Projekt bazy danych sklepu internetowego

Bazy danych

Prowadzący: Antoni Ligęza
e-mail: aligeza@pwsz-ns.edu.pl

Spis treści

1
Projektowanie Bazy Danych *Sklepu Internetowego z Akcesoriami Kuchennymi*

Projekt: Przygotowanie serwisu internetowego (sklepu internetowego) zajmującego się sprzedażą akcesorii kuchenych.
Obecnie zamawiający sprzedaje towar na popularnym serwisie aukcyjnym.
W skład projektu wchodzi przygotowanie:

- Bazy danych
- Interfejsu graficznego

Obecnie zamawiający sprzedaje towar na popularnym serwisie aukcyjnym.

1.1. Sformułowanie celu i zadań

1. Minimalizacja czasu
    - pomiędzy złożeniem zamówienia przez klienta a przekazaniem towaru do wysyłki w przypadku wysyłki za pobraniem
    - pomiędzy zaksięgowaniem wpłaty klienta a przekazaniem towaru do wysyłki w przypadku płatności z góry
2. Minimalizacja ilości osób zaangażowanych w zarządzanie sprzedażą
3. Ujednolicenie istniejącej bazy produktów i ich kategorii
4. Uniezależnienie się od wykorzystywanego portalu aukcyjnego
5. Automatyzacja wystawiania faktur/rachunków
6. Zapewnienie łatwej kontroli nad aktualnymi stanami magazynowymi produktów
7. Zapewnienie kontroli dostępu do danych
8. Ograniczenie modyfikacji danych dla poszczególnych pracowników

1.1.1. Analiza istniejacej bazy — analiza stanu wyjsciowego

Zamawiający SZBD sprzedaje towar na popularnym serwisie aukcyjnym. Ze względu na powiększenie oferowanego asortymentu istniejący model funkcjonowania przedsiębiorstwa przestaje być wystarczająco wydajny ze względu na źle przemyślany schemat przechowywania danych

1. Opisy produktów jako pliki tekstowe przechowywane w strukturze katalogowej odpowiadającej kategoriom
2. Stany magazynowe w arkuszu kalkulacyjnym (bez multidostępu do zapisu)
3. Ceny w arkuszu kalkulacyjnym
4. Rozliczenia roczne w arkuszu kalkulacyjnym
5. Dokumenty tj. faktury, raporty z przekazania do wysyłki) w formie papierowej
6. Baza kupujących przechowywana jest przez system aukcyjny

W jaki sposób baza jest wykorzystywana?
Aby dodać produkt do sprzedaży, wyszukiwany jest odpowiedni plik tekstowy z jego opisem oraz sprawdzane cena i stan magazynowy w arkuszu kalkulacyjnym. Kategoria ustalana jest ręcznie w arkuszu kalulacyjnym zawierającym stany magazynowe Produkty są wystawianie ręcznie na popularnym serwisie aukcyjnym. Informacje o kupnie: jaki towar, w jakiej ilości, w jakiej cenie oraz o wpłacie dostarczane są przez serwis aukcyjny w formie tekstowej (elektronicznie) Do osoby odpowiedzalnej za gromadzenie towaru w paczkę do wysyłki i przekazanie paczki kurierowi przekazywana jest w formie wydruku Informacje potrzebne do generowania raportów przepisywane są do arkusza kalkulacyjnego ręcznie.
Problemy:

1. Dane wyszukiwane są ręcznie.
2. Duże ryzyko błędu ludzkiego podczas ręcznego przenoszenia danych.

1.1.2. Lista tabel w projektowanej bazie

1. **Kategoria:**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
# kategoria_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
nazwa	varchar(30)	-	nie	nieznana	-
FK id_nadrzędnej	int(10)	unsigned	tak	null	-

- # kategoria_id - pole jednoznacznie identyfikujące kategorię
- nazwa - nazwa kategorii może się powtarzać (np. podkategoria ”pozostałe” w różnych kategoriach)
- FK id_nadrzednej - id kategorii nadrzędnej, kategoriami poziomu najwyższego są te, gdzie id_ nadrzędnej ma watrość NULL

2. **Klient:**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
klient_id	int(10)	unsigned	nie	brak	A_INC
typ	unum('p' 'f')	-	nie	p	-
nazwisko	varchar(50)	-	tak	null	-
imie	varchar(50)	-	tak	null	-
NIP	bigint(20)	-	tak	null	-
nazwa_firmy	varchar(100)	-	tak	null	-
dom_adr_wys	varchar(200)	-	tak	null	-
login	varchar(50)	-	nie	brak	-
haslo	varchar(32)	-	nie	brak	-

