

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ ELEKTRONIKI

---

# PROJEKT Z BAZ DANYCH

Internetowy sklep sportowy oparty o  
relacyjną bazę danych

Autorzy:

Tomasz Bartos - 209248

Jakub Dymon - 200335

Wiktor Gerstenstein - 209138

Prowadzący zajęcia:

Dr inż. Robert Wójcik, W4/K-9

OCENA PRACY:

---

Wrocław 2016

# Spis treści

Spis rysunków	I
Spis tabel	II
<b>1 Wstęp</b>	<b>1</b>
1.1 Cel projektu . . . . .	1
1.2 Zakres projektu . . . . .	1
<b>2 Analiza wymagań</b>	<b>2</b>
2.1 Opis działania i schemat logiczny systemu . . . . .	2
2.2 Wymagania funkcjonalne . . . . .	2
2.2.1 Diagram przypadków użycia . . . . .	2
2.2.2 Scenariusze wybranych przypadków użycia . . . . .	5
2.3 Wymagania niefunkcjonalne . . . . .	5
2.3.1 Wykorzystywane technologie i narzędzia . . . . .	5
2.3.2 Wymagania dotyczące rozmiaru bazy danych . . . . .	6
2.3.3 Wymagania dotyczące bezpieczeństwa systemu . . . . .	6
2.4 Przyjęte założenia projektowe . . . . .	6
<b>3 Projekt systemu</b>	<b>7</b>
3.1 Projekt bazy danych . . . . .	7
3.1.1 Analiza rzeczywistości i uproszczony model konceptualny	7
3.1.2 Model logiczny i normalizacja . . . . .	8
3.1.3 Model fizyczny i ograniczenia integralności danych . . . .	9
3.1.4 Inne elementy schematu – mechanizmy przetwarzania	
danych . . . . .	11
3.1.5 Projekt mechanizmów bezpieczeństwa na poziomie bazy	
danych . . . . .	11
3.2 Projekt aplikacji użytkownika . . . . .	11
3.2.1 Architektura aplikacji i diagramy projektowe . . . . .	11
3.2.2 Interfejs graficzny i struktura menu . . . . .	12
3.2.3 Projekt wybranych funkcji systemu . . . . .	12
3.2.4 Metoda podłączania do bazy danych – integracja z	
bazą danych . . . . .	12
3.2.5 Projekt zabezpieczeń na poziomie aplikacji . . . . .	12
<b>4 Implementacja systemu</b>	<b>13</b>
4.1 Realizacja bazy danych . . . . .	13
4.1.1 Tworzenie tabel i definiowanie ograniczeń . . . . .	13
4.1.2 Implementacja mechanizmów przetwarzania danych . .	14

4.1.3	Implementacja uprawnień i innych zabezpieczeń . . . .	16
4.2	Realizacja elementów aplikacji . . . . .	18
4.2.1	Walidacja i filtracja . . . . .	18
4.2.2	Implementacja interfejsu dostępu do bazy danych . . .	18
4.2.3	Implementacja wybranych funkcjonalności systemu . .	20
4.2.4	Implementacja mechanizmów bezpieczeństwa . . . . .	24
<b>5</b>	<b>Testowanie systemu</b>	<b>25</b>
5.1	Instalacja i konfigurowanie systemu . . . . .	25
5.2	Testowanie opracowanych funkcji systemu . . . . .	27
5.2.1	Testowanie zamówienia . . . . .	27
5.2.2	Testowanie filtrowania . . . . .	28
5.3	Testowanie mechanizmów bezpieczeństwa . . . . .	30
5.4	Wnioski z testów . . . . .	32
<b>6</b>	<b>Podsumowanie</b>	<b>33</b>
	<b>Literatura</b>	<b>35</b>

## Spis rysunków

Rysunek 1.	Schemat logiczny systemu . . . . .	2
Rysunek 2.	Diagram przypadków użycia . . . . .	4
Rysunek 3.	Model konceptualny bazy danych . . . . .	8
Rysunek 4.	Model logiczny bazy danych . . . . .	9
Rysunek 5.	Model fizyczny bazy danych . . . . .	10
Rysunek 6.	Architektura aplikacji . . . . .	11
Rysunek 7.	Panel logowania do phpMyAdmin . . . . .	17
Rysunek 8.	Panel administracyjny usługi phpMyAdmin . . . . .	18
Rysunek 9.	Interfejs użytkownika . . . . .	19
Rysunek 10.	Lista wyboru użytkownika . . . . .	20
Rysunek 11.	Panel filtrowania dostępnych produktów . . . . .	20
Rysunek 12.	Widok tabeli towarów po filtracji . . . . .	21
Rysunek 13.	Stan zamówienia - powodzenie operacji . . . . .	23
Rysunek 14.	Stan zamówienia - niepowodzenie operacji . . . . .	23
Rysunek 15.	Walidacja danych po stronie aplikacji . . . . .	31
Rysunek 16.	Walidacja danych po stronie aplikacji z komunikatem . . .	31
Rysunek 17.	Rezultat testu triggera . . . . .	32

## Spis tabel

Tabela 1.	Tabela przypadków użycia . . . . .	3
Tabela 2.	Zestawienie wybranych funkcji systemu . . . . .	12
Tabela 3.	Tabela wybranych triggerów . . . . .	24

# 1 Wstęp

## 1.1 Cel projektu

Celem projektu jest zaprojektowanie systemu bazodanowego dla internetowego sklepu sportowego oraz implementacja aplikacji webowej umożliwiającej dokonywanie wybranych transakcji zarówno od strony klienta jak i sprzedawcy.

## 1.2 Zakres projektu

System pozwala sprzedawcy na dodawanie towarów do sklepu, wystawianie ich do sprzedaży, kontrolę ilości towaru dostępnej na magazynie oraz generowanie prostych raportów na temat sprzedaży. Kupujący ma możliwość wyszukiwania towaru, zakupu i listowania dokonanych zakupów wraz ze statusem zamówienia. Aplikacja jest dostępna w formie strony internetowej umieszczonej na serwerze i dostępnej po zalogowaniu użytkownika. Istnieje możliwość założenia konta w dwóch wariantach:

- Handlowca
- Klienta

Zależnie od rodzaju konta udostępniana jest określona wersja serwisu.

## 2 Analiza wymagań

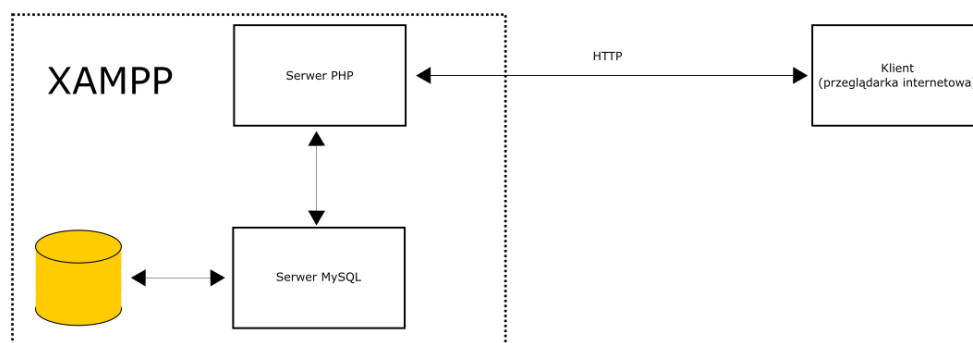
### 2.1 Opis działania i schemat logiczny systemu

#### Opis działania

System jest zrealizowany w oparciu o relacyjną bazę danych MySQL oraz interfejs dostępowy utworzony w języku PHP. Przetwarzanie danych odbywają się po stronie bazy danych, natomiast dla klienta udostępnione jest graficzne środowisko dostępne umożliwiające wydawanie żądanych zapytań. Wykonywanie poleceń w systemie bazodanowym nie wymaga od użytkownika znajomości języka SQL.

#### Schemat logiczny systemu

Schemat logiczny systemu znajduje się na Rysunku 1.



