Dokumentacja Projektu: MotoShop - Sklep Internetowy

Autorzy:

Maciej Rodzinka

Bartłomiej Rogóz

Klasa 3AP

Spis Treści

1. Wstęp

1.1. Cel projektu

Celem projektu było stworzenie w pełni funkcjonalnego sklepu internetowego dla warsztatu motocyklowego "Rodzinka & Rogóz". Aplikacja ma umożliwiać sprzedaż części motocyklowych, akcesoriów, odzieży oraz motocykli używanych, a także umożliwiać rezerwację usług serwisowych. System posiada rozbudowany panel administracyjny do zarządzania asortymentem, zamówieniami i użytkownikami.

1.2. Zakres projektu

Projekt obejmuje:

- Stronę główną z prezentacją oferty
- Katalog produktów z filtrowaniem i wyszukiwaniem
- Szczegółowe strony produktów z możliwością dodania recenzji
- System koszyka zakupowego
- Proces składania zamówienia i płatności
- Panel konta użytkownika z historią zamówień
- System rezerwacji usług serwisowych
- Panel administracyjny do zarządzania sklepem
- System zarządzania rolami użytkowników
- Moduł motocykli używanych z możliwością rezerwacji oględzin

1.3. Użytkownicy systemu

System obsługuje następujące typy użytkowników:

- **Klienci niezarejestrowani** - mogą przeglądać produkty, dodawać je do koszyka i składać zamówienia
- **Klienci zarejestrowani** - dodatkowo mogą zapisywać swoje dane, przeglądać historię zamówień, dodawać recenzje
- **Administratorzy** - mają pełny dostęp do panelu administracyjnego
- **Mechanicy** - mają dostęp do modułu zarządzania serwisem
- **Właściciele** - mają dostęp do wszystkich funkcji systemu wraz z modułem analityki

2. Opis ogólny

2.1. Przeznaczenie witryny

Witryna internetowa służy jako platforma sprzedażowa dla warsztatu motocyklowego, umożliwiająca:

- Prezentację oferty części i akcesoriów motocyklowych
- Sprzedaż produktów przez internet
- Rezerwację usług serwisowych
- Prezentację i sprzedaż motocykli używanych
- Budowanie relacji z klientami poprzez system recenzji i opinii

2.2. Główne funkcjonalności

Główne funkcjonalności systemu obejmują:

- Przeglądanie i wyszukiwanie produktów
- Zarządzanie kontem użytkownika
- Proces zakupowy (koszyk, checkout, płatności)
- System recenzji produktów
- Zarządzanie zamówieniami
- Rezerwacja usług serwisowych
- Zarządzanie motocyklami używanymi
- Panel administracyjny z różnymi poziomami dostępu

2.3. Wymagania techniczne

Do uruchomienia projektu wymagane są:

- Serwer z PHP 7.4 lub nowszym
- Serwer MySQL 5.7 lub nowszy
- XAMPP (lub alternatywne środowisko z Apache, MySQL, PHP)
- Nowoczesna przeglądarka internetowa (Chrome, Firefox, Safari, Edge)
- Połączenie z internetem dla załadowania bibliotek zewnętrznych

2.4. Struktura katalogów i plików projektu

```
/Sklep_mech_motocykl_Rodzinka_Rogoz/
|
├── admin/                    # Panel administracyjny
|   ├── includes/             # Pliki dołączane w panelu admin
|   ├── index.php             # Dashboard admina
|   ├── users.php             # Zarządzanie użytkownikami
|   ├── administrators.php    # Zarządzanie administratorami
|   ├── orders.php            # Zarządzanie zamówieniami
|   ├── reviews.php           # Zarządzanie recenzjami
|   ├── settings.php          # Ustawienia systemu
|   └── ...
|
├── assets/                   # Zasoby statyczne
|   ├── css/                  # Pliki CSS
|   ├── js/                   # Pliki JavaScript
|   └── images/               # Obrazy
|
├── database/                 # Pliki SQL do inicjalizacji bazy
|   ├── motoshop_db.sql       # Główny skrypt bazy danych
|   └── add_reviews_table.sql # Dodatkowe skrypty
|
├── includes/                 # Pliki dołączane
|   ├── config.php            # Konfiguracja połączenia z bazą
|   ├── header.php            # Nagłówek strony
|   ├── footer.php            # Stopka strony
|   └── settings_helper.php   # Helper dla ustawień
|
├── index.php                 # Strona główna
├── ProductCatalog.php        # Katalog produktów
├── product.php               # Strona produktu
├── cart.php                  # Koszyk zakupowy
├── cart-actions.php          # Akcje koszyka (AJAX)
├── checkout.php              # Strona zamówienia
├── login.php                 # Logowanie
├── register.php              # Rejestracja
├── account.php               # Panel konta użytkownika
└── ...
```