- # klient_id, - pole jednoznacznie identyfikujące klienta
- typ - 'p'- klient indywidualny 'f' -firma
- nazwisko -
- imię
- NIP
- nazwa_firmy
- domyślny_adres_wysyłki - wartość wstawiana domyślnie do pola 'adres_wysylki'
- login
- hasło

3. **Pracownik**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
pracownik_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
login	varchar(50)	-	nie	brak	-
haslo	varchar(32)	-	nie	brak	-
uprawnienia	varchar(100)	-	nie	brak	-

- # pracownik_id - pole jednoznacznie identyfikujące pracownika
- login
- haslo
- uprawnienia

4. **Produkt**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
produkt_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
nazwa	varchar(50)	-	nie	brak	-
kategoria_id	int(10)	-	tak	null	-
opis	text	-	nie	brak	-
stan_magazyn	int(10)	unsigned	nie	0	
cena	decimal(10,2)	unsigned	nie	brak	-
blokada	tinyint(1)	-	nie	0	-

- # produkt_id pole jednoznacznie identyfikujące produkt
- nazwa - nazawa produktu wyświetlana klientowi na stronie internetowej, pakującemu produkt w raportach itd
- FK kategoria_id id kategorii, do której jest przydzielany produkt, jeśli ma wartość NULL, produkt nie będzie nigdzie wyświetlany
- opis - opis produktu (HTML)
- stan_magazyn - aktualny stan magazynowy wartość wprowadzana ręcnie i zmniejszana w momencie zatwierdzenia transakcji do realizacji
- cena
- blokada - wyświetlanie i sprzedaż produktu może być wstrzymana

5. **produkt_edycja**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
produkt_edycja_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
produkt_id	int(10)	unsigned	nie	brak	-
kolumna	varchar(15)	-	nie	brak	-
poprz_wartosc	text	-	nie	brak	-
czas_edycji	timestamp	-	nie	CURRENT _TIMESTAMP	-
pracownik_id	int(10)	unsigned	nie	brak	-

- # produkt_edycja_id - pole jednoznacznie identyfikujące zmianę (edycję atrybutu lub dodanie) produktu, a więc konkretną wersję konkretnego produktu
- FK produkt_id - id edytowanego produktu
- kolumna - nazwa edytowanego atrybutu lub informacja "dodano produkt"
- poprz_wartosc - wartość atrybutu sprzed edycji lub wartość NULL gdy dodano nowy produkt
- czas_edycji - czas wprowadzenia zmian
- FK pracownik_id - id pracownika edytującego lub dodającego produkt

6. **Transakcja**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
transakcja_id	int(10)	unsigned	nie	brak	AUTO_INCREMENT
klient_id	int(10)	unsigned	tak	null	-
adres_wysylki	varchar(200)	-	nie	brak	-
kwota_wplacona	decimal(12,2)	-	nie	0.00	-
status	enum ('u','a','p', 'z','w','d')	-	nie	u	-

- # transakcja_id pole jednoznacznie identyfikujące transakcję

- FK klient_id - id zalogowanego klienta który utworzył transakcję lub NULL dla kupującego bez logowania
- adres_wysylki - adres podany w formularzu dostawy transakcji
- kwota_wplacona
- status - utworzono/potwierdzono(po wyświeltleniu podsumowania transakcji)/zapłacono/wysłano/dostarczono

7. **Transakcja_produkt_v**

Nazwa	**Typ**	**Atrybuty**	**Null**	**Ustawienia domyslne**	**Dodatkowo**
transakcja_id	int(10)	unsigned	nie	brak	-
sztuk	tinyint(3)	unsigned	nie	brak	-
produkt_edycja_id	int(10)	unsigned	nie	brak	-