Rysunek 1. Schemat logiczny systemu

### 2.2 Wymagania funkcjonalne

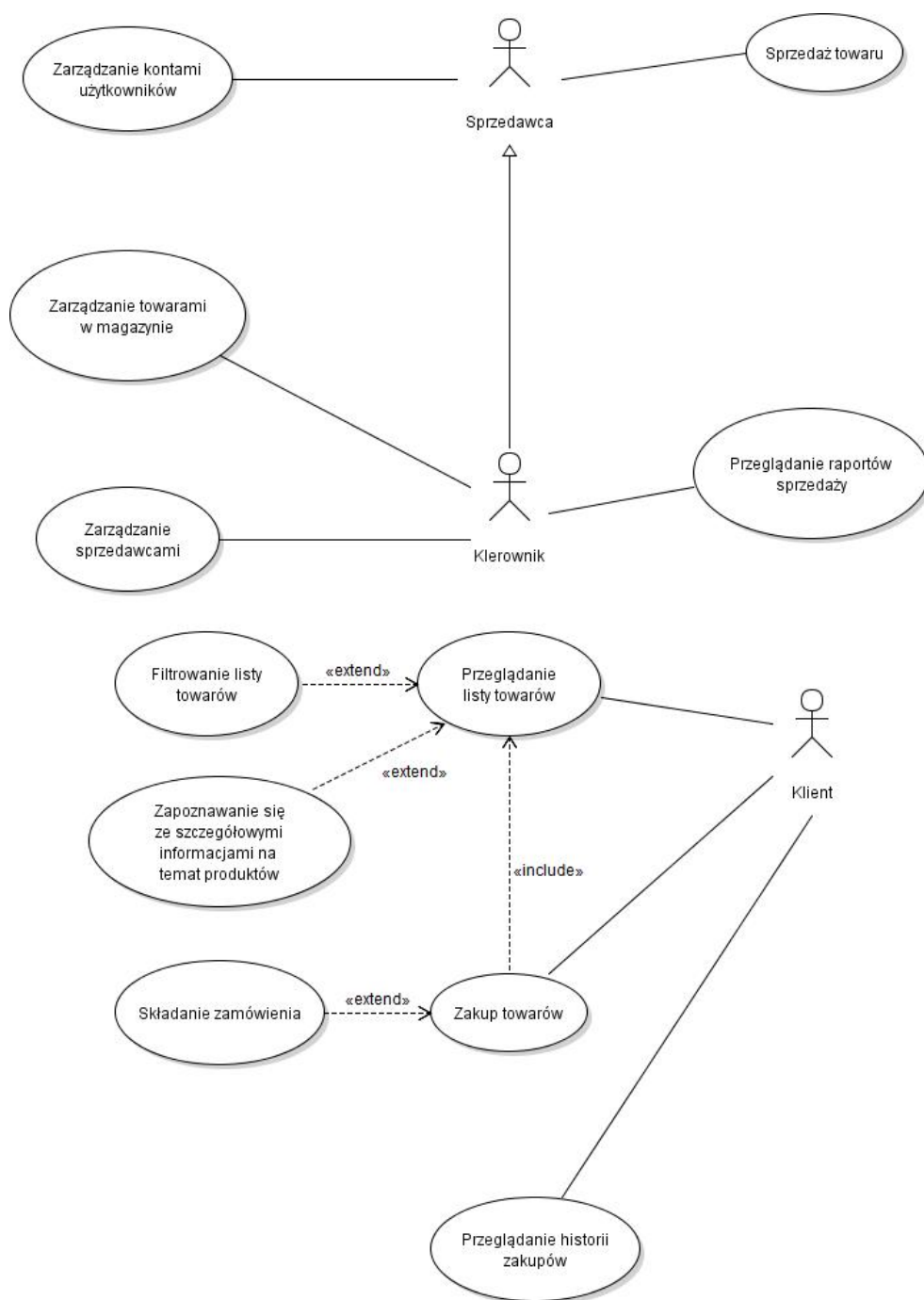
#### 2.2.1 Diagram przypadków użycia

Funkcjonalności dla poszczególnych użytkowników systemu zebrano w Tabeli 1. Diagram przypadków użycia zamieszczony został na Rysunku 2.

Tabela 1. Przypadki użycia dla poszczególnych użytkowników

Operacja/Użytkownik	Sprzedawca	Kierownik	Klient
Zarządzanie kontami użytkowników	Tak	Nie	Nie
Sprzedaż towaru	Tak	Nie	Nie
Zarządzanie towarami w magazynie	Nie	Tak	Nie
Zarządzanie sprzedawcami	Nie	Tak	Nie
Przeglądanie raportów sprzedaży	Nie	Tak	Nie
Przeglądanie listy towarów	Nie	Nie	Tak
Przeglądanie historii zakupów	Nie	Nie	Tak
Zakup towarów	Nie	Nie	Tak





Rysunek 2. Diagram przypadków użycia

### **2.2.2 Scenariusze wybranych przypadków użycia**

- Filtrowanie listy dostępnych towarów
  1. Użytkownik przegląda listę towarów.
  2. Użytkownik filtruje dostępne towary według producentów lub kategorii.
  3. System wykonuje zapytanie o towary z podanym kryterium.
  4. System przedstawia użytkownikowi znalezione towary.
  5. Użytkownik dodaje wybrane produkty do zamówienia.
- Złożenie zamówienia
  1. Użytkownik dodaje produkty do zamówienia.
  2. Użytkownik wybiera potwierdzenie zamówienia.
  3. System realizuje zamówienie.
  4. System powiadamia użytkownika o statusie operacji utworzenia zamówienia.

## **2.3 Wymagania niefunkcjonalne**

### **2.3.1 Wykorzystywane technologie i narzędzia**

Technologie:

- Implementacja systemu
  - PHP 7.0
  - MySQL 5.7.10
  - HTML 5
  - CSS 3
- Testy funkcjonalne
  - Java SE 8
  - JDBC
  - SeleniumHQ
- Dokumentacja
  - LaTeX

Narzędzia projektowania:

- MySQL 5.7.11
- MySQL Workbench 6.1

Narzędzia implementacji systemu:

- GitHub Desktop 3.0.15
- NetBeans IDE 8.1
- XAMPP 5.6.19

### **2.3.2 Wymagania dotyczące rozmiaru bazy danych**

Baza danych powinna przechowywać dane kilkuset artykułów sportowych wraz z ich aktualną ilością na magazynach. Część danych związana z administracją systemem w stosunku do ilości artykułów jest dużo mniejsza, w związku z czym w odniesieniu do rozmiaru bazy danych została zaniebana.

### **2.3.3 Wymagania dotyczące bezpieczeństwa systemu**

- System powinien być odporny na ataki typu SQL injection.
- Dostęp do bazy danych powinien odbywać się poprzez uwierzytelnienie użytkownika.
- Każdy użytkownik powinien mieć dostęp do danych, do których wglądu jest uprawniony.
- Użytkownicy, którzy nie muszą modyfikować pozycji w tabelach, nie powinni otrzymywać do tego uprawnień ze strony systemu.

## **2.4 Przyjęte założenia projektowe**

System powinien być prosty w obsłudze i intuicyjny dla użytkownika, dodatkowo musi prawidłowo realizować swoje funkcjonalności i prowadzić ciągły nadzór nad danymi wprowadzanymi przez użytkownika a także na bieżąco informować o wszelkich nieprawidłowościach. Czas przetwarzania danych oraz kierowanych zapytań nie może przekraczać kilku sekund.

## **3 Projekt systemu**

### **3.1 Projekt bazy danych**

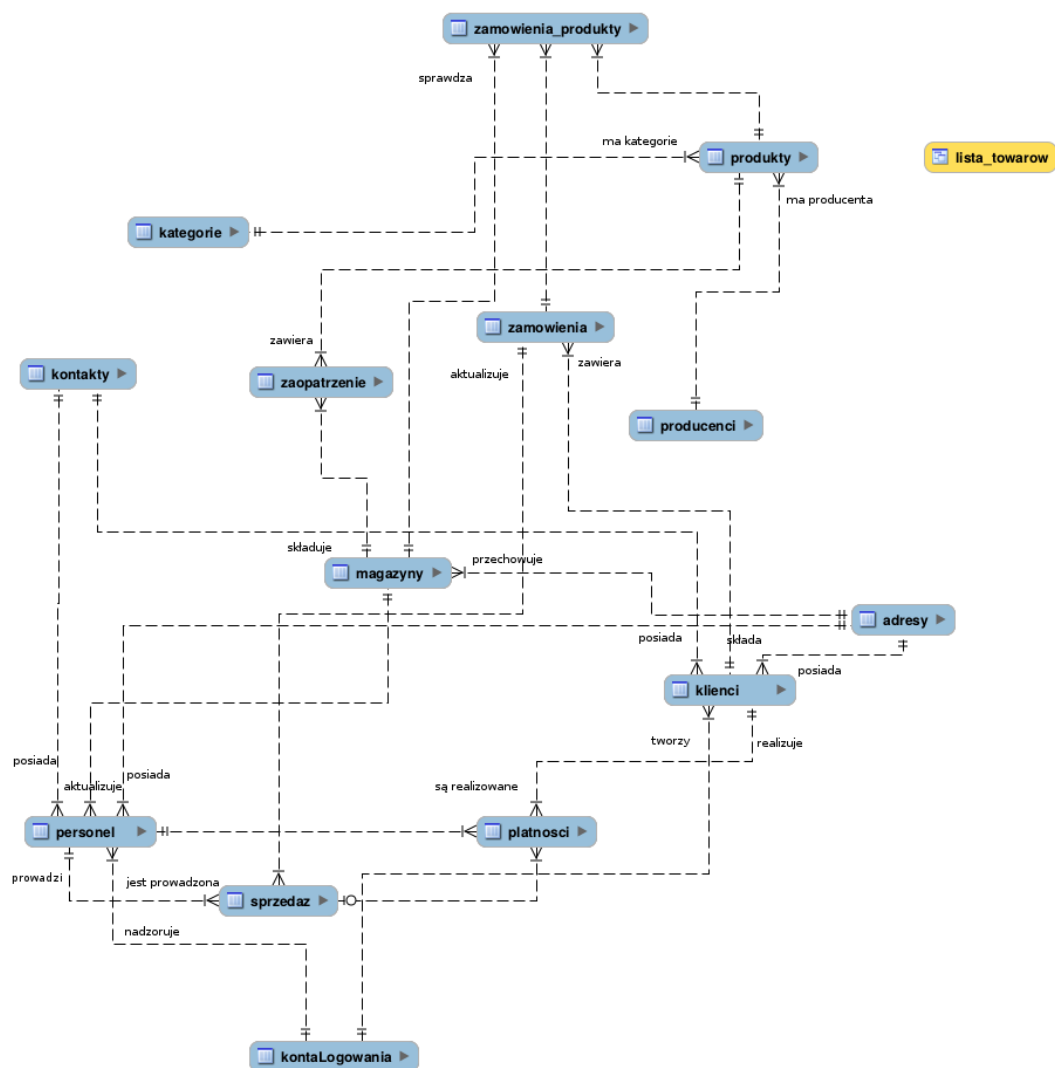
#### **3.1.1 Analiza rzeczywistości i uproszczony model konceptualny**

##### **Analiza rzeczywistości**

Sklep sportowy 'Sportpol' prowadzi sprzedaż detaliczną produktów sportowych na rynku krajowym. Zadaniem sklepu jest dostarczenie klientom szerokiego zakresu asortymentu oraz możliwości wygodnego i szybkiego dokonywania zakupów przez internet. Polityką firmy jest ciągle zwiększanie sprzedaży wysyłkowej.