3. Struktura strony internetowej

3.1. Strona główna (index.php)

Strona główna sklepu prezentuje:

- Banner z promocjami
- Wybrane kategorie produktów
- Polecane produkty
- Nowości w ofercie
- Sekcję usług serwisowych
- Sekcję motocykli używanych

3.2. Katalog produktów (ProductCatalog.php)

Katalog produktów umożliwia:

- Przeglądanie produktów z podziałem na kategorie
- Filtrowanie po atrybutach (cena, marka, itp.)
- Sortowanie wyników
- Wyszukiwanie produktów
- Paginację wyników

3.3. Strona produktu (product.php)

Strona produktu zawiera:

- Galerie zdjęć produktu
- Szczegółowy opis
- Specyfikację techniczną
- System recenzji i ocen
- Możliwość dodania produktu do koszyka
- Wybór ilości
- Informację o dostępności
- Polecane produkty

3.4. Koszyk zakupowy (cart.php)

Koszyk zakupowy umożliwia:

- Przeglądanie dodanych produktów

- Aktualizację ilości
- Usuwanie produktów
- Obliczanie wartości zamówienia
- Przejście do płatności

3.5. Proces zakupowy (checkout.php, order-confirmation.php)

Proces zakupowy obejmuje:

- Formularz danych osobowych i adresowych
- Wybór metody dostawy
- Wybór metody płatności
- Podsumowanie zamówienia
- Potwierdzenie zamówienia z instrukcjami płatności

3.6. Panel konta użytkownika (account.php)

Panel konta użytkownika zawiera:

- Zarządzanie danymi osobowymi
- Historię zamówień
- Możliwość zmiany hasła
- Zapisane adresy dostawy

3.7. System logowania i rejestracji (login.php, register.php)

System logowania i rejestracji obejmuje:

- Formularz logowania
- Formularz rejestracji
- Odzyskiwanie hasła
- Weryfikację danych

3.8. Panel administracyjny (admin/*.php)

Panel administracyjny zawiera:

- Dashboard z podsumowaniem
- Zarządzanie produktami
- Zarządzanie kategoriami
- Zarządzanie zamówieniami
- Zarządzanie użytkownikami
- Zarządzanie recenzjami
- Zarządzanie usługami serwisowymi
- Zarządzanie motocyklami używanymi
- Ustawienia systemu

4. Elementy statyczne

4.1. HTML – struktura dokumentów

Witryna wykorzystuje semantyczny HTML5 z odpowiednimi tagami dla poszczególnych elementów strony:

- `<header>` dla nagłówka strony
- `<nav>` dla nawigacji
- `<main>` dla głównej zawartości
- `<aside>` dla elementów pobocznych
- `<footer>` dla stopki
- `<section>` i `<article>` dla organizacji treści

4.2. CSS – stylizacja (TailwindCSS)

Projekt wykorzystuje framework TailwindCSS do stylizacji:

- Klasy utility dla szybkiego stylowania
- Responsywny design (mobile-first)
- System kolorów i typografii
- Customowe komponenty
- Animacje i efekty przejścia

4.3. Biblioteki zewnętrzne

W projekcie wykorzystano następujące biblioteki:

- **Remixicon** - zestaw ikon
- **Alpine.js** - lekki framework JavaScript dla interaktywności
- **Swiper.js** - biblioteka do tworzenia sliderów
- **Tailwind CSS** - framework CSS

5. JavaScript

5.1. Interakcje na stronie produktu

JavaScript na stronie produktu obsługuje:

- Galerię zdjęć produktu
- Wybór ilości produktu
- System ocen gwiazdkowych
- Przełączanie między zakładkami (opis, specyfikacja, recenzje)

5.2. Dynamiczny koszyk zakupowy

System koszyka wykorzystuje AJAX do:

- Dodawania produktów bez przeładowania strony
- Aktualizacji ilości produktów
- Usuwania produktów
- Obliczania sum częściowych i całkowitych

5.3. Walidacja formularzy

JavaScript waliduje formularze:

- Dane osobowe i adresowe
- Formularze logowania i rejestracji
- Formularze recenzji
- Formularze rezerwacji usług

5.4. System recenzji produktów

System recenzji wykorzystuje JavaScript do:

- Interaktywnego systemu ocen gwiazdkowych
- Walidacji formularza recenzji
- Dynamicznego wyświetlania nowych recenzji

6. Elementy dynamiczne (PHP)

6.1. Obsługa logowania i rejestracji użytkownika

System autoryzacji obejmuje:

- Rejestrację nowych użytkowników
- Logowanie istniejących użytkowników
- Zarządzanie sesją użytkownika
- Odzyskiwanie hasła
- Walidację danych

6.2. Pobieranie danych z bazy

Aplikacja wykorzystuje przygotowane zapytania (prepared statements) do:

- Pobierania listy produktów z filtrowaniem
- Pobierania szczegółów produktu
- Pobierania recenzji
- Pobierania danych użytkownika
- Pobierania historii zamówień

6.3. Obsługa koszyka i zamówień

System koszyka i zamówień obejmuje:

- Dodawanie, aktualizację i usuwanie produktów z koszyka
- Zapisywanie koszyka w sesji lub bazie danych
- Przetwarzanie zamówień
- Generowanie numerów zamówień
- Aktualizację stanów magazynowych

6.4. Panel administracyjny

Panel administracyjny oferuje:

- Zarządzanie produktami (CRUD)
- Zarządzanie kategoriami (CRUD)
- Zarządzanie zamówieniami
- Zarządzanie użytkownikami
- Moderację recenzji
- Konfigurację systemu

6.5. System zarządzania rolami użytkowników

System ról obejmuje:

- Role użytkowników (user, admin, mechanic, owner)
- Kontrolę dostępu do funkcji administracyjnych
- Przydzielanie i odbieranie uprawnień

7. Baza danych

7.1. Projekt struktury bazy danych

Baza danych składa się z następujących głównych elementów:

- Tabele przechowujące dane produktów i kategorii
- Tabele użytkowników i ich ról
- Tabele zamówień i ich elementów
- Tabele recenzji produktów
- Tabele konfiguracji systemu

7.2. Opis tabel i pól

Tabela users:

- `id` (INT, PRIMARY KEY) - identyfikator użytkownika
- `first_name` (VARCHAR) - imię
- `last_name` (VARCHAR) - nazwisko
- `email` (VARCHAR, UNIQUE) - adres email

- `password` (VARCHAR) - hashowane hasło
- `role` (ENUM) - rola użytkownika
- `created_at` (TIMESTAMP) - data utworzenia
- `updated_at` (TIMESTAMP) - data aktualizacji

Tabela products:

- `id` (INT, PRIMARY KEY) - identyfikator produktu
- `name` (VARCHAR) - nazwa produktu
- `slug` (VARCHAR) - URL-friendly nazwa
- `description` (TEXT) - opis produktu
- `price` (DECIMAL) - cena regularna
- `sale_price` (DECIMAL) - cena promocyjna
- `stock` (INT) - stan magazynowy
- `category_id` (INT, FOREIGN KEY) - kategoria
- `brand_id` (INT, FOREIGN KEY) - marka
- `status` (ENUM) - status produktu

Tabela orders:

- `id` (INT, PRIMARY KEY) - identyfikator zamówienia
- `user_id` (INT, FOREIGN KEY) - identyfikator użytkownika
- `order_number` (VARCHAR) - numer zamówienia
- `status` (ENUM) - status zamówienia
- `total_amount` (DECIMAL) - wartość zamówienia
- `payment_method` (ENUM) - metoda płatności
- `payment_status` (ENUM) - status płatności
- `created_at` (TIMESTAMP) - data utworzenia

Tabela product_reviews:

- `id` (INT, PRIMARY KEY) - identyfikator recenzji
- `product_id` (INT, FOREIGN KEY) - identyfikator produktu
- `user_id` (INT, FOREIGN KEY) - identyfikator użytkownika
- `rating` (TINYINT) - ocena (1-5)
- `title` (VARCHAR) - tytuł recenzji
- `content` (TEXT) - treść recenzji
- `status` (ENUM) - status recenzji

Tabela shop_settings:

- `id` (INT, PRIMARY KEY) - identyfikator ustawienia
- `setting_key` (VARCHAR) - klucz ustawienia
- `setting_value` (TEXT) - wartość ustawienia
- `setting_group` (VARCHAR) - grupa ustawień
- `is_public` (TINYINT) - czy publiczne

7.3. Relacje między tabelami

Główne relacje między tabelami to:

- `products` → `categories` (wiele do jednego)
- `products` → `brands` (wiele do jednego)
- `order_items` → `products` (wiele do jednego)
- `order_items` → `orders` (wiele do jednego)
- `orders` → `users` (wiele do jednego)
- `product_reviews` → `products` (wiele do jednego)
- `product_reviews` → `users` (wiele do jednego)

7.4. Przykładowe zapytania SQL

Pobieranie produktów z filtrowaniem:

```
SELECT p.*, c.name AS category_name, b.name AS brand_name
FROM products p
LEFT JOIN categories c ON p.category_id = c.id
LEFT JOIN brands b ON p.brand_id = b.id
WHERE p.status = 'published'
AND (p.name LIKE ? OR p.description LIKE ?)
AND p.category_id = ?
ORDER BY p.created_at DESC
LIMIT ?, ?
```

Podsumowanie zamówień:

```
SELECT
    COUNT(*) AS total_orders,
```

```
    SUM(total_amount) AS total_sales,
    COUNT(DISTINCT user_id) AS unique_customers
FROM orders
WHERE created_at BETWEEN ? AND ?
AND status != 'cancelled'
```

Średnia ocena produktów:

```
SELECT
    p.id,
    p.name,
    AVG(r.rating) AS avg_rating,
    COUNT(r.id) AS review_count
FROM products p
LEFT JOIN product_reviews r ON p.id = r.product_id
WHERE r.status = 'approved'
GROUP BY p.id, p.name
ORDER BY avg_rating DESC
```

8. Bezpieczeństwo

8.1. Walidacja danych wejściowych

Wszystkie dane wejściowe są walidowane za pomocą:

- Funkcji sanityzujących (sanitize())
- Typowania i castowania danych
- Walidacji formularzy (zarówno po stronie klienta jak i serwera)
- Ograniczania długości pól

8.2. Obsługa sesji i uwierzytelnianie

System uwierzytelniania obejmuje:

- Bezpieczne przechowywanie haseł (password_hash)

- Weryfikację haseł (password_verify)
- Zarządzanie sesją użytkownika
- Ochronę przed atakami typu session hijacking

8.3. Zabezpieczenia przed atakami typu SQL Injection

Zabezpieczenia przed SQL Injection obejmują:

- Używanie prepared statements
- Parametryzację zapytań SQL
- Walidację i sanityzację danych wejściowych
- Unikanie dynamicznego generowania zapytań SQL

8.4. Zarządzanie uprawnieniami użytkowników

System uprawnień obejmuje:

- Kontrolę dostępu bazującą na rolach
- Weryfikację uprawnień przed wykonaniem akcji
- Ograniczanie dostępu do funkcji administracyjnych
- Logowanie akcji administracyjnych

9. Możliwości rozbudowy

9.1. Integracja z systemami płatności

Możliwe rozszerzenia systemu płatności:

- Integracja z PayU
- Integracja z PayPal
- Integracja z BLIK
- Obsługa płatności kartami kredytowymi

9.2. System powiadomień e-mail

Rozbudowa systemu powiadomień:

- Powiadomienia o statusie zamówień
- Powiadomienia o nowych recenzjach
- Newsletter
- Powiadomienia o promocjach

9.3. Aplikacja mobilna

Rozwój aplikacji mobilnej:

- Wersja na Android i iOS
- Powiadomienia push
- Skanowanie kodów QR
- Wersja offline katalogu

9.4. Analityka i raportowanie

Rozbudowa modułu analityki:

- Raporty sprzedażowe
- Analiza zachowań użytkowników
- Analiza skuteczności promocji
- Prognozowanie sprzedaży