- FK transakcja_id
- sztuk
- FK produkt_edycja_id

1.1.3. Związki między elementami:

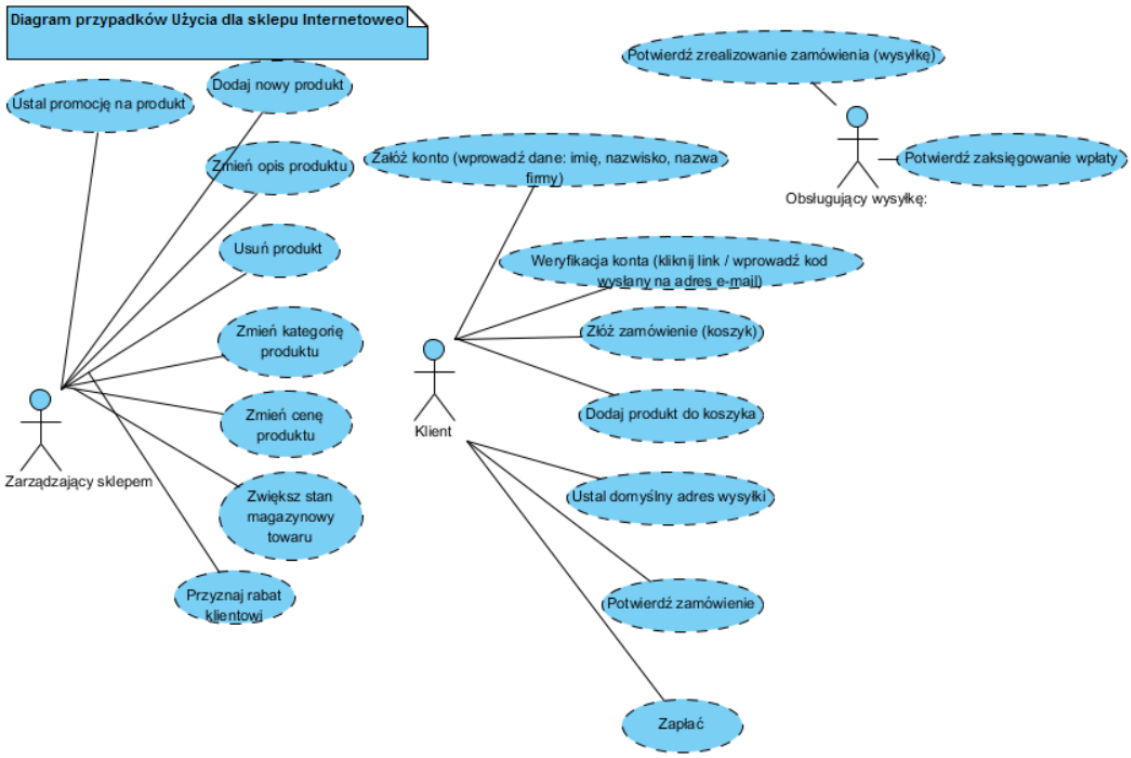
- Kategorie są wielopoziomowe (z podkategoriami)
- Produkt należy tylko do jednej (pod)kategorii
- Dane produktów mogą zmieniać się w każdej chwili
- Dla każdej transakcji należy przetrzymywać informacje o produkcie z momentu wyświetlenia podsumowania zamówienia klientowi
- Raport podsumowujący transakcję przed "zatwierdzam i płacę" = świętość, dane tam nie mogą ulec zmianie przez modyfikację parametrów produktu (w szczególności: cena) Bezwzględnie musi istnieć możliwość wyświeltlenia stanu produktu:
  - nazwa
  - opis
  - cena

  z momentu w którym klient dokonał zakupu
- Użytkownik niezalogowany ma możliwość zrobienia zakupów w sklepie internetowym

1.1.4. Diagram przypadków użycia

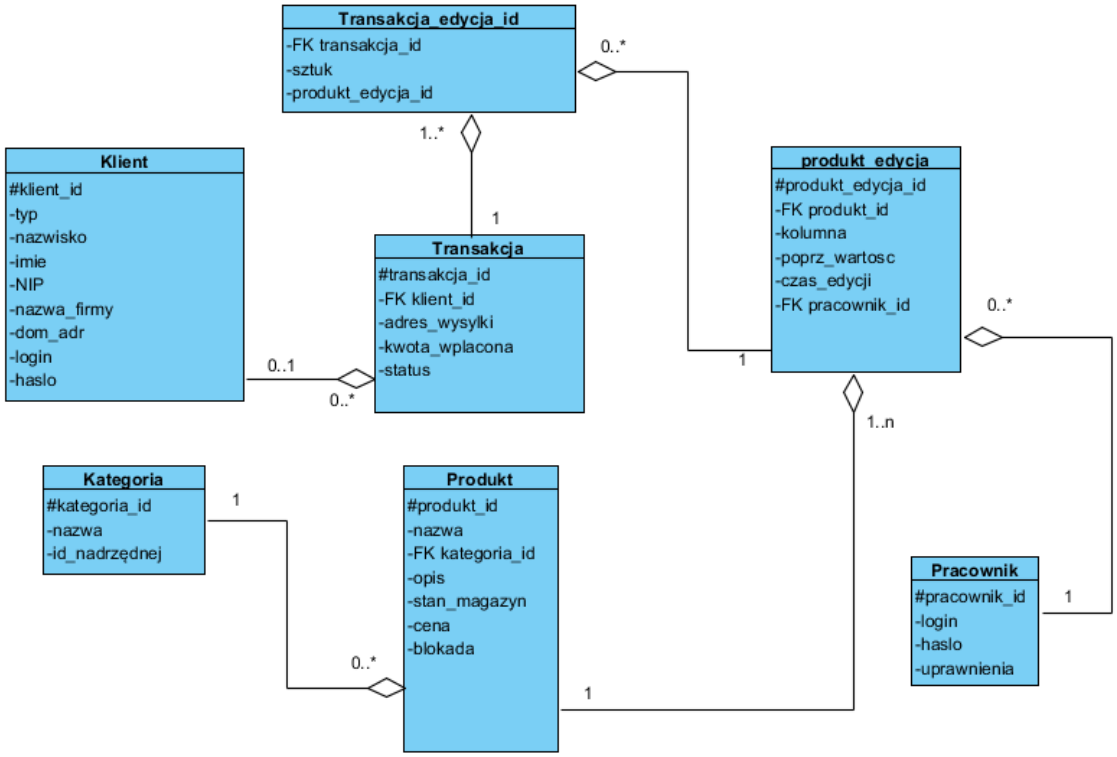
Diagram przypadków użycia (ang. use case) – graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi.