##### **Model konceptualny**

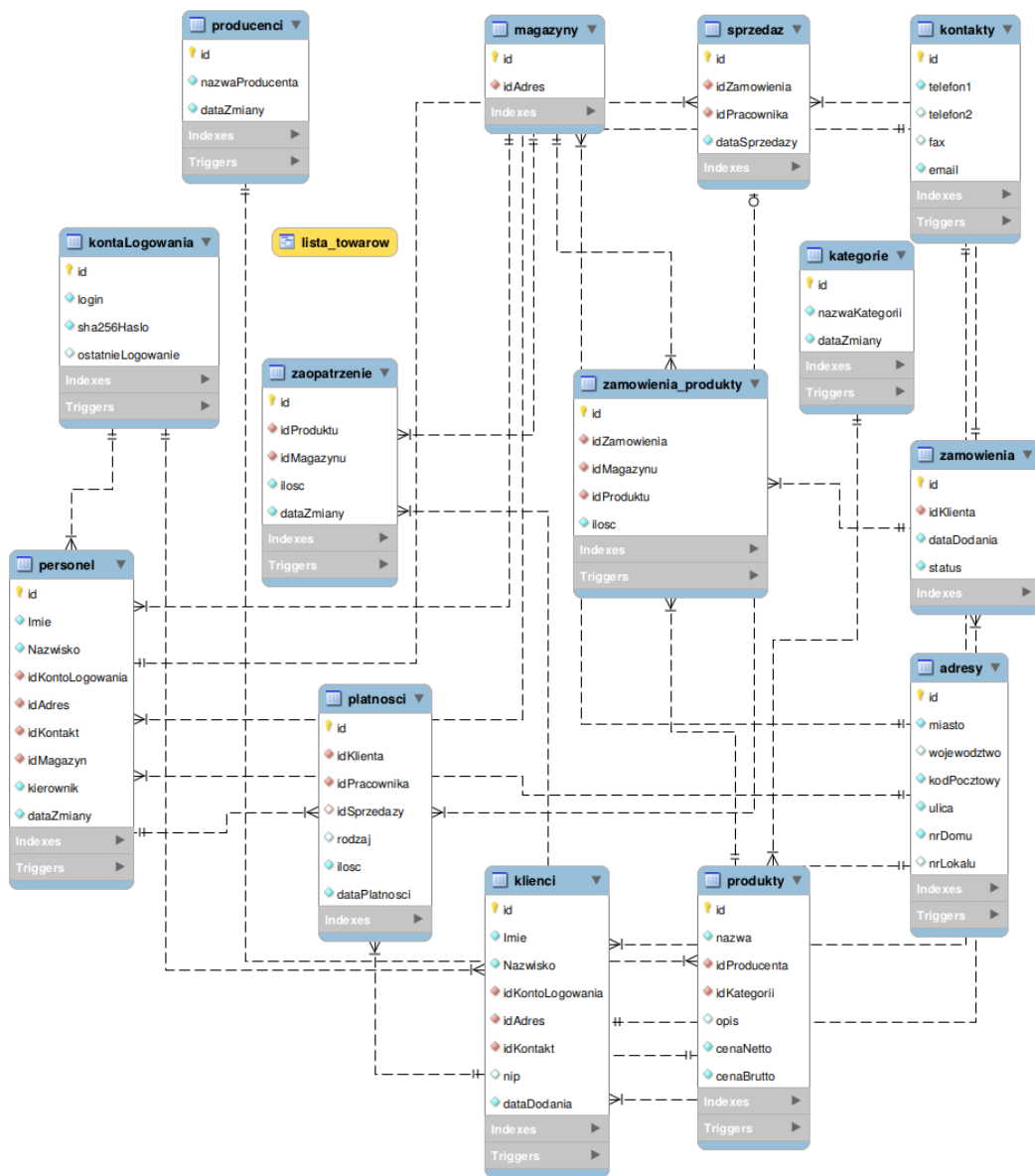
Model konceptualny bazy danych przedstawiono na Rysunku 3.



Rysunek 3. Model konceptualny bazy danych

### 3.1.2 Model logiczny i normalizacja

Model logiczny bazy danych zamieszczono na Rysunku 4.



Rysunek 4. Model logiczny bazy danych

### 3.1.3 Model fizyczny i ograniczenia integralności danych

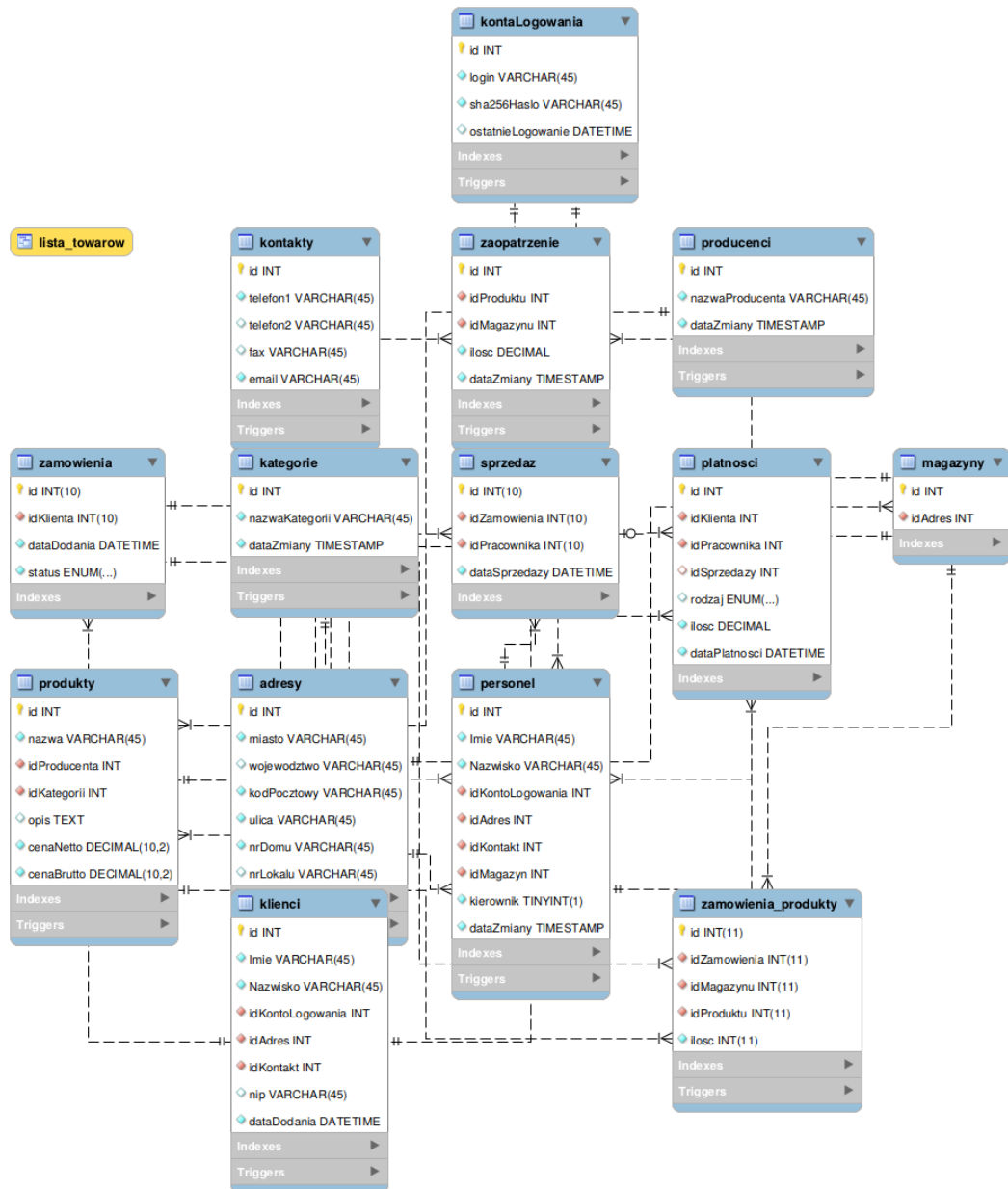
#### Ograniczenia integralności danych

Integralność i spójność danych powinna być zapewniona triggerami i procedurami. Produkty dodawane do zamówień powinny być jednocześnie odejmowane z magazynów. W przypadku gdy zamówienie nie może zostać zrealizowane, transakcja z bazą danych powinna zostać wycofana i nie modyfikować

stanu rekordów.

