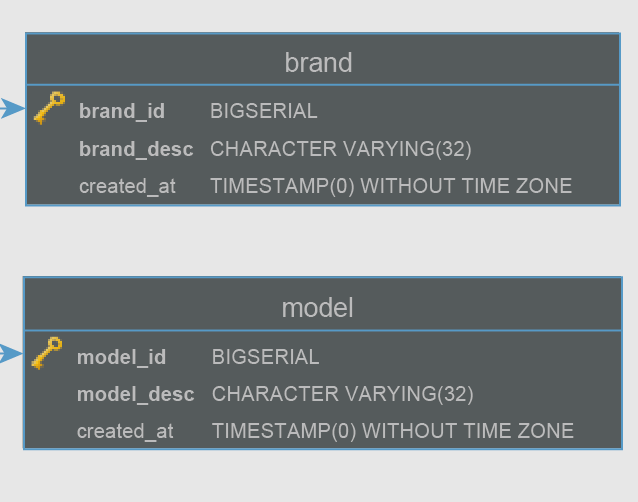
**Tabele transakcyjne**

Tabele transakcyjne posiadają informacje o ofertach oraz użytkownikach. Przykładową tabelą transakcyjną jest przykładowo tabela *offers* ze schematu *offer\_electronics.*



Posiada takie atrybuty jak krótki opis/tytuł, długi – szczegółowy opis oferty, cena oraz data utworzenia I modyfikacji. Klucz główny *offer\_id* jest unikatowy dla schematu *offer\_electronics.* W przypadku odwołania się do *offer\_id* (offer\_id powtarza się również na pozostałych schematach) musimy wiedzieć do której kategorii chcemy się odwołać. Klucze obce *brand\_id, model\_id,* odwołują się tabel słownikowych z ww. schematu.

**Tabele słownikowe**

Tabele słownikowe posiadają podstawowe informacje o markach, modelach, stanu, rozmiarze, kolorze. W zależności od kategorii tabele słownikowe są inne. Schematy posiadają rozdzielone względem siebie tabele przykładowo *brand*, *model*.   
W przypadku wyszukiwania pojazdu na podstawie marki nie musimy określać z jaką kategorią dana marka jest powiązana. Przy dodawaniu ogłoszenia z kategorii motoryzacja w podpowiedziach nie pojawią się marki z innych kategorii.

**Przykładowe polecenie SELECT**

Polecenia SELECT pojawiają się przy wyświetlaniu informacji o ofertach oraz przykładowo przy logowaniu użytkownika do portalu. Wewnątrz projektu nie są wykorzystane bezpośrednie zapytania, lecz funkcje. Poniżej jedna z funkcji, która służy do logowania użytkownika, w przypadku gdy login i hasło się zgadzają zwracane są podstawowe informacje o użytkowniku: user\_id, imię, zakodowane hasło, miasto, numer\_telefonu

|  |
| --- |
| CREATE OR REPLACE FUNCTION login\_user(varchar, varchar) RETURNS login\_result AS $$ DECLARE  lr login\_result;  var\_passwd varchar; BEGIN  var\_passwd = hash\_passwd($2);  **SELECT user\_id, user\_name, user\_passwd, user\_city, user\_phone   INTO lr.tab\_hash, lr.tab\_name, var\_passwd, lr.tab\_city, lr.tab\_phone   FROM users   WHERE user\_email=$1 AND deleted\_at IS NULL AND user\_passwd=var\_passwd;**  RETURN lr; END; $$ LANGUAGE plpgsql; |

Wywołanie:

|  |
| --- |
| SELECT \* FROM login\_user('test@test.pl', 'testowee') |

**Przykładowe polecenie INSERT**

Polecenia INSERT pojawiają się przy umieszczaniu nowych ogłoszeń, dodawaniu nowych użytkowników oraz nowych kluczy, których nie ma w bazie danych. W poniższym przykładzie dodawana jest nowa fotografia do nowego ogłoszenia z kategorii motoryzacja, które zawiera *offer\_id* jako odniesienie do tabeli *offers.* Wewnątrz tej funkcji jest zabezpieczenie na wypadek próby dodawania zdjęć do oferty, która nie została dodana (id=0).