Diagram przypadków użycia tworzony jest w początkowych fazach modelowania. Diagram ten stanowi tylko przegląd możliwych działań w systemie, szczegóły ich przebiegu są modelowane za pomocą innych technik. Diagram przypadków użycia przedstawia usługi, które system świadczy aktorom, lecz bez wskazywania konkretnych rozwiązań technicznych.



Rys. 1.1. Diagram przypadków uycia

1.1.5. Diagram klas



Rys. 1.2. Diagram klas

1.1.6. Wyjściowa funkcjonalność

Sklep/seris internetowy z możliwością założenia konta użytkownika.

Rodzaje kont w serwisie:

Co najmniej jedno konto z uprawnieniami administratora z zastrzeżeniem że nie może usunąć własnego konta (zabezpieczenia aby w serwisie istniało konto administatora)

Konta pracowników którym można nadać odpowiednie uprawnienia dodawanie/usuwanie kategorii/produktów generowanie raportów sprzedaży. Konto pracownika może posiadać wszystkie uprawnienia lub tylko wybrane. Konto pracownika może być utworzone/skasowane tylko przez innego pracownika z uprawnieniami do zarządzania kontami pracowników.

Konta użytkowników. Standardowe konta użytkowników. Użytkownik ma prawo edytowania/zmiany swoich danych osobowych. Do rozróżniania użytowników wykorzystywane są ich unikalne loginy podawane przy rejestracji w serwisie. Zasady tworzenia loginów:

1. Login musi być unikalny w bazie
2. Nie jest rozróżniana wielkość znaków w loginie np. ADMIN, admin, AdMin - traktowany jest jak jeden login

Loginy są zapisywane w bazie, raz wykorzystany login nie może być użyty ponownie.

Użytkownik ma możliwość usunięcia konta z serwisu. w takim przypadku muszą zostać usunięte wszystkie jego dane osobowe. Zachowane zostają dane o przeprowadzonych z tym klientem transakcjach bez możliwości jego identyfikacji.

1.1.7. Opis projektu w całkowitym ujęciu

Projekt polega na przygotowaniu interfejsu dla

1. Administratora całego serwisu
2. Pracownika z przyznanymi mu uprawnieniami
3. Klienta

Administrator ma mieć możliwość:

1. zarządzania kategoriami tzn.: dodawanie/usuwanie/edycja istniejących kategorii
2. dodawania/usuwania/edycji parametrów produktów

Dla użytkownika serwisu przewidziano nastepujące możliwości

1. Wyświetlanie listy kategorii produktów
2. Zarejestrowanie konta w serwisie
3. Dostep do ustawień konta w których znajduje się:
   - możliwośc edycji podstawowych danych
   - możliwość zmiany hasła do konta
4. Koszyk
   - dodawanie produktów do koszyka z możliwością wyboru ilości sztuk
   - potwierdzenie transakcji.