## Model fizyczny

Model fizyczny bazy danych zamieszczono na Rysunku 5.



Rysunek 5. Model fizyczny bazy danych

### 3.1.4 Inne elementy schematu – mechanizmy przetwarzania danych

- Przetwarzanie danych powinno odbywać się w większości przypadków po stronie bazy danych
- Użytkownik nie powinien uczestniczyć w przetwarzaniu danych między systemem a bazą danych
- Wymiana danych powinna odbywać się wyłącznie za pomocą mechanizmów takich jak:
  - Zapytania
  - Funkcje
  - Procedury
  - Widoki

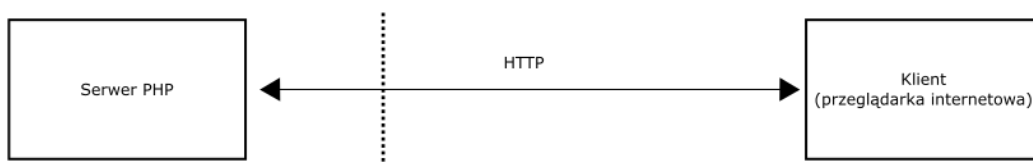
### 3.1.5 Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

Zabezpieczenia po stronie bazy danych powinny sprowadzać się wyłącznie do ograniczenia dostępu nieautoryzowanym użytkownikom oraz walidacji wprowadzanych danych za pomocą wyrażeń regularnych zaimplementowanych w postaci triggerów. Inne sposoby ochrony danych nie są wymagane, ze względu na walidację po stronie aplikacji dostępowej.

## 3.2 Projekt aplikacji użytkownika

### 3.2.1 Architektura aplikacji i diagramy projektowe

Aplikacja zbudowana jest w oparciu o część webową w postaci strony internetowej *HTML*, a także serwerową napisaną za pomocą języka *PHP*. Schemat połączeń pomiędzy elementami aplikacji przedstawiono na Rysunku 6.



Rysunek 6. Model architektoniczny aplikacji dostępowej



### 3.2.2 Interfejs graficzny i struktura menu

### 3.2.3 Projekt wybranych funkcji systemu

Tabela 2. Zestawienie wybranych funkcji systemu

Operacja	Dane wejściowe	Dane wyjściowe
Przeglądanie produktów	Lista <i>ID</i> producentów i lista <i>ID</i> kategorii	Przefiltrowana tabela produktów
Tworzenie zamówienia	<i>ID</i> klienta i lista towarów wraz z ilościami	Powodzenie operacji

### 3.2.4 Metoda podłączania do bazy danych – integracja z bazą danych

Komunikacja z bazą danych ze strony serwerowej odbywa się za pomocą biblioteki języka PHP, umożliwiającej wydawanie zapytań SQL.

### 3.2.5 Projekt zabezpieczeń na poziomie aplikacji

Wydawanie zapytań ze strony aplikacji możliwe jest wyłącznie za pomocą interfejsu dostępowego. Wszelkie operacje związane z bazą danych są ukryte poprzez interfejs użytkownika, w związku z czym nie jest możliwe uzyskanie dostępu do danych w sposób inny, niż ten oferowany przez aplikację.

## 4 Implementacja systemu

### 4.1 Realizacja bazy danych

#### 4.1.1 Tworzenie tabel i definiowanie ograniczeń

**Tworzenie tabel** Inicjalizacja systemu bazodanowego odbywa się za pomocą usługi phpmyadmin. W celu zalogowania do usługi, należy wybrać w przeglądarce internetowej adres <http://localhost/phpmyadmin> oraz wprowadzić dane logowania do formularza przedstawionego na Rysunku 7. Następnie należy wybrać z menu opcję *Import* (Rysunek 8), *Import z pliku .sql*, podać ścieżkę do przygotowanego pliku z bazą danych i zatwierdzić operację poprzez wybranie *OK*.

Listing 1. Przykładowe zapytanie tworzące tabelę

```
CREATE TABLE IF NOT EXISTS 'sklepbd'.'klienci' (  
  'id' INT UNSIGNED NOT NULL AUTOINCREMENT,  
  'Imie' VARCHAR(45) NOT NULL,  
  'Nazwisko' VARCHAR(45) NOT NULL,  
  'idKontoLogowania' INT UNSIGNED NOT NULL,  
  'idAdres' INT UNSIGNED NOT NULL,  
  'idKontakt' INT UNSIGNED NOT NULL,  
  'nip' VARCHAR(45) NULL,  
  'dataDodania' DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP  
,  
PRIMARY KEY ('id'),  
INDEX 'index1' ('idKontoLogowania' ASC),  
INDEX 'index2' ('idAdres' ASC),  
INDEX 'index3' ('idKontakt' ASC),  
CONSTRAINT 'fk_klienci_1'  
  FOREIGN KEY ('idKontoLogowania')  
  REFERENCES 'sklepbd'.'kontoLogowania' ('id')  
  ON DELETE NO ACTION  
  ON UPDATE NO ACTION,  
CONSTRAINT 'fk_klienci_2'  
  FOREIGN KEY ('idAdres')  
  REFERENCES 'sklepbd'.'adresy' ('id')  
  ON DELETE NO ACTION  
  ON UPDATE NO ACTION,  
CONSTRAINT 'fk_klienci_3'  
  FOREIGN KEY ('idKontakt')  
  REFERENCES 'sklepbd'.'kontakty' ('id')  
  ON DELETE NO ACTION
```

```
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

**Definiowanie ograniczeń** Ograniczenia dotyczące poprawności wprowadzanych danych przez użytkownika zaimplementowane są w postaci *triggerów*, które są automatycznie ładowane do bazy danych w chwili importowania przygotowanego pliku *.sql*.

Listing 2. Przykładowy wyzwalacz

```
DELIMITER //
```

```
CREATE DEFINER = CURRENT_USER TRIGGER `sklepbd`.``
  klienci_BEFORE_INSERT BEFORE INSERT ON `klienci` FOR
  EACH ROW
BEGIN
  IF NEW.Imie NOT REGEXP '^[A-Za-z]{3,100}$' THEN
    SIGNAL SQLSTATE '10007'
      SET MESSAGE_TEXT = '[tabla:klienci] - kolumna `Imie`
        jest niepoprawna!';
  END IF;

  IF NEW.Nazwisko NOT REGEXP '^[A-Za-z]{3,100}$' THEN
    SIGNAL SQLSTATE '10008'
      SET MESSAGE_TEXT = '[tabla:klienci] - kolumna `
        Nazwisko` jest niepoprawna!';
  END IF;

  IF NEW.nip NOT REGEXP '
    ^[0-9]{10}|[0-9]{3}\-[0-9]{3}\-[0-9]{2}\-[0-9]{2}$' THEN
    SIGNAL SQLSTATE '10008'
      SET MESSAGE_TEXT = '[tabla:klienci] - kolumna `nip`
        jest niepoprawna!';
  END IF;

END//
```

#### 4.1.2 Implementacja mechanizmów przetwarzania danych

Mechanizmy przetwarzania danych pomiędzy systemem bazodanowym a bazą danych zrealizowane są za pomocą *procedur*, *funkcji* oraz *widoków*. Wybrane mechanizmy stosowane w systemie zestawiono poniżej:

1. Procedura - dodajDoZamowienia

- Dodaje do zamówienia wybrane towary.
- Aktualizuje stan magazynów.
- W przypadku wyczerpania zasobów na magazynie, usuwa rekord z tabeli.

Listing 3. Przykładowa procedura

```

DELIMITER //
CREATE PROCEDURE dodajProduktDoZamowienia(
    IN id_produktu INT UNSIGNED,
    IN il INT UNSIGNED,
    IN id_klienta INT UNSIGNED,
    IN id_zamowienia INT UNSIGNED
)
BEGIN
    DECLARE zid INT UNSIGNED DEFAULT 0;
    DECLARE zil INT UNSIGNED DEFAULT 0;
    DECLARE zidm INT UNSIGNED DEFAULT 0;
    DECLARE zap CURSOR FOR SELECT id, idMagazynu,
        ilosc FROM zaopatrzenie WHERE idProduktu =
        id_produktu AND ilosc > 0 ORDER BY ilosc DESC
    ;
    OPEN zap;
    WHILE il > 0 DO
        FETCH zap INTO zid, zidm, zil;
        IF zil <= il THEN
            DELETE FROM zaopatrzenie WHERE id = zid;
            INSERT INTO zamowienia_produkty (
                idZamowienia, idMagazynu, idProduktu,
                ilosc) VALUES (id_zamowienia, zidm,
                id_produktu, zil);
            SET il = il - zil;
        ELSE
            UPDATE zaopatrzenie SET ilosc = (zil - il)
            WHERE id = zid;
            INSERT INTO zamowienia_produkty (
                idZamowienia, idMagazynu, idProduktu,
                ilosc) VALUES (id_zamowienia, zidm,
                id_produktu, il);
            SET il = 0;
        END IF;
    END WHILE;
    CLOSE zap;
END //
```

## 2. Procedura - filtrujProdukty

- Przyjmuje listę *ID* producentów i/lub kategorii.
- Filtruje dane w tabeli produktów względem zadanych kryteriów.
- Zwraca użytkownikowi wyłącznie dopasowane rekordy.

## 3. Funkcja - otwórzZamówienie

- Tworzy rekord w tabeli z zamówieniami.
- Zwraca id utworzonego zamówienia.

Listing 4. Przykładowa funkcja