10. Załączniki

10.1. Fragmenty kodu źródłowego

Funkcja obsługi dodawania do koszyka (cart-actions.php):

```
function addToCart() {
    global $conn;

    $product_id = isset($_POST['product_id']) ? (int)$_POST['product_id'] : (
    $quantity = isset($_POST['quantity']) ? (int)$_POST['quantity'] : 1;

    if ($product_id <= 0) {
```

```
        echo json_encode(['success' => false, 'message' => 'Nieprawidłowy ide
        exit;
    }

    if ($quantity <= 0) {
        $quantity = 1;
    }

    // Sprawdzenie czy produkt istnieje i jest dostępny
    $product_query = "SELECT id, name, price, sale_price, stock FROM products
    $stmt = $conn->prepare($product_query);
    $stmt->bind_param("i", $product_id);
    $stmt->execute();
    $result = $stmt->get_result();

    if (!$result || $result->num_rows === 0) {
        echo json_encode(['success' => false, 'message' => 'Produkt nie istni
        exit;
    }

    $product = $result->fetch_assoc();

    // Sprawdzenie stanu magazynowego
    if ($product['stock'] < $quantity) {
        echo json_encode(['success' => false, 'message' => 'Niewystarczająca
        exit;
    }

    // Dodanie do koszyka
    $cart_id = getOrCreateCart();

    // Sprawdzenie czy produkt już jest w koszyku
    $check_query = "SELECT id, quantity FROM cart_items WHERE cart_id = ? AND
    $check_stmt = $conn->prepare($check_query);
    $check_stmt->bind_param("ii", $cart_id, $product_id);
    $check_stmt->execute();
    $check_result = $check_stmt->get_result();

    if ($check_result && $check_result->num_rows > 0) {
        // Aktualizacja ilości
        $cart_item = $check_result->fetch_assoc();
        $new_quantity = $cart_item['quantity'] + $quantity;

        if ($new_quantity > $product['stock']) {
            $new_quantity = $product['stock'];
```

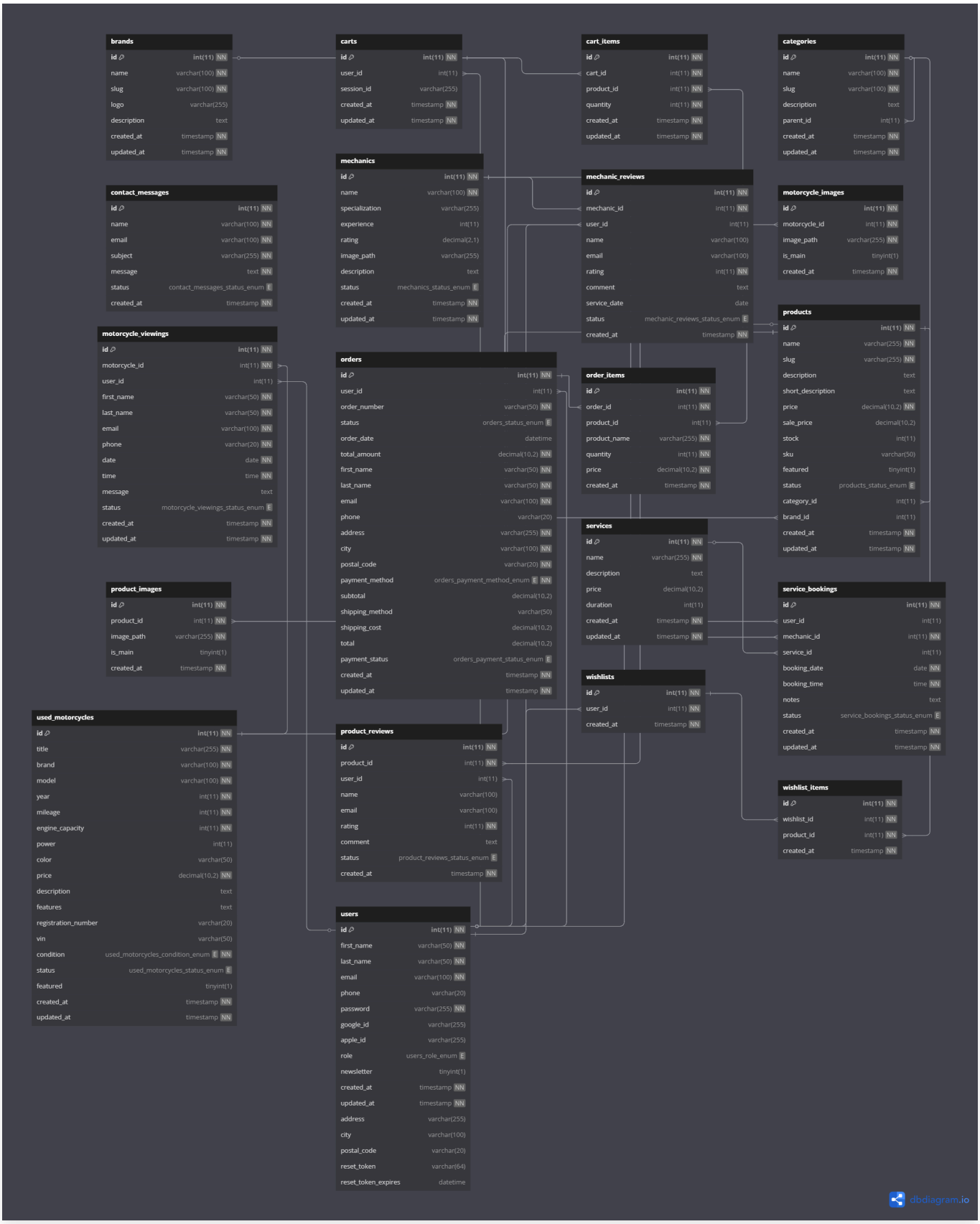
```
        }

        $update_query = "UPDATE cart_items SET quantity = ? WHERE id = ?";
        $update_stmt = $conn->prepare($update_query);
        $update_stmt->bind_param("ii", $new_quantity, $cart_item['id']);

        if ($update_stmt->execute()) {
            echo json_encode([
                'success' => true,
                'message' => 'Zaktualizowano ilość produktu w koszyku',
                'cart_count' => getCartItemCount(),
                'product_name' => $product['name']
            ]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Błąd podczas
        }
    } else {
        // Dodanie nowego produktu do koszyka
        $insert_query = "INSERT INTO cart_items (cart_id, product_id, quanti
        $insert_stmt = $conn->prepare($insert_query);
        $insert_stmt->bind_param("iii", $cart_id, $product_id, $quantity);

        if ($insert_stmt->execute()) {
            echo json_encode([
                'success' => true,
                'message' => 'Produkt został dodany do koszyka',
                'cart_count' => getCartItemCount(),
                'product_name' => $product['name']
            ]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Błąd podczas
        }
    }

    exit;
}
```

