Restauracja

Mikołaj Misiarz i Weronika Drega

1.Wstęp

1.1 Cel Projektu

Głównym celem projektu jest stworzenie nowoczesnej, responsywnej i funkcjonalnej strony internetowej dla restauracji która:

- Prezentuje ofertę gastronomiczną (menu, promocje, sezonowe dania).
- Umożliwia klientom rezerwację stolików online.
- Zapewnia informacje kontaktowe i lokalizację na mapie.
- Pozwala na zamówienie dań na wynos/dostawę (opcjonalnie).
- Buduje wizerunek marki poprzez atrakcyjny design i treści (galeria, historia restauracji).

1.2 Zakres projektu

Strona restauracji obejmuje:

1. **Stronę główną** z:
    a. Sekcją hero z przyciskiem "Zamów Online" (wymaga logowania)
    b. Opisem restauracji, promocjami, galerią zdjęć z powiększeniem
    c. Listą opinii (MySQL) i formularzem kontaktowym
2. **System logowania** z:
    a. Autoryzacją ról (user/admin/właściciel)
    b. Funkcją "zapamiętaj mnie" (cookies)
3. **Podstrony** do:
    a. Rezerwacji stolików
    b. Składania zamówień
    c. Panelu administracyjnego
4. **Technologie**: PHP, MySQL, HTML/CSS, JavaScript

1.3 Użytkownicy

- **Goście restauracji** (klienci):
    - Przeglądanie menu i promocji.
    - Składanie rezerwacji/zamówień online.
    - Kontakt z restauracją.
- **Personel restauracji:**
    - Obsługa rezerwacji i zamówień.
    - Aktualizacja treści (menu, godziny otwarcia).
- **Administrator strony:**
    - Zarządzanie treścią, użytkownikami i integracjami.

2.Opis ogólny

2.1 Przeznaczenie witryny

Witryna **Restauracja Kraftowa** służy jako platforma internetowa umożliwiająca:

- Prezentację menu, promocji oraz galerii dań.
- Rezerwację stolików i zamawianie jedzenia online.
- Kontakt z restauracją oraz wyrażanie opinii przez klientów.
- Zarządzanie treścią przez personel (panel administracyjny).

Głównym celem jest zwiększenie wygody klientów oraz promocja usług restauracji.

2.2 Główne funkcjonalnosci

1. **Dla gości:**
    a. Przeglądanie menu, promocji i galerii.
    b. Rezerwacja stolików (`rezerwacja.php`).
    c. Formularz kontaktowy i mapa lokalizacji.
    d. Przeglądanie opinii innych klientów.
2. **Dla zalogowanych użytkowników:**
    a. Składanie zamówień online (`zamowienie.php`).
    b. Dodawanie opinii (`dodaj_opinie.php`).
3. **Dla personelu (admin/właściciel):**
    a. Panel administracyjny (`admin_dashboard.php`).
    b. Zarządzanie treścią (menu, promocje, opinie).
4. **System logowania:**
    a. Rejestracja/logowanie z rolami (user/admin/właściciel).
    b. Opcja "zapamiętaj mnie" (cookies).

2.3 Wymagania Techniczne

- **Serwer lokalny:** XAMPP (Apache, MySQL, PHP).
- **Baza danych:** MySQL (tabele: `uzytkownicy`, `opinie`).
- **Przeglądarki:** Chrome, Firefox, Edge (wersje najnowsze).
- **Wersje technologii:**
    - PHP ≥ 7.4 (obsługa sesji, PDO).
    - MySQL ≥ 5.7.
    - JavaScript (ES6).

3.Struktura strony internetowej

3.1 Strona Głowna (index.php)

- **Cel:** Wizytówka restauracji z szybkim dostępem do głównych funkcji.
- **Zawartość:**
    - Nagłówek z nawigacją (menu, rezerwacje, logowanie).
    - Sekcja hero z przyciskiem "Zamów Online".
    - Sekcje: *O nas*, *Promocje*, *Galeria*, *Opinie*, *Kontakt*.
    - Stopka z danymi kontaktowymi i mediami społecznościowymi.

3.2 Menu (menu.php)

- **Cel:** Prezentacja pełnej oferty kulinarnej.
- **Zawartość:**
    - Podział na kategorie (przystawki, dania główne, desery, napoje).
    - Opisy dań, ceny, zdjęcia (opcjonalnie filtry np. wegetariańskie).

3.3 Rezerwacje (rezerwacje.php)

- **Cel:** Formularz rezerwacji stolików.
- **Zawartość:**
    - Wybór daty, godziny, liczby osób.
    - Dane kontaktowe (dla niezalogowanych).
    - Potwierdzenie rezerwacji mailem/SMS (opcjonalnie).

3.4 Zamówienia online (zamowienia.php)

- **Cel:** Składanie zamówień na wynos/dostawę.

3.5. Panel logowania i administracji

1. **Logowanie/rejestracja (`login.php`)**
    a. Formularz logowania z opcją "zapamiętaj mnie".
    b. Link do rejestracji (dla nowych użytkowników).
    c. Przekierowanie po zalogowaniu (np. do `zamowienie.php`).
2. **Panel administracyjny (`admin.php` lub `admin_dashboard.php`)**
    a. **Dla admina:**
        i. Zarządzanie menu, promocjami, opiniami.
        ii. Podgląd rezerwacji/zamówień.

b. **Dla właściciela:**
    i. Statystyki sprzedaży, zarządzanie pracownikami.