```
DELIMITER //
```

```
CREATE FUNCTION otworzZamowienie(  
    id_klienta INT UNSIGNED  
)  
RETURNS INT UNSIGNED  
NOT DETERMINISTIC  
BEGIN  
    INSERT INTO zamowienia (idKlienta , status)  
        VALUES (id_klienta , 'przyjete');  
    RETURN LAST_INSERT_ID();  
END //
```

## 4. Widok - listaTowarów

- Dołącza do tabeli produktów
  - Nazwy producentów
  - Nazwy kategorii
  - Informacje o stanie magazynów

### 4.1.3 Implementacja uprawnień i innych zabezpieczeń

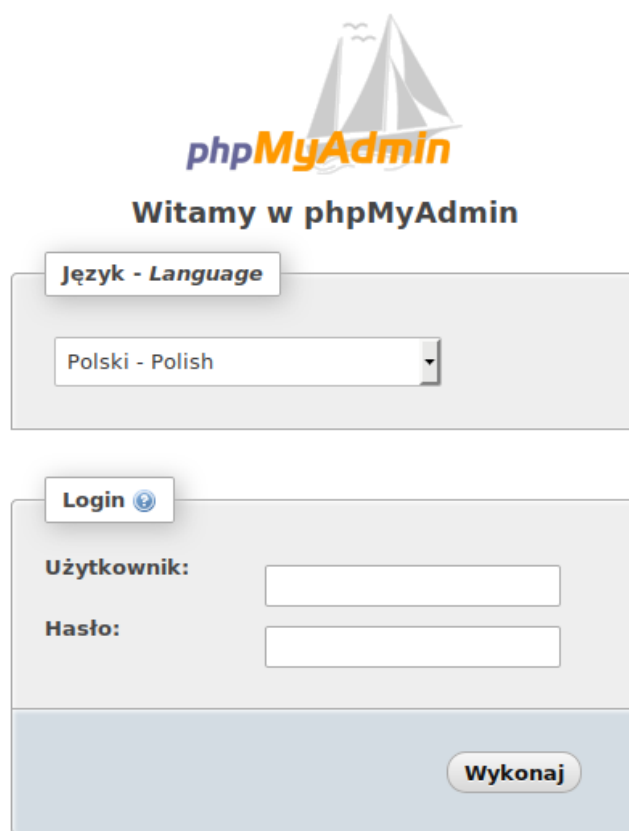
#### Dostęp do bazy danych

Dostęp do bazy danych odbywa się za pośrednictwem usługi phpMyAdmin, świadczonej jako usługa webowa pod adresem <http://localhost/phpmyadmin> lub <http://127.0.0.1/phpmyadmin>. W celu uzyskania dostępu do bazy danych, należy przejść przez procedurę uwierzytelnienia przedstawioną na Rysunku 7.

### Uwierzytelnianie w systemie

Każdy użytkownik serwisu posiada własne konto w systemie, wraz z przydzielonymi uprawnieniami do wglądu bądź modyfikacji danych. Powyższa funkcjonalność pozwala ograniczyć przypadki ingerencji osób nieupoważnionych.

Serwis umożliwia także wygodną administrację dla użytkownika *root*, co przedstawiono na Rysunku 8.



phpMyAdmin

Witamy w phpMyAdmin

Język - Language

Polski - Polish

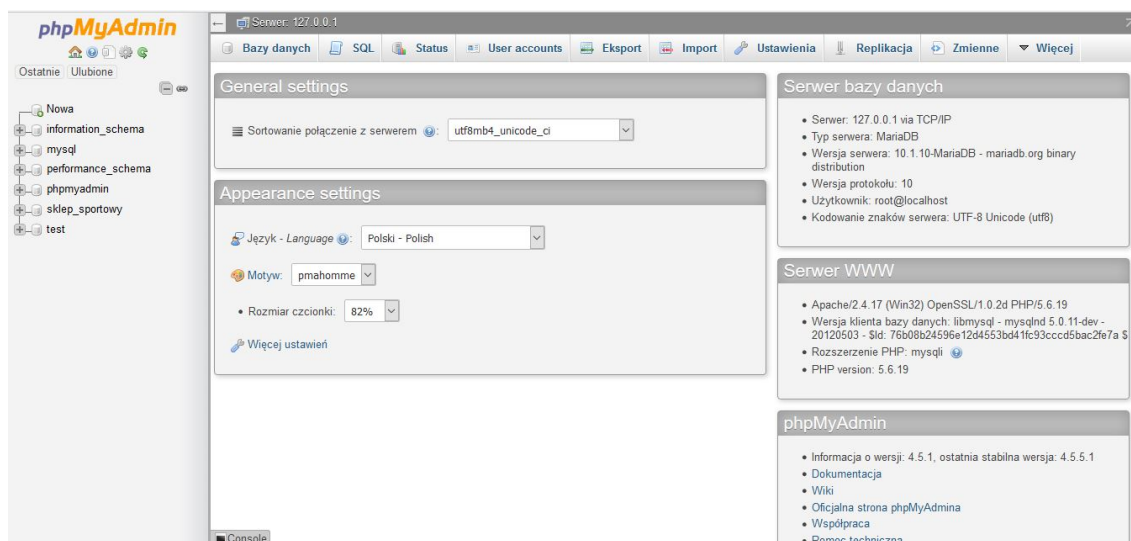
Login ?

Użytkownik:

Hasło:

Wykonaj

Rysunek 7. Panel logowania do usługi phpMyAdmin



Rysunek 8. Panel administracyjny usługi phpMyAdmin

## 4.2 Realizacja elementów aplikacji

### 4.2.1 Walidacja i filtracja

System zapewnia walidację po stronie aplikacji jak i bazy danych. Uzupełnianie pól w aplikacji jest analizowane z poziomu interfejsu użytkownika oraz w przypadku nieprawidłowości, natychmiast kierowane są do użytkownika stosowne ostrzeżenia. Po stronie systemu bazodanowego, przed złożeniem zamówienia, sprawdzany jest stan produktów na magazynach. W przypadku niewystarczającej ilości towarów, zamówienie jest wycofywane z systemu i użytkownik otrzymuje powiadomienie o niepowodzeniu operacji.

### 4.2.2 Implementacja interfejsu dostępu do bazy danych

Dostęp do systemu odbywa się za pomocą interfejsu przedstawionego na Rysunku 9.

Klient:  
Jan Kowalski

Kategoria	Nazwa	Producent	Cena brutto	Dostępne	Zamówienie
tenis	rakieta czarna, z grafenu	ZIMBABWE CONTINENTAL	102.86	1	0
tenis	siatka 100m, nylonowa, biała	ZIMBABWE CONTINENTAL	12.86	Brak towaru!	
wspinaczka	lina 25 m	MKPZ WISLA	121.70	2	0
tenis	zbierek czarna, z grafenu	ZIMBABWE CONTINENTAL	23.86	35	0
tenis	piłki do tenisa zółte, 18 szt.	EIGENWERT	22.50	53	0
tenis	rakieta czarna, z grafenu	ZIMBABWE CONTINENTAL	102.86	17	0
tenis	siatka 100m, nylonowa, biała	ZIMBABWE CONTINENTAL	12.86	11	0
baseball	kij baseballowy stalowy	ZIMBABWE CONTINENTAL	80.90	Brak towaru!	
baseball	kij baseballowy aluminiowy	ZIMBABWE CONTINENTAL	80.90	Brak towaru!	
wspinaczka	kask zielony, czerwony, żółty	MKPZ WISLA	0.90	2	0
baseball	piłki do baseballa białe, 2 szt.	EIGENWERT	62.86	34969	0
tenis	buty do tenisa rozmiary: 12-80	EIGENWERT	122.86	34558	0

Zamów

**Producent**

- ☐ WIERTEX
- ☐ PRO ONE
- ☒ ZIMBABWE CONTINENTAL
- ☐ TRANSBARD
- ☒ MKPZ WISLA
- ☐ DIO
- ☐ DEM BA DUR
- ☒ EIGENWERT
- ☐ AEROPLANT
- ☐ WINDPOL
- ☐ WINDPOL CONTINENTAL

**Kategoria**

- ☐ football
- ☐ tenis
- ☐ pływanie
- ☐ baseball
- ☐ wspinaczka

Filtruj

Rysunek 9. Interfejs użytkownika aplikacji dostępowej

Użytkownik ma możliwość wyboru z listy rozwijanej (Rysunek 10) nazwy konta na które będzie realizowane zamówienie. Dodatkowo istnieje możliwość filtrowania listy aktualnie wyświetlonych produktów za pomocą panelu znajdującego się na Rysunku 11.

Listing 5. Połączenie z bazą danych z poziomu PHP