|  |
| --- |
| CREATE OR REPLACE FUNCTION offer\_automotive.add\_picture(integer, varchar) RETURNS boolean AS $$ BEGIN  IF $1<1 THEN  RETURN false;  END IF;  **INSERT INTO offer\_automotive.pictures(offer\_id, picture\_name, created\_at) VALUES ($1, $2, NOW());**  RETURN true; END; $$ LANGUAGE plpgsql; |

Wywołanie:

|  |
| --- |
| SELECT offer\_automotive.add\_picture(12, 'przykładowe\_zdjęcie.jpg'); |

**Przykładowe polecenie UPDATE**

Polecenia UPDATE służą do aktualizowania danych ofert oraz danych użytkownika (telefon, miasto, hasło). Poniższa funkcja służy do aktualizacji danych miasta użytkownika. W przypadku kiedy użytkownik jest na siebie zalogowany(istnieje w bazie) następuje zmiana danych.

|  |
| --- |
| CREATE FUNCTION change\_address(bigint, varchar) RETURNS boolean AS $$  BEGIN  IF EXISTS(SELECT \* FROM users WHERE user\_id=$1) THEN  **UPDATE users SET user\_city=$2, updated\_at=NOW() WHERE user\_id=$1**;  RETURN true;  END IF;  RETURN false;  END;  $$ LANGUAGE plpgsql; |

Wywołanie:

|  |
| --- |
| SELECT change\_address(12, 'Chrząszczyżewoszyce'); |

**Przykładowe polecenie DELETE**

Polecenia DELETE służą do usuwania ofert. Ponieważ w przypadku oszustwa chcielibyśmy uniknąć dodania możliwości użytkownikowi usunięcia konta poprzez usunięcie jego danych z bazy w związku z tym użytkownik otrzymuje przypisanie daty, w której zgłosił chęć usunięcia konta. Przypisanie powoduje, że przy próbie zalogowania użytkownik nie zostaje zalogowany. Przedstawiona poniżej funkcja służy do usuwania ofert z portalu ogłoszeniowego.

|  |
| --- |
| CREATE FUNCTION offer\_electronics.delete\_offer(bigint, bigint) RETURNS boolean AS $$  BEGIN  IF EXISTS(SELECT \* FROM offer\_electronics.offers WHERE offer\_id=$1 AND user\_id=$2) THEN  **DELETE FROM offer\_electronics.offers WHERE offer\_id=$1;**  RETURN true;  END IF;  RETURN false;  END;  $$ LANGUAGE plpgsql; |

Wywołanie:

|  |
| --- |
| SELECT offer\_electronics.delete\_offer(12, 15); |

**Spis funkcji**

**Schemat „public”:**

|  |  |
| --- | --- |
| **Funkcja** | **Typ zwracany** |
| *domain\_name*() | character varying |
| *hash\_passwd*(character varying) | character varying |
| *login\_user*(character varying, character varying) | login\_result\* |
| *register\_user*(character varying, character varying, character varying, character varying, integer) | character varying |
| *change\_address*(bigint, character varying) | boolean |
| *change\_passwd*(bigint, character varying, character varying, character varying) | boolean |
| *change\_phone\_num*(bigint, character varying, character varying) | boolean |
| *show\_user\_offers*(bigint) | TABLE(tab\_delete text, tab\_edit text, tab\_image text, tab\_desc\_short character varying, tab\_desc\_long text, tab\_city character varying, tab\_price double precision, tab\_created\_at timestamp without time zone) |

**\*** tab\_name(varchar(64)), tab\_hash(varchar(128)), tab\_city(varchar(64)), tab\_phone(integer), tab\_id(bigint)

**Schemat „offer\_automotive”:**

|  |  |
| --- | --- |
| **Funkcja** | **Typ zwracany** |
| *add\_picture*(integer, character\_varying) | boolean |
| *create\_offer\_automotive*(character varying, text, character varying, character varying, character varying, double precision, bigint, integer, double precision, integer) | bigint |
| *get\_images*(integer) | TABLE(image character varying) |
| *show\_offers*(character varying) | TABLE (tab\_id bigint, tab\_image varchar, tab\_desc\_short varchar, tab\_desc\_long varchar, tab\_city varchar, tab\_price float8, tab\_created\_at timestamp without time zone) |
| *show\_offer\_info*(bigint) | TABLE(desc\_short character varying, desc\_long text, brand character varying, price numeric, contition character varying, size character varying, sex character varying, color character varying, created\_at timestamp without time zone, user\_name character varying, city character varying, phone\_num character varying) |
| *delete\_offer*(bigint, bigint) | boolean |