4.Elementy statyczne

4.1 HTML – struktura dokumentow

Wszystkie strony wykorzystują **spójną strukturę HTML5** z następującymi sekcjami:

- **Nagłówek (<head>):**

```
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Logowanie</title>
<link rel="stylesheet" href="login.css" />
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
```

- **Część główna (<body>)**:
- Nawigacja (<nav>) z linkami do podstron.
- Sekcje treściowe (`<section>`) dla każdej funkcjonalności.
- Stopka (`<footer>`) z danymi kontaktowymi.

- **Wspólne elementy**:
- Modal logowania (`loginModal`).
- Formularze (kontakt, opinie, zamówienia).
- Galeria zdjęć z funkcją powiększania (popup).

4.2 CSS – stylizacja (style.css i inne pliki CSS)

Główne cechy stylizacji

1. **Spójna kolorystyka** (głównie odcienie czerwieni #b52d22 i neutralne tła):
    a. Dominujące kolory: #b52d22 (czerwień kraftowa), #ff5722 (pomarańcz), #2c2c2c (ciemne tło stopki).
    b. Tła sekcji: #fff8f2 (kremowy), #f5f5f5 (jasnoszary).
2. **Responsywność**:
    a. Media queries (np. w `style.css` dla mobilnych wersji galerii i nawigacji).
    b. Flexbox/grid do układania elementów (np. `.gallery`, `.footer-content`).
3. **Animacje i efekty hover**:
    a. Przejścia (`transition`) dla przycisków, kart promocji, zdjęć w galerii.

b. Skalowanie (`transform: scale()`) przy najechaniu na zdjęcia lub promocje.

4.3 Biblioteki zewnętrzne

1.Mapy Google (iframe)5.Java script

5.1 Obsługa formularza

```
<script>
document.getElementById('data').addEventListener('change', function () {
    const data = this.value;
    fetch('godziny.php?data=' + data)
        .then(res => res.json())
        .then(godziny => {
            const kontener = document.getElementById('godziny');
            kontener.innerHTML = '<h3>Dostępne godziny:</h3>';
            godziny.forEach(item => {
                const blok = document.createElement('div');
                blok.className = 'godzina';
                if (item.status === 'zajeta') {
                    blok.classList.add('niedostepna');
                    blok.textContent = item.godzina;
                } else {
                    blok.textContent = item.godzina;
                    blok.style.cursor = 'pointer';
                    blok.onclick = function () {
                        document.querySelectorAll('.godzina').forEach(el => el.classList.remove('wybrana'));
                        blok.classList.add('wybrana');
                        document.getElementById('wybranaGodzina').value = item.godzina;
                    };
                }
                kontener.appendChild(blok);
            });
        });
});

// Obsługa telefonu - tak jak wcześniej

const telefonInput = document.getElementById('telefon');

telefonInput.addEventListener('focus', function () {
    if (!this.value.startsWith('+48-')) {
        this.value = '+48-';
        this.setSelectionRange(this.value.length, this.value.length);
    }
});

telefonInput.addEventListener('input', function () {
    let cursorPos = this.selectionStart;
    let originalLength = this.value.length;

    let digits = this.value.replace(/\D/g, '');

    if (digits.startsWith('48')) {
        digits = digits.substring(2);
    }

    digits = digits.substring(0, 9);

    let formatted = '+48-';

    if (digits.length > 6) {
        formatted += digits.substring(0, 3) + '-' + digits.substring(3, 6) + '-' + digits.substring(6);
    } else if (digits.length > 3) {
        formatted += digits.substring(0, 3) + '-' + digits.substring(3);
    } else {
        formatted += digits;
    }

    this.value = formatted;

    let newLength = this.value.length;
    cursorPos = cursorPos + (newLength - originalLength);
    this.setSelectionRange(cursorPos, cursorPos);
});

// Dodatkowo możesz zablokować wysłanie formularza, jeśli nie wybrano godziny

document.getElementById('rezerwacjaForm').addEventListener('submit', function(e) {
    const godzinaVal = document.getElementById('wybranaGodzina').value;
    if (!godzinaVal) {
        alert('Proszę wybrać godzinę rezerwacji.');
        e.preventDefault();
    }
});

  const thumbnail = document.getElementById("thumbnail");
  const lightbox = document.getElementById("lightbox");
  const lightboxImg = document.getElementById("lightbox-img");
  const closeBtn = document.getElementById("closeBtn");

  thumbnail.addEventListener("click", () => {
    lightbox.style.display = "block";
    lightboxImg.src = thumbnail.src;
  });

  closeBtn.addEventListener("click", () => {
    lightbox.style.display = "none";
  });

  window.addEventListener("click", (event) => {
    if (event.target === lightbox) {
      lightbox.style.display = "none";
    }
  });
```