```
<?php
$link = mysqli_connect('localhost', 'root', 'admin', '
    sklepbd');
if (!$link){
    die("<h1>Connection error</h1>");
}
mysqli_set_charset($link, "utf8");
?>
```



Klient:  
Jan Kowalski

Rysunek 10. Lista wyboru użytkownika składającego zamówienie

### Producent

- ☐ WIERTEX
- ☐ PRO ONE
- ☒ ZIMBABWE CONTINENTAL
- ☐ TRANSBARD
- ☒ MKPZ WISLA
- ☐ DIO
- ☐ DEM BA DUR
- ☒ EIGENWERT
- ☐ AEROPLANT
- ☐ WINDPOL
- ☐ WINDPOL CONTINENTAL

### Kategoria

- ☐ football
- ☐ tenis
- ☐ plywanie
- ☐ baseball
- ☐ wspinaczka

Filtruj

Rysunek 11. Panel filtrowania dostępnych produktów

#### 4.2.3 Implementacja wybranych funkcjonalności systemu

##### Przeglądanie listy towarów

System wykonuje zapytanie do bazy danych o zestawienie danych dotyczących produktów wraz z podaniem aktualnego stanu ze wszystkich magazynów. Zapytanie sprowadza się do wywołania procedury *filtrujProdukty*, która zwraca widok *lista\_towarow* z posumowanymi stanami ze wszystkich magazynów. Ponadto, jeżeli jako parametry procedury zostanie podana lista id producentów i/lub lista id kategorii, zwrócone zostaną jedynie te rekordy, których kategorie czy produkty znajdują się na podanych listach.

##### Filtrowanie produktów

Podczas przeglądania tabeli dostępnych produktów użytkownik może dokonać filtracji poprzez wybranie w panelu bocznym z Rysunku 11 producentów bądź kategorii. Po naciśnięciu przycisku *Filtruj* wywoływana jest procedura

*filtrujProdukty* zwracająca widok. Następnie dane są wyświetlane użytkownikowi. Przykładowy wynik operacji zamieszczono na Rysunku 12.

Klient:

Jan Kowalski

Kategoria	Nazwa	Producent	Cena brutto	Dostępne	Zamówienie
tenis	rakieta czarna, z grafenu	ZIMBABWE CONTINENTAL	102.86	1	0
tenis	siatka 100m, nylonowa, biała	ZIMBABWE CONTINENTAL	12.86	Brak towaru!	
wspinaczka	lina 25 m	MKPZ WISLA	121.70	2	0
tenis	zbierek czarna, z grafenu	ZIMBABWE CONTINENTAL	23.86	35	0
tenis	piłki do tenisa zółte, 18 szt.	EIGENWERT	22.50	53	0
tenis	rakieta czarna, z grafenu	ZIMBABWE CONTINENTAL	102.86	17	0
tenis	siatka 100m, nylonowa, biała	ZIMBABWE CONTINENTAL	12.86	11	0
baseball	kij baseballowy żelazny	ZIMBABWE CONTINENTAL	80.90	Brak towaru!	
baseball	kij baseballowy aluminiowy	ZIMBABWE CONTINENTAL	80.90	Brak towaru!	
wspinaczka	kask niebieski, czerwony, żółty	MKPZ WISLA	0.90	2	0
baseball	piłki do baseballa białe, 2 szt.	EIGENWERT	62.86	34969	0
tenis	buty do tenisa rozmiary: 12-80	EIGENWERT	122.86	34558	0

Zamów

Producent

☐ WIERTEX  
☐ PRO ONE  
☒ ZIMBABWE CONTINENTAL  
☐ TRANSBARD  
☒ MKPZ WISLA  
☐ DIO  
☐ DEM BA DUR  
☒ EIGENWERT  
☐ AEROPLANT  
☐ WINDPOL  
☐ WINDPOL CONTINENTAL

Kategoria

☐ football  
☐ tenis  
☐ pływanie  
☐ baseball  
☐ wspinaczka

Filtruj

Rysunek 12. Widok tabeli towarów po filtracji

Listing 6. Wywołanie i wyświetlenie wyników procedury *filtrujProdukty*

```

<?php
function wareList() {
    $pros = "";
    if (isset($_GET['producer'])) {
        $pros = implode(",", $_GET['producer']);
    }
    $cats = "";
    if (isset($_GET['category'])) {
        $cats = implode(",", $_GET['category']);
    }
    return "CALL filtrujProdukty('$pros', '$cats');";
}
$wynik = mysqli_query($link, wareList());
print "<table class='border' style='width:100%;'>"
    . "<thead>"
    . "<tr>"
    . "<th>Kategoria</th>"
    . "<th>Nazwa</th>"
    . "<th>Producent</th>"
    . "<th>Cena brutto</th>"
    . "<th>Dostępne</th>"
    . "<th>Zamówienie</th></tr>"

```

```

        . "</thead>"
        . "<tbody>";
if ($wynik){
    while($rekord = mysqli_fetch_assoc($wynik)){
        $id = $rekord['idProduktu'];
        $nazwa = $rekord['nazwa'];
        $producent = $rekord['nazwaProducenta'];
        $kategoria = $rekord['nazwaKategorii'];
        $opis = $rekord['opis'];
        $cenaBrutto = $rekord['cenaBrutto'];
        $ilosc = $rekord['ilosc'];
        $zamowienie = "";
        if($ilosc != NULL)
            $zamowienie = "<input type='number' id='order '
                name='$id' min='0' max='".($ilosc==null ? 0 :
                    $ilosc)." ' value='0'>";
        print "<tr class='product' id='$id'>
<td id='kategoria'>$kategoria</td>
<td id='nazwa'>$nazwa</td><br /><td style='
font-size:10pt;'>$opis</td>
<td id='producent'>$producent</td>
<td>$cenaBrutto</td>
<td id='ilosc'>".($ilosc==null ? "Brak towaru!" :
    $ilosc)."</td>
<td>$zamowienie</td></tr>";
    }
}
print "</tbody></table></body></html>";
?>

```

## Tworzenie zamówienia

Po wybraniu przez klienta produktów wraz z ilościami i naciśnięciu przycisku *Zamów* zostaje otwarta transakcja do bazy danych. Pierwsza w tej transakcji jest wywoływana funkcja *otworzZamowienie*, która tworzy wpis w tabeli *zamowienia* i zwraca jego id. Następnie każdy z zamawianych produktów jest sprawdzany pod kątem dostępności i jeżeli jest go wystarczająca ilość wywoływana jest dla niego procedura *dodajDoZamowienia*, która odejmuje odpowiednią ilość z tabeli *zaopatrzenie* (bądź kasuje rekord, jeżeli ilość spadła do zera) i tworzy nowy rekord w tabeli *zamowienia\_produkty*. Jeżeli przy jednym z produktów zdarzy się, że jest go niewystarczająca ilość, następuje rollback transakcji. W innym wypadku cała transakcja jest commitowana do bazy. Użytkownik jest informowany o powodzeniu (rys. 13) lub niepowodze-

niu (rys. 14) operacji.

Listing 7. Skrypt PHP odpowiadający za stworzenie zamównienia

```
<?php
mysql_query( 'START TRANSACTION; ' );
$id_klienta = $_GET[ 'client' ];
$id_zamowienia = mysql_fetch_row( mysql_query( "SELECT
    otworzZamowienie( $id_klienta );" ) ) [0];
$failure = false;
$empty = true;
foreach ( $_GET as $key => $value ) {
    if ( is_numeric( $key ) !== false && $value !== 0 ) {
        $empty = false;
        $wynik = mysql_query( "SELECT SUM(ilosc) ilosc FROM
            zaopatrzenie WHERE idProduktu = $key GROUP BY
            idProduktu;" );
        if ( mysql_fetch_assoc( $wynik ) [ 'ilosc' ] >= $value ) {
            mysql_query( "CALL dodajProduktDoZamowienia( $key,
                $value, $id_klienta, $id_zamowienia );" );
        } else {
            $failure = true;
            break;
        }
    }
}
if ( $failure || $empty ) {
    print( "Błąd w zamówieniu!" );
    mysql_query( 'ROLLBACK;' );
} else {
    print( "Zamowienie zostało stworzone." );
    mysql_query( 'COMMIT;' );
}
?>
```

**Zamowienie zostało stworzone.**

Rysunek 13. Komunikat o stanie utworzonego zamówienia - powodzenie operacji

**Błąd w zamówieniu!**

Rysunek 14. Komunikat o stanie utworzonego zamówienia - niepowodzenie operacji

#### 4.2.4 Implementacja mechanizmów bezpieczeństwa

##### Sprawdzanie stanu magazynów

Mechanizmy bezpieczeństwa systemu opierają się o sprawdzenie ilości stanu danych na magazynie, co sprowadza się do kierowania zapytań do bazy danych o aktualny stan towarów i uniemożliwienie użytkownikowi wybrania większej ilości niż jest dostępne.

##### Triggery

Dodatkowym zaimplementowanym mechanizmem są trigery pozwalające na walidację rekordów bazy poprzez zastosowanie wyrażeń regularnych. W systemie są wykorzystywane podczas wstawiania rekordów do tabel, w przypadku niezgodności z wyrażeniem rekord jest odrzucany. Wybrane trigery zaimplementowane w systemie zestawiono w Tabeli 3.

Tabela 3. Tabela wybranych triggerów zaimplementowanych w systemie

Trigger	Wyrażenie regularne
Kod pocztowy	$\wedge [0-9]\{2\}\backslash-[0-9]\{3\}$
Ulica	$\wedge ((ul.   al.   os.   rondo   zaułek   skwer) \backslash ) ([A-Za-z]\{2,100\} \backslash ?)^+$
NIP	$\wedge [0-9]\{10\}   [0-9]\{3\} \backslash-[0-9]\{3\} \backslash-[0-9]\{2\} \backslash-[0-9]\{2\}$
Telefon	$\wedge (\\+?[0-9]\{1,4\}-?)?[0-9]\{3,10\}$
Nazwa producenta	$\wedge ([A-Za-z0-9]\{2,100\} \backslash ?)\{1,100\}$
Hasło sha256	$\wedge [0-9a-f]\{32\}$

## 5 Testowanie systemu

### 5.1 Instalacja i konfigurowanie systemu

#### Do pobrania

1. XAMPP - <https://www.apachefriends.org/pl/index.html>
2. GitHub - <https://desktop.github.com/>
3. NetBeans dla PHP - <https://netbeans.org/downloads/>

#### XAMPP

1. Zainstaluj XAMPP'a
2. Otwórz „XAMPP Control Panel”
3. W linijce „Apache” wybierz „Start”
4. Wejdź do przeglądarki internetowej
5. Wejdź na <http://localhost> (ewentualnie <http://localhost:80>). Jeżeli zobaczyłeś stronę internetową, to znaczy, że się udało.

#### Github

1. Pobierz projekt z repozytorium <https://github.com/shortname/dazybanych/archive/master.zip>
2. Rozpakuj pobrane archiwum w wybranej lokalizacji.
3. Znajdź w folderze, gdzie zainstalowałeś XAMPP'a plik `apache/conf/extra/httpd-vhosts.conf`
4. Dopisz na końcu znalezionej pliku tekst