Rozliczenie z klientem będzie odbywać się za pomocą zewnętrzego systemu płatniczego.
Po stronie sklepu pozostaje wskazanie

1. Numeru konta do przelewu
2. Tytułu wpłaty

1.2. Definiowanie tabel, wiezów integralnosci, perspektyw

1.2.1. Procedury wykorzystane w projekcie

1. menu_drzewo_kat Procedura dla danego kategoria_id zwraca kategorie bezpośrednio podrzędne dla wszystkich kategorii leżących na ścieżce od zdefinionanej argumentem do korzenia. Zapewnia to wyświetlenie w menu sklepu minimalnej listy kategorii niezbędnej do swobodnej nawigacji po drzewie kategorii.

```
CREATE DEFINER=`root`@`localhost`
PROCEDURE `menu_drzewo_kat`(IN `pkat_id` INT)
READS SQL DATA
BEGIN
DECLARE kat_id INT;
IF pkat_id IS NOT NULL THEN
SET kat_id = pkat_id;
WHILE kat_id IS NOT NULL DO
SELECT * FROM kategoria WHERE id_nadrzednej=kat_id;
SELECT id_nadrzednej FROM kategoria
WHERE kategoria_id=kat_id INTO kat_id;
END WHILE;
END IF;
SELECT * FROM kategoria WHERE id_nadrzednej IS NULL;
END
```

2. produkt_by_kategoria Procedura zwraca dane wszystkich produktów należących do kategorii zdefiniowanej jako argument lub do jej podkategorii, dla dowolnego poziomu zagłębienia drzewa.

```
BEGIN
DECLARE n INT;
SET n=1;
CREATE TEMPORARY TABLE podrzedne
(`kategoria\_id` int(10) unsigned NOT NULL,
`przetworzona` tinyint(1) DEFAULT 0);
CREATE TEMPORARY TABLE podrzedne2
(`kategoria\_id` int(10) unsigned NOT NULL,
`przetworzona` tinyint(1) DEFAULT 0);

INSERT INTO podrzedne VALUES(id\_kat,0);

WHILE n>0 DO
INSERT INTO podrzedne2 SELECT kategoria\_id, 0 FROM kategoria
WHERE id\_nadrzednej IN
(SELECT kategoria\_id FROM podrzedne WHERE przetworzona=0);
UPDATE podrzedne SET przetworzona=1;
INSERT INTO podrzedne SELECT * FROM podrzedne2;
TRUNCATE TABLE podrzedne2;
SELECT COUNT(*) FROM podrzedne WHERE przetworzona=0 INTO n;
END WHILE;

SELECT p.produkt\_id, p.nazwa, p.kategoria\_id, k.nazwa AS nazwa\_kat
FROM produkt p JOIN kategoria k USING(kategoria\_id)
WHERE p.blokada=0 AND k.kategoria\_id IN
(SELECT kategoria\_id FROM podrzedne);

DROP TEMPORARY TABLE podrzedne;
DROP TEMPORARY TABLE podrzedne2;
END
```

3. produkt_hist

```
BEGIN
DECLARE vprodukt\_id INT;
DECLARE poprz\_nazwa VARCHAR(50);
DECLARE poprz\_cena DECIMAL(10,2);
DECLARE poprz\_opis TEXT;
DECLARE poprz\_blokada TINYINT(1);
SELECT produkt\_id INTO vprodukt\_id FROM produkt\_edycja
WHERE produkt\_edycja\_id=pprodukt\_edycja\_id;
SELECT poprz\_wartosc INTO poprz\_nazwa FROM produkt\_edycja
WHERE produkt\_edycja\_id>pprodukt\_edycja\_id
AND produkt\_id=vprodukt\_id AND kolumna='nazwa'
```

```
ORDER BY produkt\_edycja\_id ASC LIMIT 1;
SELECT poprz\_wartosc INTO poprz\_cena FROM produkt\_edycja
WHERE produkt\_edycja\_id>pprodukt\_edycja\_id
AND produkt\_id=vprodukt\_id AND kolumna='cena'
ORDER BY produkt\_edycja\_id ASC LIMIT 1;
SELECT poprz\_wartosc INTO poprz\_opis FROM produkt\_edycja
WHERE produkt\_edycja\_id>pprodukt\_edycja\_id
AND produkt\_id=vprodukt\_id AND kolumna='opis'
ORDER BY produkt\_edycja\_id ASC LIMIT 1;
SELECT poprz\_wartosc INTO poprz\_blokada FROM produkt\_edycja
WHERE produkt\_edycja\_id>pprodukt\_edycja\_id
AND produkt\_id=vprodukt\_id AND kolumna='blokada'
ORDER BY produkt\_edycja\_id ASC LIMIT 1;

SELECT
COALESCE(poprz\_nazwa,nazwa) AS nazwa,
COALESCE(poprz\_cena,cena) AS cena,
COALESCE(poprz\_opis,opis) AS opis,
COALESCE(poprz\_blokada,blokada) AS blokada
FROM produkt WHERE produkt\_id=vprodukt\_id;
END
```

1.2.2. Funkcje wykorzystane w projekcie

Funkcje wykorzystane w projektowanej bazie danych

1. ostatnia_produkt_edycja_id

```
        RETURN (SELECT MAX(`produkt\_edycja\_id`)
        FROM `produkt\_edycja` WHERE `produkt\_id`=produktid)
```

1.2.3. Wyzwalacze wykorzystane w projekcie

Wyzwalacze wykorzystane w bazie danych służące zapisywaniu istotnych zmian dokonywanych w tabeli produkt.