6. Elementy dynamiczne (PHP)

6.1 Obsługa logowania i rejestracji uzytkownika

Plik: login.php i rejestracja.php

```php
<?php
session_start();
require_once 'db_config.php';

if (isset($_SESSION['user_id'])) {
    header('Location: index.php');
    exit;
}

$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    try {
        $stmt = $pdo->prepare("SELECT id, imie, nazwisko, haslo, uprawnienia FROM uzytkownicy WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if ($user && $password === $user['haslo']) { // Porównanie w czystym tekście
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['user_name'] = $user['imie'] . ' ' . $user['nazwisko'];
            $_SESSION['user_role'] = $user['uprawnienia'];

            if (isset($_POST['remember'])) {
                setcookie('remember_user', $user['id'], time() + (30 * 24 * 60 * 60), "/");
            }

            header('Location: index.php');
            exit;
        } else {
            $error = 'Nieprawidłowy email lub hasło';
        }
    } catch (PDOException $e) {
        $error = 'Błąd systemu. Spróbuj ponownie.';
    }
}
?>
```

```php
<?php
$komunikat = ''; // zmienna na komunikat

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $conn = new mysqli('localhost', 'root', '', 'restauracja');
    if ($conn->connect_error) {
        die("Błąd połączenia z bazą danych: " . $conn->connect_error);
    }

    $imie = $conn->real_escape_string($_POST['imie']);
    $nazwisko = $conn->real_escape_string($_POST['nazwisko']);
    $nazwa_uzytkownika = $conn->real_escape_string($_POST['nazwa_uzytkownika']);
    $email = $conn->real_escape_string($_POST['email']);
    $haslo = password_hash($_POST['password'], PASSWORD_DEFAULT);
    $uprawnienia = 'uzytkownik';

    $checkQuery = "SELECT * FROM uzytkownicy WHERE nazwa_uzytkownika = '$nazwa_uzytkownika' OR email = '$email'";
    $result = $conn->query($checkQuery);

    if ($result->num_rows > 0) {
        $komunikat = "Użytkownik o podanej nazwie lub e-mailu już istnieje.";
    } else {
        $sql = "INSERT INTO uzytkownicy (imie, nazwisko, nazwa_uzytkownika, email, haslo, uprawnienia)
                VALUES ('$imie', '$nazwisko', '$nazwa_uzytkownika', '$email', '$haslo', '$uprawnienia')";

        if ($conn->query($sql) === TRUE) {
            $komunikat = "Rejestracja zakończona sukcesem!";
        } else {
            $komunikat = "Błąd podczas rejestracji: " . $conn->error;
        }
    }

    $conn->close();
}
?>
```

6.2 Pobieranie danych z bazy

Plik: admin_dashboard.php

```php
<?php foreach ($reservations as $reservation): ?>
<tr>
    <td><?= htmlspecialchars($reservation['id']) ?></td>
    <td><?= htmlspecialchars($reservation['imie']) ?></td>
    <td><?= htmlspecialchars($reservation['nazwisko']) ?></td>
    <td><?= htmlspecialchars($reservation['data']) ?></td>
    <td><?= htmlspecialchars($reservation['godzina']) ?></td>
    <td><?= htmlspecialchars($reservation['ilosc_osob']) ?></td>
    <td>
        <form method="POST" style="display:inline;">
            <input type="hidden" name="reservation_id" value="<?= $reservation['id'] ?>">
            <button type="submit" name="delete_reservation" class="btn">Usuń</button>
        </form>
    </td>
</tr>
<?php endforeach; ?>
```

6.3 Obsługa formularza kontaktowego

Plik: kontakt.php

```php
<?php
session_start();
require_once 'db_config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Sprawdzenie czy użytkownik jest zalogowany
    if (!isset($_SESSION['user_id'])) {
        // Przekierowanie do logowania, z powrotem do kontakt.php po zalogowaniu
        header("Location: login.php?redirect=kontakt.php");
        exit;
    }

    $name = trim($_POST['name'] ?? '');
    $email = trim($_POST['email'] ?? '');
    $message = trim($_POST['message'] ?? '');
    $user_id = $_SESSION['user_id'];

    if ($name !== '' && $email !== '' && $message !== '') {
        try {
            $stmt = $pdo->prepare("INSERT INTO pytania (uzytkownik_id, imie, email, tresc, data_dodania) VALUES (?, ?, ?, ?, NOW())");
            $stmt->execute([$user_id, $name, $email, $message]);

            header("Location: index.php?success=1#kontakt");
            exit;
        } catch (PDOException $e) {
            // Możesz dodać logowanie błędu lub wyświetlić komunikat
            header("Location: index.php?success=0#kontakt");
            exit;
        }
    } else {
        header("Location: index.php?success=0#kontakt");
        exit;
    }
} else {
    // Jeśli ktoś otworzy kontakt.php bez POST, można przekierować lub pokazać formularz
    header("Location: index.php#kontakt");
    exit;
}
```

6.4 Panel administracyjny

Plik: admin_dashboard.php

```php
<?php
session_start();
require_once 'db_config.php';


if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit;
}

if ($_SESSION['user_role'] !== 'admin' && $_SESSION['user_role'] !== 'wlasciciel') {
    header('Location: index.php');
    exit;
}


$isOwner = ($_SESSION['user_role'] === 'wlasciciel');


if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if ($isOwner && isset($_POST['delete_user'])) {
        $userId = $_POST['user_id'];
        $stmt = $pdo->prepare("DELETE FROM uzytkownicy WHERE id = ?");
        $stmt->execute([$userId]);
    }


    if (isset($_POST['delete_reservation'])) {
        $reservationId = $_POST['reservation_id'];
        $stmt = $pdo->prepare("DELETE FROM rezerwacje WHERE id = ?");
        $stmt->execute([$reservationId]);
    }


    if (isset($_POST['delete_opinion'])) {
        $opinionId = $_POST['opinion_id'];
        $stmt = $pdo->prepare("DELETE FROM opinie WHERE id = ?");
        $stmt->execute([$opinionId]);
    }


    if (isset($_POST['delete_question'])) {
        $questionId = $_POST['question_id'];
        $stmt = $pdo->prepare("DELETE FROM pytania WHERE id = ?");
        $stmt->execute([$questionId]);
    }


    if ($isOwner && isset($_POST['promote_to_admin'])) {
        $userId = $_POST['user_id'];
        $stmt = $pdo->prepare("UPDATE uzytkownicy SET uprawnienia = 'admin' WHERE id = ?");
        $stmt->execute([$userId]);
    }
}
if (isset($_POST['delete_pracownik'])) {
    $pracownikId = $_POST['pracownik_id'];
    $stmt = $pdo->prepare("DELETE FROM pracownicy WHERE id = ?");
    $stmt->execute([$pracownikId]);
    $_SESSION['success_msg'] = "Pracownik został usunięty";
    header("Location: admin_dashboard.php");
    exit;
}


$users = $pdo->query("SELECT * FROM uzytkownicy")->fetchAll();
$reservations = $pdo->query("SELECT * FROM rezerwacje")->fetchAll();
$opinions = $pdo->query("SELECT * FROM opinie")->fetchAll();
```