```
<VirtualHost *:80>
    DocumentRoot "XXX"
    ServerName nacao.local
    ErrorLog "logs/nacao.localhost-error.log"
    <Directory "XXX">
        Options Indexes FollowSymLinks Includes ExecCGI
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

- , zamieniając XXX na ścieżkę do katalogu z pobranym projektem.
5. Uruchom serwer „Apache” z poziomu „XAMPP Control Panel”
  6. Wejdź do przeglądarki internetowej
  7. Wejdź na <http://localhost>. Jeżeli zobaczyłeś stronę „Hello world!”, to znaczy, że operacja powiodła się

## MySQL Server

1. Zainstaluj XAMPP.
2. Otwórz „XAMPP Control Panel”
3. W liniach „Apache”, „MySQL” wybierz „Start”
4. Otwórz przeglądarkę internetową i wejdź na <http://localhost/phpmyadmin> i sprawdź czy pojawiło się okno logowania z Rysunku 7.
5. Wejdź w zakładkę „User accounts”, zaznacz wszystkich użytkowników poza „root” na „localhost” i kliknij „Wykonaj” u dołu strony.
6. W ustawieniach zmień hasło na „admin” i kliknij „Wykonaj”. Po odświeżeniu strony powinien pokazać się następujący komunikat: „Nie udało się nawiązać połączenia: błędne ustawienia.”.
7. Znajdź w folderze instalacyjnym XAMPP’a plik phpMyAdmin/config.inc.php
8. W znalezionym pliku zmień wartość przypisania:  
  

```
$cfg[ 'Servers ' ][ $i ][ 'password ' ] = '';
```

  
na 'admin'
9. Po odświeżeniu strony powinien zobaczyć panel administracyjny bazy danych.
10. Z górnego menu wybierz „Import”. W widoku importu wybierz plik z Baza Danych/sklep\_sportowy.sql, upewnij się, że ustawione jest kodowanie znaków jako „utf-8”. Kliknij „Wykonaj”.
11. Otwórz przeglądarkę i wejdź na <http://localhost/login.php>
12. Po zalogowaniu („admin”, „admin”) powinna wyświetlić się lista towarów. Wszystkie polskie znaki powinny być wyświetlane poprawnie.

## 5.2 Testowanie opracowanych funkcji systemu

Testowanie funkcjonalności odbywało się za pomocą środowiska *Selenium*, dzięki któremu można było symulować zachowanie użytkownika naciskającego kolejne elementy interfejsu dostępowego. Dodatkowo do komunikacji z bazą danych wykorzystano framework *JDBC*, za pomocą którego kierowane były zapytania do bazy danych z języka *JAVA*. Przed przeprowadzeniem każdego testu baza danych była ładowana od nowa, w celu uniknięcia nieprawidłowości po poprzednich testach.

### 5.2.1 Testowanie zamówienia

#### Zależności do sprawdzenia

- Modyfikacje wprowadzane przez użytkownika modyfikują bazę danych.
- Wielu użytkowników może wykonywać zamówienia bez kolizji.
- Zamówienie jest zapisywane w bazie danych.
- Stan magazynów jest aktualizowany po złożeniu każdego z zamówień.
- Ilość produktów w zamówieniu odzwierciedla ilość podaną przez użytkownika.
- Wszystkie zamówione produkty znajdują się w zamówieniu.

#### Przebieg testu

1. Zresetowanie bazy do stanu wyjściowego.
2. Pobranie rekordów dotyczących modyfikowanych produktów z tabeli zaopatrzenie.
3. Zasymulowanie zmian na podanych rekordach oraz na tabelach ‘zamowienia’ i ‘zamowienia\_produkty’.
4. Wejście na stronę z listą towarów.
5. Wyklikanie pierwszego zamówienia (ustawienie klienta, wpisanie ilości produktów).
6. Wejście na stronę z listą towarów.
7. Zasymulowanie zmian na podanych rekordach oraz na tabelach ‘zamowienia’ i ‘zamowienia\_produkty’.
8. Porównanie stanu bazy danych przed i po transakcji.



### 5.2.2 Testowanie filtrowania

#### Zależności do sprawdzenia

- Kategorie oraz producenci zostali przypisani prawidłowo.
- Suma ilości artykułów z magazynów jest prawidłowa.
- Bez filtrowania wyświetlają się wszystkie produkty.
- Przy podaniu kryteriów wynik działania obu będzie uwzględniony.

#### Przebieg testu

1. Stworzenie zamówienia dla dwóch różnych klientów.
2. Sprawdzenie czy dane z tabel zamówienia, zamówienia\_produkty oraz zaopatrzenie zostały zmodyfikowane.

Listing 8. Fragment przykładowego testu

```
@BeforeClass
public void init () {
    super.init ();
    db = new DBConnector ();
    db.reset ();
    firstOrder = new HashMap<>();
    firstOrder.put (12, 236);
    firstOrder.put (13, 10);
    firstOrder.put (2, 44);
    firstOrder.put (4, 1);
    secondOrder = new HashMap<>();
    secondOrder.put (23, 6);
    secondOrder.put (27, 100);
    productIds = new ArrayList<>(firstOrder.keySet ());
    productIds.addAll (secondOrder.keySet ());
    storageDataStore = db.findStorageDataOfProducts (
        productIds);
}

@Test
public void shouldCreateTwoOrdersAndUpdateStorageData () {
    //given
    List<Order> expectedOrders = createExpectedOrders ();
```

```

List<OrderDetails> expectedFirstOrderDetails =
    createExpectedOrder(firstOrder , 1);
List<OrderDetails> expectedSecondOrderDetails =
    createExpectedOrder(secondOrder , 2);

//when
order(firstOrder , 1);
order(secondOrder , 2);

//then
List<Order> actualOrders = db.findOrders();
Assertions.assertThat(actualOrders).
    containsOnlyElementsOf(expectedOrders);
Assertions.assertThat(db.findOrderDetails(actualOrders.
    get(0).getId())).containsOnlyElementsOf(
    expectedFirstOrderDetails);
Assertions.assertThat(db.findOrderDetails(actualOrders.
    get(1).getId())).containsOnlyElementsOf(
    expectedSecondOrderDetails);
Assertions.assertThat(db.findStorageDataOfProducts(
    productIds).getStorageData()).containsOnlyElementsOf(
    (storageDataStore.getStorageData()));
}

private void order(Map<Integer , Integer> details , int
clientId){
    getTo("/");
    chooseClient(clientId);
    details.entrySet().stream().forEach(detail ->
        typeOrderAmount(detail.getKey() , detail.getValue()))
    ;
    clickZamow();
}

private List<OrderDetails> createExpectedOrder(Map<Integer ,
Integer> details , int orderId){
    List<OrderDetails> result = new ArrayList<>();
    details.entrySet().stream().forEach(detail -> result.
        addAll(storageDataStore.realizeOrder(detail.getKey()
        , detail.getValue() , orderId)));
    return result;
}

```

```

private List<Order> createExpectedOrders(){
    List<Order> result = new ArrayList<>();
    result.add(new Order(0, 1, "przyjete"));
    result.add(new Order(0, 2, "przyjete"));
    return result;
}

private List<OrderDetails> createExpectedSecondOrderDetails
(){
    List<OrderDetails> result = new ArrayList<>();
    result.add(new OrderDetails(2, 23, 6));
    result.add(new OrderDetails(3, 27, 100));
    return result;
}

```

### 5.3 Testowanie mechanizmów bezpieczeństwa

Zabezpieczenia interfejsu użytkownika przed niepoprawnym formatem danych zostały sprawdzone pod kątem przekroczenia stanu towarów na magazynie oraz poprawnej sygnalizacji błędów ze strony aplikacji. Rezultat testu przedstawiono na Rysunku 15 i 16.