System zarządzania ustawieniami sklepu (settings_helper.php):

```
/**
 * Pobiera wartość ustawienia z bazy danych
```

```
 *
 * @param string $key Klucz ustawienia
 * @param mixed $default Domyślna wartość jeśli ustawienie nie istnieje
 * @param bool $force_refresh Czy wymusić odświeżenie z bazy danych
 * @return mixed Wartość ustawienia
 */
function get_setting($key, $default = null, $force_refresh = false) {
    global $conn;
    static $settings_cache = [];

    // Jeśli wartość jest w cache i nie wymuszamy odświeżenia
    if (!$force_refresh && isset($settings_cache[$key])) {
        return $settings_cache[$key];
    }

    $query = "SELECT setting_value FROM shop_settings WHERE setting_key = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("s", $key);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result && $result->num_rows > 0) {
        $row = $result->fetch_assoc();
        $settings_cache[$key] = $row['setting_value'];
        return $row['setting_value'];
    }

    // Jeśli ustawienie nie istnieje, zwracamy wartość domyślną
    return $default;
}
```

10.3. Diagram bazy danych



10.4. Lista błędów/uwag i pomysłów na ulepszenia

Znane błędy:

1. Nieprawidłowe wyświetlanie niektórych znaków diakrytycznych w mailu z potwierdzeniem zamówienia
2. Brak automatycznego odświeżania statusu zamówienia
3. Problem z wyświetlaniem zdjęć na niektórych urządzeniach mobilnych
4. Nieprawidłowe działanie filtrów w katalogu na starszych przeglądarkach
5. Błąd w kalkulacji ceny przy dużej ilości produktów z rabatem

Pomysły na ulepszenia:

1. Integracja z systemami płatności online (PayU, PayPal)
2. Wdrożenie systemu punktów lojalnościowych
3. Dodanie zaawansowanego modułu wyszukiwania części wg modelu motocykla
4. Implementacja chatbota dla obsługi klienta
5. Integracja z mediami społecznościowymi
6. Rozbudowa systemu powiadomień email
7. Dodanie wersji PWA aplikacji
8. Integracja z systemem magazynowym
9. Wdrożenie systemu śledzenia przesyłek
10. Dodanie konfiguratora części kompatybilnych z modelem motocykla

---

Dokumentacja Projektu MotoShop

Autorzy: Maciej Rodzinka, Bartłomiej Rogóz

Klasa 3AP