7. Baza danych

7.1. Projekt struktury bazy danych

Baza danych `restauracja` składa się z 6 głównych tabel:

1. **`uzytkownicy`** – przechowuje dane użytkowników (klientów i personelu).
2. **`pracownicy`** – informacje o pracownikach restauracji.
3. **`opinie`** – opinie klientów o restauracji.
4. **`pytania`** – pytania od klientów (formularz kontaktowy).
5. **`rezerwacje`** – rezerwacje stolików.
6. **`zamowienia`** – zamówienia online.

7.2. Opis tabel i pól

Tabela uzytkownicy

Pole	Typ	Opis
`id`	INT	Klucz główny
`imie`, `nazwisko`	VARCHAR(50)	Dane osobowe
`nazwa_uzytkownika`	VARCHAR(50)	Unikalny login
`email`	VARCHAR(100)	Unikalny adres email
`haslo`	VARCHAR(200)	Hasło (hashowane)
`uprawnienia`	ENUM('admin','uzytkownik','wlasciciel')	Rola użytkownika
`remember_token`	VARCHAR(64)	Token do "zapamiętaj mnie"
`token_expiry`	DATETIME	Ważność tokenu

Tabela pracownicy

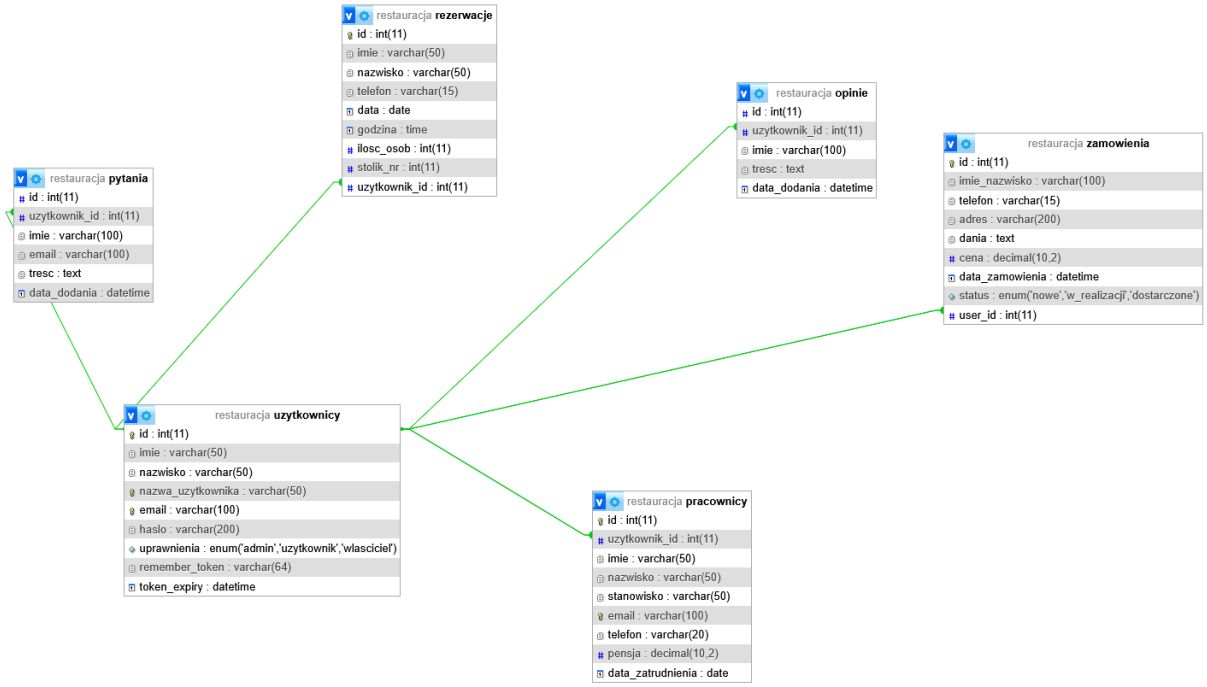
Pole	Typ	Opis
`id`	INT	Klucz główny
`uzytkownik_id`	INT	Powiązanie z tabelą `uzytkownicy` (FK)
`imie`, `nazwisko`	VARCHAR(50)	Dane pracownika
`stanowisko`	VARCHAR(50)	np. "Kelner", "Kucharz"
`pensja`	DECIMAL(10,2)	Wynagrodzenie

`data_zatrudnienia`	DATE	Data rozpoczęcia pracy

Tabela opinie

Pole	Typ	Opis
`id`	INT	Klucz główny
`uzytkownik_id`	INT	Autor opinii (FK do `uzytkownicy`)
`imie`	VARCHAR(100)	Imię autora (dla niezalogowanych)
`tresc`	TEXT	Treść opinii
`data_dodania`	DATETIME	Data dodania