	Cena brutto	Dostępne	Zamówienie
	102.86	1	0
	12.86	Brak towaru!	
	121.70	2	0
	23.86	35	1000
	22.50	53	10000
	102.86	17	0
	12.86	11	0
	80.90	Brak towaru!	
	80.90	Brak towaru!	
	0.90	2	10
	62.86	34969	0
	122.86	34558	0

Rysunek 15. Walidacja danych po stronie aplikacji

a brutto	Dostępne	Zamówienie
	1	0
	Brak towaru!	
	2	0
	35	1000
	53	Proszę wybrać wartość nie większą niż 35
	17	0
	11	0
	Brak towaru!	
	Brak towaru!	

Pr

- ☐ V
- ☐ P
- ☒ Z
- ☐ T
- ☒ M
- ☐ E
- ☐ E
- ☒ E
- ☐ A
- ☐ V
- ☐ V

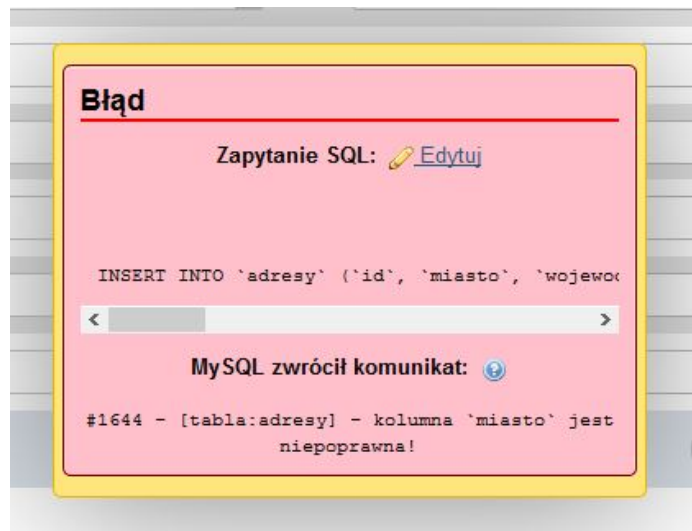
Ka

- ☐ f
- ☐ t
- ☐ p

Rysunek 16. Walidacja danych po stronie aplikacji z komunikatem

Sprawdzono także odporność bazy danych na nieporawny format danych

w przypadku błędu ze strony aplikacji dostępowej. W tym celu podjęta została próba ręcznego wprowadzenia danych w formacie niezgodnym ze specyfikacją. Rezultat powyższej operacji przedstawiony został na Rysunku 17.



Rysunek 17. Komunikat informujący o nieprawidłowym formacie danych

Weryfikacja odporności bazy danych na ataki SQL Injection nie była konieczna, ze względu na zabezpieczenie bazy danych przed nieupoważnionymi użytkownikami oraz ukryciem istnienia bazy danych przed klientem. Wszystkie zapytania obsługiwane są za pomocą aplikacji dostępowej, użytkownik nie ma możliwości wykonania zapytania bez pośrednictwa systemu.

## 5.4 Wnioski z testów

Testy zweryfikowały poprawność działania zaimplementowanych funkcjonalności. Ich różnorodność pozwoliła na sprawdzenie systemu na różnych poziomach abstrakcji. Zastosowanie testów sprawdzających działanie interfejsu poprzez symulowanie zachowania użytkownika pozwoliło na sprawdzenie systemu w środowisku jak najbardziej zbliżonym do rzeczywistego użytkowania. Wszystkie testy zwróciły wynik pozytywny.

## 6 Podsumowanie

### **Cel projektu**

Celem projektu było zaprojektowanie oraz zaimplementowanie relacyjnej bazy danych na potrzeby sportowego sklepu internetowego i stworzenie prostej webowej aplikacji do obsługi wybranych funkcjonalności. Dodatkowo, w celu weryfikacji poprawności działania systemu z bazą danych, przeprowadzone zostały wielopoziomowe testy integracyjne.

### **Analiza wymagań**

Pierwszym etapem pracy było zebranie wymagań od klienta oraz przeprowadzenie analizy rzeczywistości. Na podstawie zebranych informacji określone zostały przypadki użycia i zebrane w formie diagramu. Kolejnym etapem fazy projektowej było stworzenie modelu conceptualnego, który obrazował relacje zachodzące pomiędzy zamodelowanymi tabelami. Gdy powyższe zależności zostały określone, rozpoczęto prace nad diagramem logicznym, pozwalającym dokładnie określić relacje pomiędzy encjami, oraz modelem fizycznym, dokładnie opisującym typy danych i fizyczną strukturę działającego systemu bazodanowego.

### **Projekt systemu**

W fazie projektowej dobrane zostały narzędzia i technologie za pomocą których zrealizowana została strona bazy danych jak i aplikacja dostępowa. Wymagania technologiczne wyspecyfikowane zostały w kontekście tematu zadania projektowego, dzięki czemu na etapie projektowania udało się zapobiec późniejszym problemom integracyjnym oraz lepiej dopasować efekt końcowy do oczekiwań klienta. Baza danych realizowana była w oparciu o wolnodostępny system zarządzania relacyjnymi bazami danych MySQL. Aplikacja webowa oparta była po stronie serwerowej o język PHP, natomiast od strony interfejsu użytkownika o stronę internetową napisaną w języku HTML. W celu zapewnienia najwyższej jakości usług, system zaprojektowano z myślą o zabezpieczeniach przed nieautoryzowanym dostępem a także walidacją niepoprawnych danych zarówno po stronie bazy danych jak i aplikacji dostępowej.

### **Implementacja systemu**

System zrealizowano z założeniem, aby jak największa część przetwarzania informacji odbywała się po stronie bazy danych. Do weryfikacji wprowadzanych danych wykorzystane zostały triggery, dzięki którym odbywała się walidacja za pomocą wyrażeń regularnych w chwili modyfikacji rekordów w tabelach, co pozwoliło na uniknięcie niespójności danych. Dodatkowo wy-

korzystane zostały procedury oraz funkcje, umożliwiające realizowanie bardziej złożonych zapytań przy pomocy uproszczonych zapytań. W celu zestawienia większej ilości danych użytkownikowi wykorzystany został widok, którego zadaniem była agregacja danych z wielu tabel i przedstawienie ich w zrozumiałej i czytelnej formie. W związku z przeniesieniem dużej ilości funkcjonalności związanych z przetwarzaniem informacji na stronę bazy danych, udało się odciążyć stronę serwerową oraz zapewnić wyższą wydajność interfejsu dostępowego.

Aplikacja webowa stworzona została z myślą o ułatwieniu użytkownikowi dostępu do danych, odciążeniu go z potrzeby znajomości języka SQL a także umożliwienia łatwego i wygodnego zestawiania i modyfikowania danych dotyczących zamówień produktów. Do zadań aplikacji można zaliczyć odbieranie zapytań od użytkownika, kierowanie ich do systemu bazodanowego oraz zwracanie danych w przejrzystej formie. Dodatkowo aplikacja wspierała dwuwarstwową walidację danych, zarówno ze strony serwerowej w języku PHP jak i strony internetowej HTML. Stan kierowanych zapytań wyświetlany był w formie komunikatów, gdzie wszelkie błędy związane z niepoprawnością danych były natychmiast zgłaszane użytkownikowi.

### **Testowanie systemu**

Zadaniem przeprowadzonych testów było zweryfikowanie poprawności działania systemu a także ułatwienie dalszego rozwoju aplikacji, poprzez wczesne wykrycie błędów w dotychczas zaimplementowanych funkcjonalnościach. Docelowy produkt przetestowany został na wielu poziomach abstrakcji, zarówno po stronie samej bazy danych jak i interfejsu. W celu sprawdzenia poprawności działania mechanizmów ochronnych bazy danych, została ona odłączona od reszty systemu a następnie poddana próbom ręcznego wprowadzenia niepoprawnych danych. Dodatkowo dzięki wykorzystaniu biblioteki specjalistycznych narzędzi, sprawdzono poprawność działania aplikacji dostępowej, poprzez automatyzację testów sterujących zapytaniami do bazy danych, symulujących interakcję pomiędzy interfejsem a użytkownikiem a także symulację zachowania bazy danych w celu porównania otrzymywanych danych. Poza testami funkcjonalnymi obrazującymi praktyczne działanie systemu, opracowana została również kompleksowa dokumentacja techniczna, pozwalająca administratorowi na zestawienie środowiska, uruchomienie produktu a także na modyfikowanie danych jak i nadzorowanie pracy systemu.

## Literatura

- [1] M. Lis, *PHP i MySQL. Dla każdego*, 2nd ed. Helion, Dec. 2012.
- [2] A. Kierzkowski, *PHP5. Tworzenie stron WWW. Ćwiczenia praktyczne*, 2nd ed. Helion, Oct. 2008.