1. produkt_insert_hist

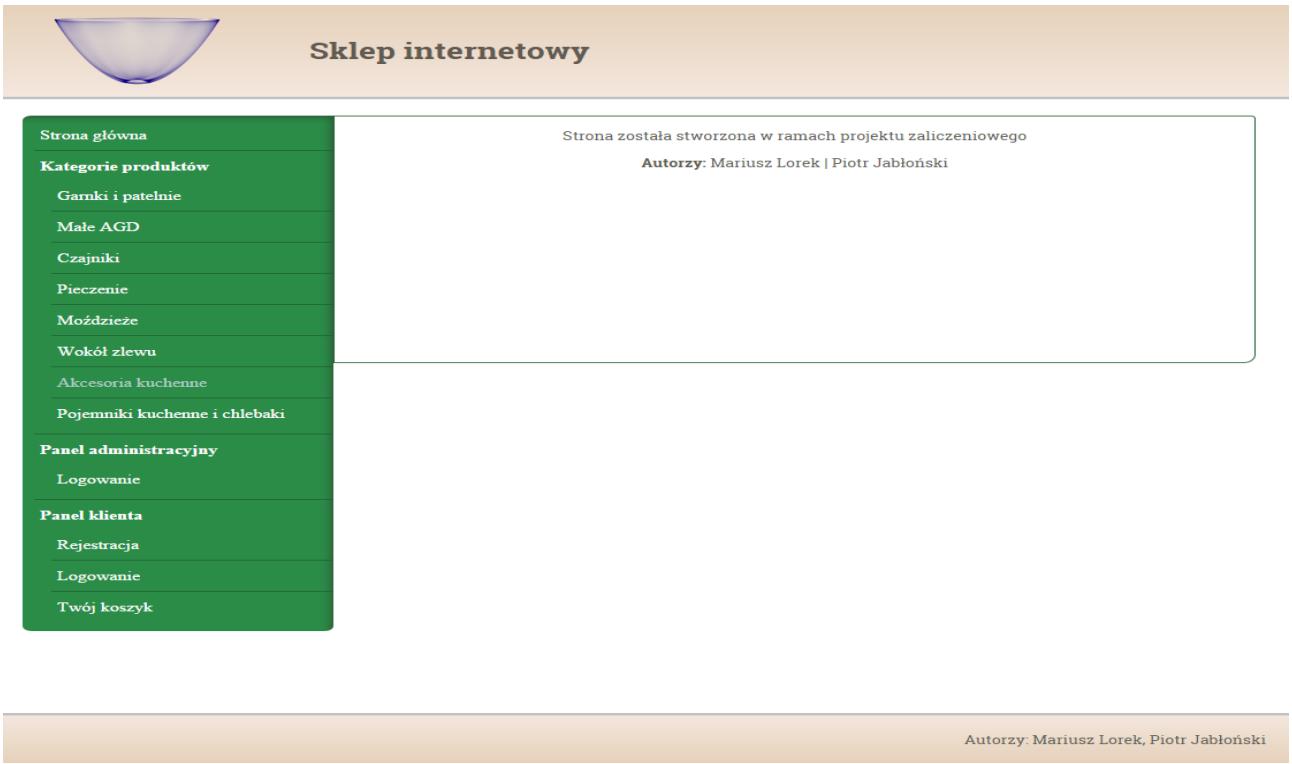
```
CREATE TRIGGER `produkt_insert_hist` AFTER INSERT ON `produkt`
FOR EACH ROW INSERT INTO produkt_edycja (produkt_id, kolumna, pracownik_id)
VALUES (NEW.produkt_id, 'dodanie', @pracownikid)
```