7.3 Relacje między danymi



8. Bezpieczeństwo

8.1 Walidacja danych wejściowych

```
function validatePasswords() {
  const password = document.getElementById('password').value;
  const confirm = document.getElementById('confirm').value;
  const hasUppercase = /[A-Z]/.test(password);
  const hasDigit = /\d/.test(password);

  if (!hasUppercase || !hasDigit) {
    alert('Hasło musi zawierać co najmniej jedną wielką literę i jedną cyfrę.');
    return false;
  }

  if (password !== confirm) {
    alert('Hasła nie są takie same.');
    return false;
  }

  return true;
}
```

9. Możliwości rozbudowy systemu

9.1. Dodanie wyszukiwarki dań i promocji

Funkcjonalność:

- Wyszukiwanie dań po nazwie, składnikach lub kategorii
- Filtrowanie wyników (np. wegetariańskie, ostrość, cena)

9.2. System ocen i komentarzy

Funkcjonalność:

- Oceny w skali 1-5 gwiazdek
- Komentarze do dań/restauracji
- Moderacja treści (dla adminów)

9.3. Rozbudowany system rezerwacji online

Nowe funkcje:

1. **Wybór stolika na mapie sali**
2. **Powiadomienia SMS/email**
3. **System lojalnościowy**
4. **Integracje:**
   Kalendarz Google/Outlook
   Płatności online (np. Przelewy24)

9.4. Responsywność i aplikacja mobilna

Rozwiązania:

1. **Progressive Web App (PWA):**
   a. Plik `manifest.json`
   b. Service Worker dla offline mode

10. Załączniki

10.1 Fragmenty kodu źródłowego

```php
<?php
$komunikat = ''; // zmienna na komunikat

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $conn = new mysqli('localhost', 'root', '', 'restauracja');
    if ($conn->connect_error) {
        die("Błąd połączenia z bazą danych: " . $conn->connect_error);
    }

    $imie = $conn->real_escape_string($_POST['imie']);
    $nazwisko = $conn->real_escape_string($_POST['nazwisko']);
    $nazwa_uzytkownika = $conn->real_escape_string($_POST['nazwa_uzytkownika']);
    $email = $conn->real_escape_string($_POST['email']);
    $haslo = password_hash($_POST['password'], PASSWORD_DEFAULT);
    $uprawnienia = 'uzytkownik';

    $checkQuery = "SELECT * FROM uzytkownicy WHERE nazwa_uzytkownika = '$nazwa_uzytkownika' OR email = '$email'";
    $result = $conn->query($checkQuery);

    if ($result->num_rows > 0) {
        $komunikat = "Użytkownik o podanej nazwie lub e-mailu już istnieje.";
    } else {
        $sql = "INSERT INTO uzytkownicy (imie, nazwisko, nazwa_uzytkownika, email, haslo, uprawnienia)
                VALUES ('$imie', '$nazwisko', '$nazwa_uzytkownika', '$email', '$haslo', '$uprawnienia')";

        if ($conn->query($sql) === TRUE) {
            $komunikat = "Rejestracja zakończona sukcesem!";
        } else {
            $komunikat = "Błąd podczas rejestracji: " . $conn->error;
        }
    }

    $conn->close();
}
?>
```

ten kod PHP obsługuje rejestrację użytkownika w systemie restauracji. Łączy się z bazą danych MySQL, pobiera dane z formularza (imię, nazwisko, login, email, hasło), sprawdza, czy użytkownik już istnieje, a jeśli nie – zapisuje go do bazy z zahashowanym hasłem i domyślnymi uprawnieniami.

```php
<?php
session_start();
require_once 'db_config.php';

if (isset($_SESSION['user_id'])) {
    header('Location: index.php');
    exit;
}

$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    try {
        $stmt = $pdo->prepare("SELECT id, imie, nazwisko, haslo, uprawnienia FROM uzytkownicy WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if ($user && $password === $user['haslo']) { // Porównanie w czystym tekście
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['user_name'] = $user['imie'] . ' ' . $user['nazwisko'];
            $_SESSION['user_role'] = $user['uprawnienia'];

            if (isset($_POST['remember'])) {
                setcookie('remember_user', $user['id'], time() + (30 * 24 * 60 * 60), "/");
            }

            header('Location: index.php');
            exit;
        } else {
            $error = 'Nieprawidłowy email lub hasło';
        }
    } catch (PDOException $e) {
        $error = 'Błąd systemu. Spróbuj ponownie.';
    }
}
?>
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Logowanie</title>
  <link rel="stylesheet" href="login.css" />
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
</head>
<body>
  <div class="login-container">
    <h2>Zaloguj się</h2>
    <?php if (!empty($error)): ?>
      <div class="error-message"><?= htmlspecialchars($error) ?></div>
    <?php endif; ?>

    <form action="login.php" method="post">
      <input type="email" name="email" placeholder="Adres email" required />
      <input type="password" name="password" placeholder="Hasło" required />

      <label class="remember-me">
        <input type="checkbox" name="remember" id="remember">
        <span>Zapamiętaj mnie</span>
      </label>

      <div class="buttons">
        <button type="submit" class="login-btn">Zaloguj</button>
        <a href="rejestracja.php" class="register-btn">Zarejestruj się</a>
      </div>
    </form>

    <a href="index.php" class="back-btn">← Powrót do strony głównej</a>
  </div>
</body>
</html>
```