**Schemat „offer\_clothes”:**

|  |  |
| --- | --- |
| **Funkcja** | **Typ zwracany** |
| *add\_picture*(integer, character\_varying) | boolean |
| *create\_offer\_automotive*(character varying, text, character varying, character varying, character varying, double precision, bigint, integer, double precision, integer) | bigint |
| *get\_images*(integer) | TABLE(image character varying) |
| *show\_offers*(character varying) | TABLE (tab\_id bigint, tab\_image varchar, tab\_desc\_short varchar, tab\_desc\_long varchar, tab\_city varchar, tab\_price float8, tab\_created\_at timestamp without time zone) |
| *show\_offer\_info*(bigint) | TABLE(desc\_short character varying, desc\_long text, brand character varying, price numeric, contition character varying, size character varying, sex character varying, color character varying, created\_at timestamp without time zone, user\_name character varying, city character varying, phone\_num character varying) |
| *delete\_offer*(bigint, bigint) | boolean |

**Schemat „offer\_electronics”:**

|  |  |
| --- | --- |
| **Funkcja** | **Typ zwracany** |
| *add\_picture*(integer, character\_varying) | boolean |
| *create\_offer\_automotive*(character varying, text, character varying, character varying, character varying, double precision, bigint, integer, double precision, integer) | bigint |
| *get\_images*(integer) | TABLE(image character varying) |
| *show\_offers*(character varying) | TABLE (tab\_id bigint, tab\_image varchar, tab\_desc\_short varchar, tab\_desc\_long varchar, tab\_city varchar, tab\_price float8, tab\_created\_at timestamp without time zone) |
| *show\_offer\_info*(bigint) | TABLE(desc\_short character varying, desc\_long text, brand character varying, price numeric, contition character varying, size character varying, sex character varying, color character varying, created\_at timestamp without time zone, user\_name character varying, city character varying, phone\_num character varying) |
| *delete\_offer*(bigint, bigint) | boolean |

**Schemat „offer\_music\_accessories”:**

|  |  |
| --- | --- |
| **Funkcja** | **Typ zwracany** |
| *add\_picture*(integer, character\_varying) | boolean |
| *create\_offer\_automotive*(character varying, text, character varying, character varying, character varying, double precision, bigint, integer, double precision, integer) | bigint |
| *get\_images*(integer) | TABLE(image character varying) |
| *show\_offers*(character varying) | TABLE (tab\_id bigint, tab\_image varchar, tab\_desc\_short varchar, tab\_desc\_long varchar, tab\_city varchar, tab\_price float8, tab\_created\_at timestamp without time zone) |
| *show\_offer\_info*(bigint) | TABLE(desc\_short character varying, desc\_long text, brand character varying, price numeric, contition character varying, size character varying, sex character varying, color character varying, created\_at timestamp without time zone, user\_name character varying, city character varying, phone\_num character varying) |
| *delete\_offer*(bigint, bigint) | boolean |

**Przechowywanie plików**

Przechowywanie plików odbywa się poprzez zapisywanie fotografii ogłoszeń według poniższej struktury:

**offers\_picture / kategoria / id\_ogłoszenia / nazwa\_pliku**

przykłady zastosowania:

offers\_picture/automotive/29/892079297\_.webp  
offers\_picture/electronics/1/smartfon-huawei-p20-lite-czarny-dual-sim-323882f1.jpg

Uzyskiwanie informacji o zdjęciach pochodzących z pojedynczej oferty odbywa się poprzez wykorzystanie funkcji:

|  |
| --- |
| CREATE FUNCTION offer\_clothes.get\_images(bigint) RETURNS TABLE (image varchar) AS $$  BEGIN  RETURN QUERY (SELECT concat('**offers\_picture**/clothes/', $1, '/', picture\_name)::varchar as image FROM offer\_clothes.pictures WHERE offer\_id=$1);  END;  $$ LANGUAGE plpgsql; |

Wywołanie:

|  |
| --- |
| SELECT \* FROM offer\_automotive.get\_images(29) |

1. 4.3. Calling Functions - <https://www.postgresql.org/docs/9.1/sql-syntax-calling-funcs.html> [↑](#footnote-ref-1)