2. produkt_update_hist

```
CREATE TRIGGER `produkt_update_hist` BEFORE UPDATE ON `produkt`
FOR EACH ROW BEGIN
IF NEW.nazwa!=OLD.nazwa THEN
INSERT INTO produkt_edycja
(produkt_id, kolumna, poprz_wartosc, pracownik_id)
VALUES (NEW.produkt_id, 'nazwa', OLD.nazwa, @pracownikid);
END IF;
IF NEW.opis!=OLD.opis THEN
INSERT INTO produkt_edycja
(produkt_id, kolumna, poprz_wartosc, pracownik_id)
VALUES (NEW.produkt_id, 'opis', OLD.opis, @pracownikid);
END IF;
IF NEW.cena!=OLD.cena THEN
INSERT INTO produkt_edycja
(produkt_id, kolumna, poprz_wartosc, pracownik_id)
VALUES (NEW.produkt_id, 'cena', OLD.cena, @pracownikid);
END IF;
```

```
IF NEW.blokada!=OLD.blokada THEN
INSERT INTO produkt_edycja
(produkt_id, kolumna, poprz_wartosc, pracownik_id)
VALUES (NEW.produkt_id, 'blokada', OLD.blokada, @pracownikid);
END IF;
END
```

2
Interfejs graficzny - strona internetowa



Rys. 2.1. Interfejs - strona główna

Kod pliku wejściowego strony index.php, na który przekierowywane są wszystkie zapytania z serwisu poprzez moduł mod_rewrite (ustawienia w pliku .htaccess) możemy zobaczyć poniżej

```
<?php
session_start();
```

```
include('class/database.php');
$DB = database::instance();
include('class/upr.php');
include('ctrl/functions.php');
include('class/klient.php');
include('class/koszyk.php');
include('class/transakcja.php');
sprawdz_uprawnienia();

if(!isset($_GET['site'])) $_GET['site']='';
include('ctrl/drzewo-kategorii.php');
ob_start();
switch($_GET['site']) {
case 'logout':
        session_destroy();
        header("Location: {$dir}glowna.html");
        break;
case 'index':
case '':
        header("Location: {$dir}glowna.html");
        break;
case 'glowna':
        include('view/glowna.php');
        break;
case 'produkt':
        include('ctrl/produkt.php');
        $htitle = 'Lista produktów';
        include('view/produkt.php');
        break;
case 'koszyk':
        include('ctrl/koszyk.php');
        $htitle = 'Twój koszyk';
        include('view/koszyk.php');
        break;
case 'login':
        if(isset($_POST['login']) AND isset($_POST['password']))
                include('ctrl/login.php');
        $htitle = 'Klient - logowanie';
        include('view/logowanie.php');
        break;
case 'login-pracownik':
        ...
case 'kategoria':
        ...
case 'produkt-edycja':
        ...
case 'produkt-edycja-lista':
        ...
case 'rejestracja':
        ...
case 'profil':
        ...
case 'baza_klientow':
        ...
case 'transakcja':
        ...
default:
        header("Location: {$dir}glowna.html");
        exit;
}
$content = ob_get_contents();
ob_end_clean();
include('view/view.php');
/**/
?>
```

Wykorzystywane klasy:

- Baza danych w modelu singletona (model, gdzie może istnieć tylko jedna instancja obiektu danej klasy)

```
class database extends mysqli{
static $THIS; //instancja singletona

private $host   = 'localhost';
private $login  = 'interfejs';
private $pass   = '******';
private $db     = 'projekt_bazy_sklep';

private function __construct(){
        parent::__construct($this->host, $this->login,
                $this->pass, $this->db);
}

static function instance(){
        if(empty(self::$THIS) OR !self::$THIS->ping()) {
                self::$THIS = new database();
                self::$THIS->query('SET NAMES \'utf8\'');
        }
        return self::$THIS;
}
}
```

- Klasa klient

```
class klient {
        static $zalogowany;
        static $login;
        static $klient_id;
        static $typ;
        static $imie;
        static $nazwisko;
        static $NIP;
        static $nazwa_firmy;
        static $dom_adr_wys;

        static function wyciagnij_dane() {
        ...
```

- Koszyk

```
class koszyk {
        static function &produkty() { ... }
        static function dodaj($produkt_id, $sztuk) { ... }
        static function usun($produkt_id) { ... }
}
```

- Uprawnienia

```
class upr {
        const pracownik = 'pracownik';
        const kategoria = 'kategoria';
        const produkt   = 'produkt';
        const wysylka = 'wysylka';
        const klient = 'klient';
        const raport = 'raport';

        static function maUprawnienia($upr) {
                ...
        }
```

```
        static function wyrzucBezUprawnien($upr) {
                if(self::maUprawnienia($upr)) return;
                putMessage('Nie masz uprawnień
                        do przeglądania tej zawartości');
                header("Location: {$dir}glowna.html");
                exit;
        }
        static function wyrzucStad() {
                putMessage('Nie masz uprawnień do
                        przeglądania tej zawartości');
                header("Location: {$dir}glowna.html");
                exit;
        }
}
```