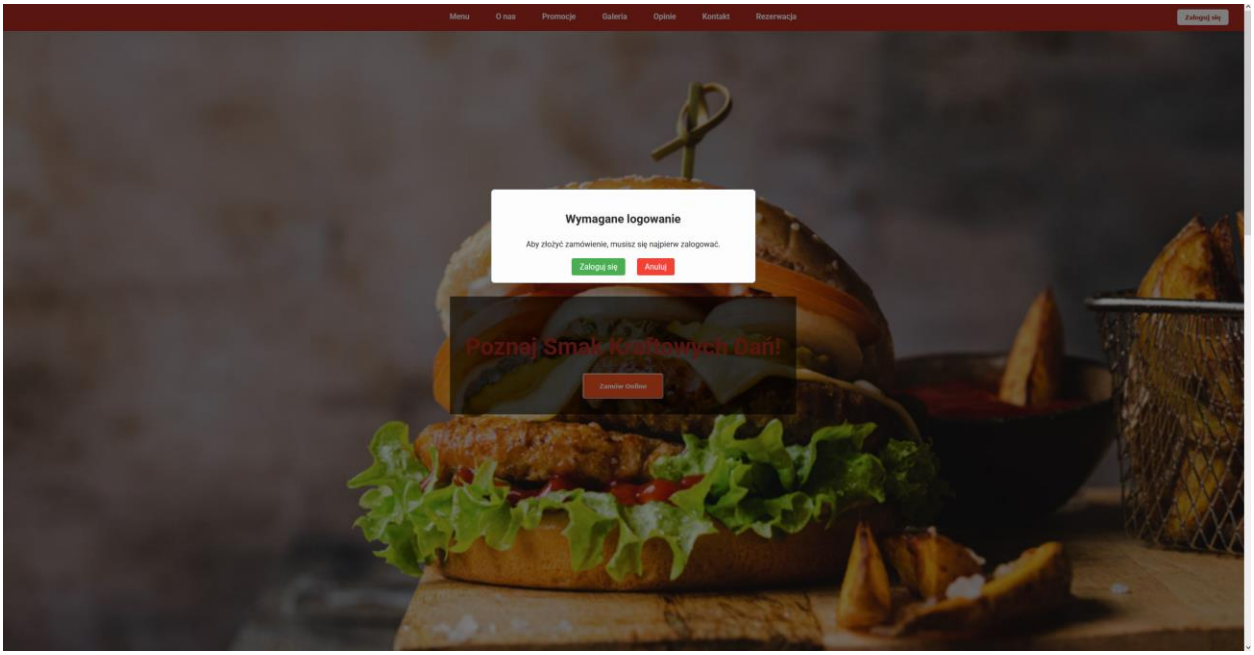
Logowanie użytkownika:

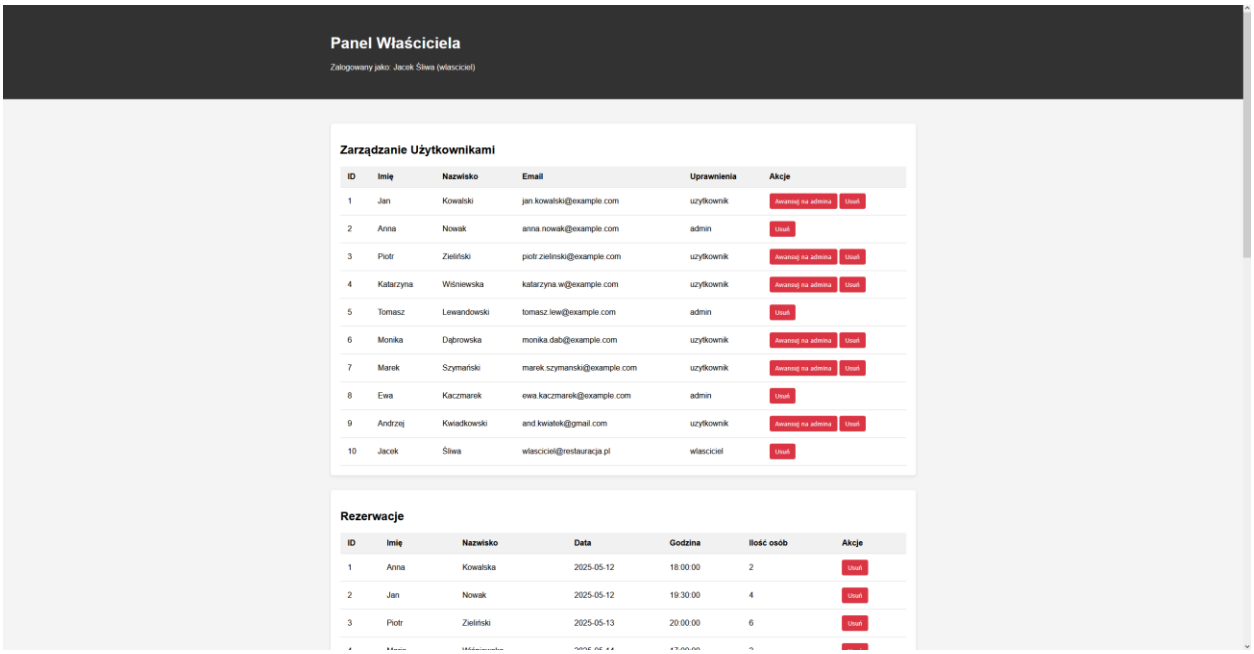
1. Sprawdza, czy użytkownik jest już zalogowany (przez sesję).
2. Pobiera email i hasło z formularza.
3. **BŁĄD:** Porównuje hasło w tekście jawnym
4. Po zalogowaniu tworzy sesję i opcjonalnie ciasteczko ("Zapamiętaj mnie").
5. Wyświetla formularz logowania z linkami do rejestracji i strony głównej.

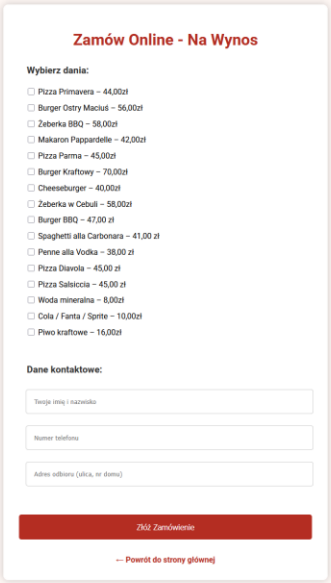
```php
<div class="container">
    <?php if ($isOwner): ?>
    <div class="section owner-only">
        <h2>Zarządzanie Użytkownikami</h2>
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Imię</th>
                    <th>Nazwisko</th>
                    <th>Email</th>
                    <th>Uprawnienia</th>
                    <th>Akcje</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($users as $user): ?>
                <tr>
                    <td><?= htmlspecialchars($user['id']) ?></td>
                    <td><?= htmlspecialchars($user['imie']) ?></td>
                    <td><?= htmlspecialchars($user['nazwisko']) ?></td>
                    <td><?= htmlspecialchars($user['email']) ?></td>
                    <td><?= htmlspecialchars($user['uprawnienia']) ?></td>
                    <td>
                        <?php if ($user['uprawnienia'] === 'uzytkownik'): ?>
                        <form method="POST" style="display:inline;">
                            <input type="hidden" name="user_id" value="<?= $user['id'] ?>">
                            <button type="submit" name="promote_to_admin" class="btn promote-btn">Awansuj na admina</button>
                        </form>
                        <?php endif; ?>

                        <form method="POST" style="display:inline;">
                            <input type="hidden" name="user_id" value="<?= $user['id'] ?>">
                            <button type="submit" name="delete_user" class="btn"
                                onclick="return confirm('Czy na pewno chcesz usunąć tego użytkownika?')">Usuń</button>
                        </form>
                    </td>
                </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>
    <?php endif; ?>
```

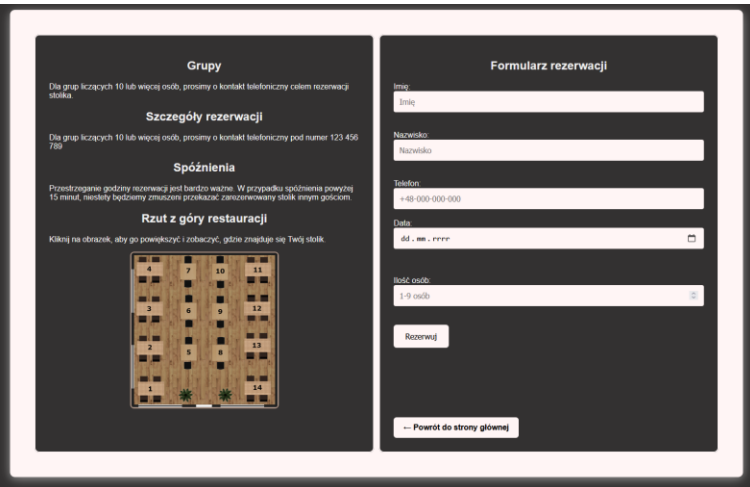
- Pokazuje tabelę użytkowników (tylko jeśli `$isOwner=true`)
- Dla każdego użytkownika:
  - Wyświetla dane (ID, imię, email, uprawnienia)
  - **Przyciski:**
    - Awansuj na admina (dla zwykłych użytkowników)
    - Usuń (z potwierdzeniem)