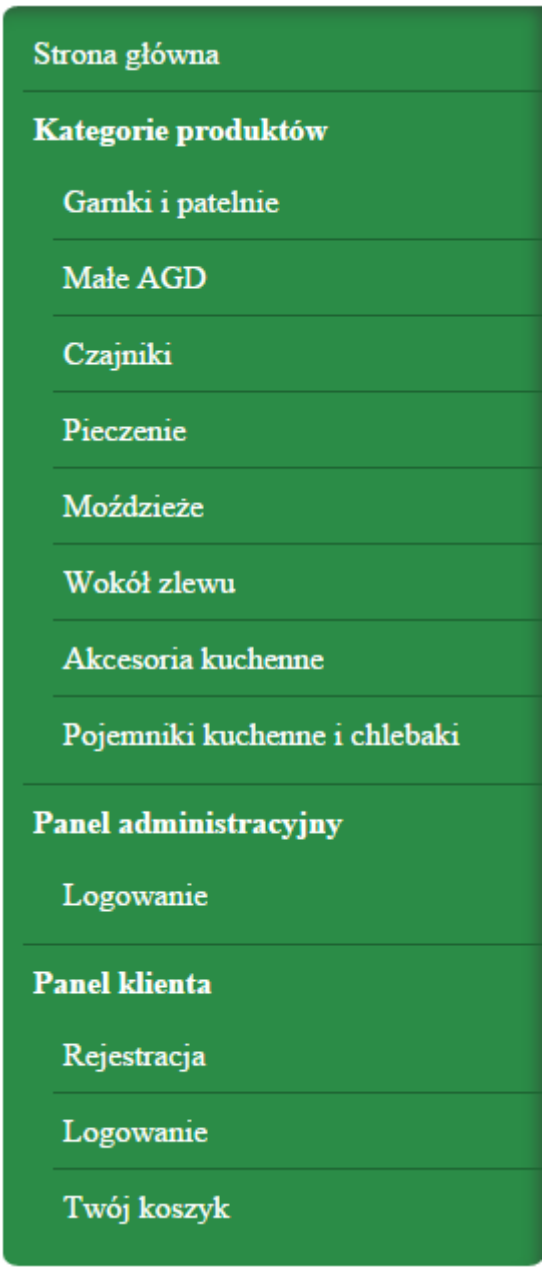
Każdą z podstron obsługują dołączane z poziomu index.php:

- Kontroler - plik z folderu ctrl, zawierający zapytania do bazy danych i warunkowe wykonanie instrukcji takich jak modyfikacja czy wprowadzanie danych. Przykładowy fragment takiego pliku:

```
if(isset($gkat) && is_numeric($gkat)) {
        $DB->real_query("call menu_drzewo_kat($gkat)")
                OR die($DB->error);
        $resultSets = array();
        $i = 0;
        do {
                if ($res = $DB->store_result()) {
                        $resultSets[$i++] =
                                $res->fetch_all(MYSQLI_ASSOC);
                        $res->free();
                } else if($DB->errno)
                putMessage($DB->error);
        } while ($DB->more_results() && $DB->next_result());
        for($l=count($resultSets)-1; $l>=0; $l--) {
                foreach($resultSets[$l] as $row) putKat($row);
        }
}
```

- Widok - plik z folderu view zawierający głównie elementy HTML, tj częsci definiującej wygląd, przykład zawartości:

```
<h2>Klient - logowanie</h2>
<p>Aby zalogować się na konto, wprowadź login i hasło</p>
<form class="formularz" action="login.html" method="post" >
<ul>
        <li>
        <label for="login">Login</label>
        <input type="text" name="login" value="<?php
          if(isset($_POST['login'])) echo $_POST['login']; ?>" >
        </li>
        <li>
        <label for="password">Hasło</label>
        <input type="password" name="password" value="<?php
          if(isset($_POST['password'])) echo $_POST['password']; ?>" >
        </li>
        <li>
        <input type="submit">
        </li>
</ul>
</form>
```



Rys. 2.2. Widok menu dla niezalogowanego użytkownika

Rys. 2.3. Rozwijanie menu

Bibliografia

[1] Balcerzak J., Pansiuk J.: *Wprowadzenie do kartografii matematycznej*, Warszawa, OWPW 2005.

Spis rysunków