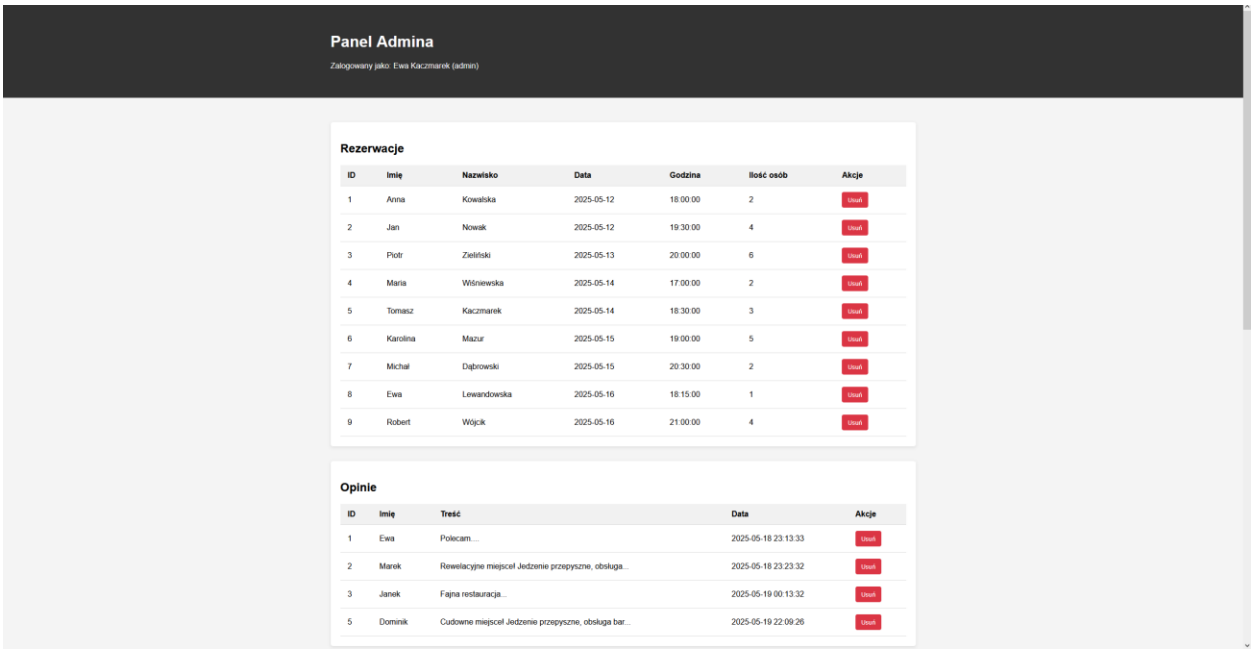
10.2 Zrzuty ekranu działania systemu



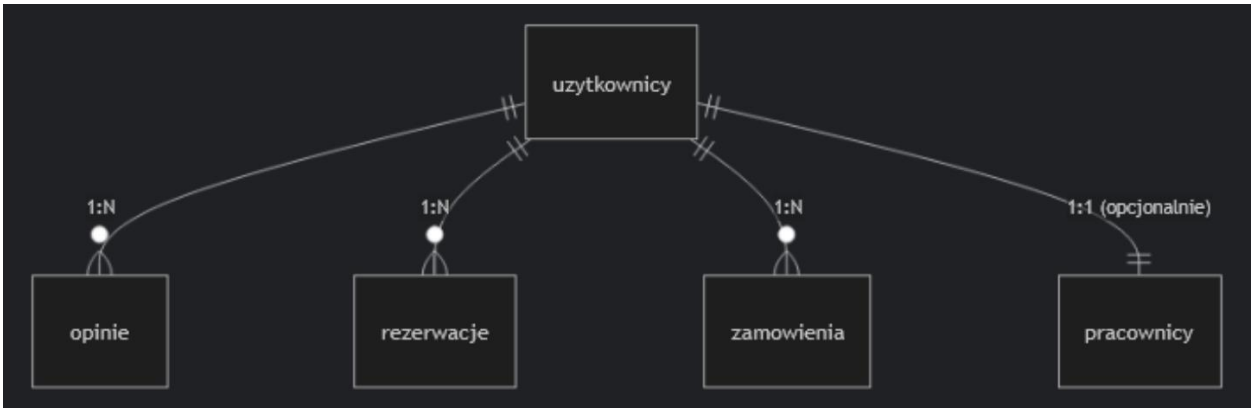



Zamów Online - Na Wynos
Wybierz dania:
Pizza Primavera – 44,00zł
Burger Ostry Maciuś – 56,00zł
Żeberka BBQ – 58,00zł
Makaron Pappardelle – 42,00zł
Pizza Parma – 45,00zł
Burger Kraftowy – 70,00zł
Cheeseburger – 40,00zł
Żeberka w Cebuli – 58,00zł
Burger BBQ – 47,00 zł
Spaghetti alla Carbonara – 41,00 zł
Penne alla Vodka – 38,00 zł
Pizza Diavola – 45,00 zł
Pizza Salsiccia – 45,00 zł
Woda mineralna – 8,00zł
Cola / Fanta / Sprite – 10,00zł
Piwo kraftowe – 16,00zł
Dane kontaktowe:
Twoje imię i nazwisko
Numer telefonu
Adres odbioru (ulica, nr domu)
Złóż Zamówienie
← Powrót do strony głównej


Grupy
Dla grup liczących 10 lub więcej osób, prosimy o kontakt telefoniczny celem rezerwacji stolika.
Szczegóły rezerwacji
Dla grup liczących 10 lub więcej osób, prosimy o kontakt telefoniczny pod numer 123 456 789
Spóźnienia
Przestrzeganie godziny rezerwacji jest bardzo ważne. W przypadku spóźnienia powyżej 15 minut, niestety będziemy zmuszeni przekazać zarezerwowany stolik innym gościom.
Rzut z góry restauracji
Kliknij na obrazek, aby go powiększyć i zobaczyć, gdzie znajduje się Twój stolik.
4
7
10
11
3
6
9
12
2
5
8
13
1
14
Formularz rezerwacji
Imię:
Imię
Nazwisko:
Nazwisko
Telefon:
+48-000-000-000
Data:
dd . mm . rrrr
Ilość osób:
1-9 osób
Rezerwuj
← Powrót do strony głównej



10.3 Diagram bazy danych



10.4 Lista błędów/uwag i pomysłów na ulepszenia

Lp.	Typ	Opis	Priorytet

1	Błąd	Brak walidacji numeru telefonu w formularzu rezerwacji	Wysoki
2	Ulepszenie	Dodanie paginacji do listy dań w menu.php	Średni
3	Nowa funkcja	Integracja z Google Maps dla śledzenia dostaw	Niski
4	Bezpieczeństwo	Brak limitu prób logowania (ryzyko brute force)	Krytyczny
5	UI/UX	Brak potwierdzenia SMS przy rezerwacji	Wysoki