Sprawozdanie z projektu “SYSTEM POS”

Wprowadzenie do ASP.NET

Julia Trzeciakiewicz

02.06.2025

Program to system POS (Point of Sale) stworzony z myślą o obsłudze gastronomii, restauracji, kawiarni czy pubów. Głównym celem aplikacji jest wsparcie pracy kelnera podczas przyjmowania i obsługi zamówień gości, zarządzania rachunkami oraz ich zamykania po zakończeniu wizyty. System zapewnia również rozbudowany panel administracyjny, który umożliwia właścicielowi lub managerowi lokalu pełną kontrolę nad menu, kategoriami, użytkownikami oraz historią sprzedaży. Dzięki temu rozwiązaniu możliwe jest usprawnienie obsługi klienta, przyspieszenie realizacji zamówień oraz zwiększenie efektywności zarządzania restauracją.

Technologie

1. Framework:
   System został zbudowany w technologii ASP.NET Web Forms z wykorzystaniem języka C#.
2. Baza danych:
   Do przechowywania danych wykorzystano Microsoft SQL Server. Komunikacja z bazą odbywa się poprzez ADO.NET oraz procedury składowane (stored procedures), co zapewnia bezpieczeństwo i wydajność operacji na danych.

Najważniejsze funkcje systemu POS

1. Logowanie kelnera i obsługa zamówień
   a. Każdy kelner loguje się do systemu za pomocą indywidualnego PIN-u.
   b. Po zalogowaniu może wybrać stolik, przyjąć zamówienie od gościa i dodać produkty do rachunku.
   c. System umożliwia szybkie dodawanie, edycję lub usuwanie pozycji z zamówienia w trakcie obsługi gościa oraz dodawanie komentarzy.
2. Obsługa rachunków i płatności
   a. Kelner może w dowolnym momencie podejrzeć aktualny rachunek, rozdzielić go na kilka osób lub zakończyć i rozliczyć rachunek, wybierając formę płatności (gotówka/karta).
   b. Po zamknięciu rachunku stolik staje się ponownie dostępny dla nowych gości.
3. Panel administracyjny
   a. Administrator ma dostęp do zarządzania kategoriami i produktami w menu (dodawanie, edycja, usuwanie, zmiana statusu aktywności, dodawanie zdjęć).
   b. Możliwe jest zarządzanie użytkownikami (kelnerami) i przeglądanie historii zamówień.
   c. System rejestruje wszystkie zamówienia wraz z ich statusem.
4. Bezpieczeństwo i wygoda
   a. Każdy użytkownik ma przypisane uprawnienia (kelner, admin).
   b. System zapewnia walidację danych i bezpieczne przechowywanie informacji o zamówieniach oraz użytkownikach.

Główna struktura programu

1. Pliki .aspx – Warstwa prezentacji (widok)

Pliki .aspx to pliki odpowiadające za wygląd i strukturę każdej strony internetowej w systemie. Znajduje się w nich kod HTML, kontrolki serwerowe ASP.NET oraz deklaracje stylów i layoutu. To właśnie te pliki są wysyłane do przeglądarki użytkownika po przetworzeniu przez serwer.

2. Pliki .aspx.cs – Kod zaplecza (code-behind, logika strony)

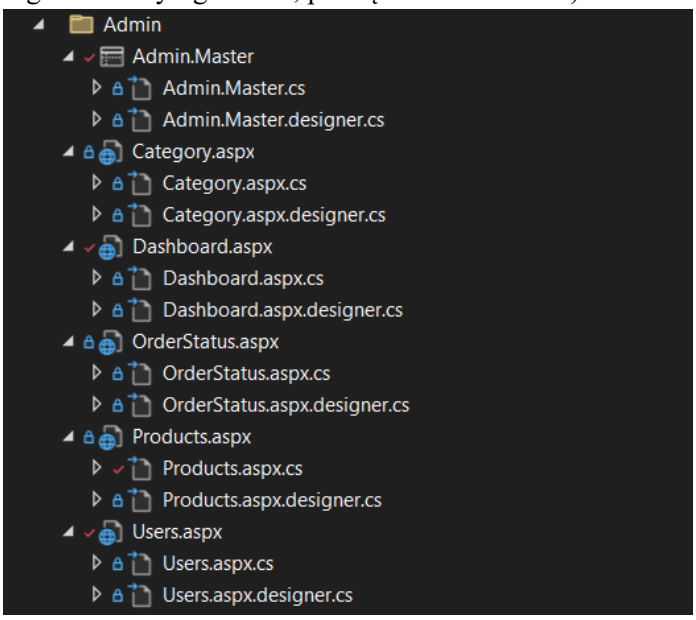
Każdy plik .aspx ma swój odpowiadający plik .aspx.cs. W tych plikach znajduje się cała logika działania strony:

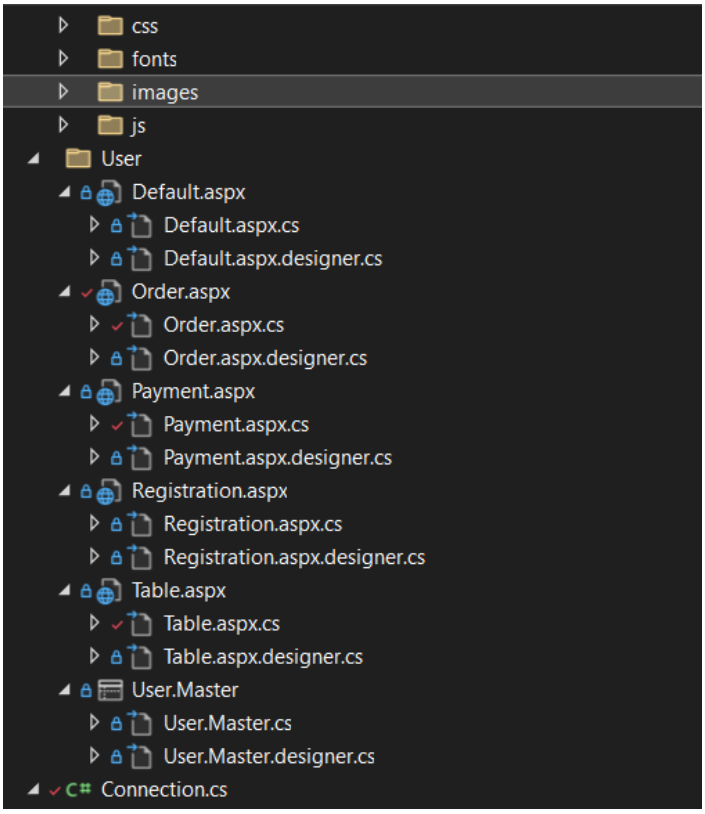
- obsługa zdarzeń (np. kliknięcie przycisku, wybór z listy),
- komunikacja z bazą danych (pobieranie, zapisywanie, edycja danych),
- walidacja danych użytkownika,
- zarządzanie sesją,
- dynamiczne przypisywanie danych do kontrolek na stronie.

3. Pliki szablonów (Master Pages: .master i .master.cs)

Pliki .master to tzw. szablony (Master Pages), które definiują wspólny układ i elementy dla wielu stron w aplikacji.
Dzięki nim każda strona ma ten sam nagłówek, menu, stopkę, styl i logikę wspólną dla całej aplikacji.

- .master – zawiera HTML, kontrolki wspólne dla wszystkich stron (np. menu, logo, panel użytkownika).
- .master.cs – kod zaplecza szablonu, obsługuje wspólne zdarzenia (np. logowanie/wylogowanie, przełączanie widoków).





Category.aspx.cs
Zarządzanie kategoriami produktów w panelu administratora.
Najważniejsze funkcje:

- getCategories() – pobiera i wyświetla listę kategorii z bazy danych.
- btnAddOrUpdate_Click – dodaje nową kategorię lub aktualizuje istniejącą (wraz z obsługą zdjęcia i walidacją).
- rCategpry_ItemCommand – obsługuje edycję i usuwanie kategorii.

Product.aspx.cs
Zarządzanie produktami w menu przez administratora.

Najważniejsze funkcje:

- getProducts() – pobiera i wyświetla listę wszystkich produktów z bazy, wraz z kategoriami, cenami, statusami i zdjęciami.
- btnAddOrUpdate_Click – dodaje nowy produkt lub aktualizuje istniejący (obsługa formularza, walidacja, zapis zdjęcia).
- rProduct_ItemCommand – obsługuje edycję i usuwanie produktów.

OrderStatus.aspx.cs
Wyświetlanie historii zamówień w panelu administratora.
Najważniejsze funkcje:

- GetAllOrders() – pobiera i prezentuje wszystkie zamówienia.

- GetStatusBadgeClass(string status) – zwraca status zamówienia.
- rOrderStatus_ItemCommand – obsługuje wyświetlanie szczegółów zamówienia.
- rOrderStatus_ItemDataBound – ładuje szczegóły pozycji zamówienia

Users.aspx.cs
Zarządzanie użytkownikami przez administratora.
Najważniejsze funkcje:

- getUsers() – pobiera i wyświetla listę użytkowników.
- rUsers_ItemCommand – umożliwia usuwanie użytkowników z systemu.

Default.aspx.cs
Logowanie użytkownika do systemu za pomocą PIN-u.
Najważniejsze funkcje:

- Pobieranie PIN-u z czterech pól tekstowych i jego walidacja.
- Logowanie jako admin (PIN: 0000) lub jako użytkownik (sprawdzenie w bazie).

Payment.aspx.cs
Obsługa procesu płatności za zamówienie.
Najważniejsze funkcje:

- CompletePayment(string paymentMode) – obsługuje płatność (gotówka/karta), zapisuje płatność w bazie, aktualizuje status zamówienia i stolika, obsługuje transakcje SQL.

Registration.aspx.cs
Rejestracja nowego użytkownika.
Najważniejsze funkcje:

- getUserDetails() – pobiera dane użytkownika.
- btnRegister_Click – obsługuje rejestrację, waliduje dane i zdjęcie, zapisuje do bazy.

Table.aspx.cs
Wybór stolika przez kelnera oraz obsługa przypisania zamówienia do stolika.
Najważniejsze funkcje:

- LoadTables() – ładuje listę dostępnych stolików z bazy.
- GetTableStatus(int tableId) – generuje status stolika (wolny/zajęty).
- rTables_ItemCommand – obsługuje wybór stolika i przypisuje zamówienie (nowe lub kontynuacja).
- rTables_ItemDataBound – dynamicznie ustawia status i linki dla stolików.

Order.aspx.cs
Obsługa zamówienia przy wybranym stoliku – dodawanie produktów do zamówienia, edycja koszyka, składanie i finalizacja zamówienia.

Najważniejsze funkcje:

- LoadOrder() – pobiera aktualne zamówienie dla wybranego stolika i wyświetla jego zawartość (produkty, ilości, sumę).
- btnAddToCart_Click – dodaje wybrany produkt do zamówienia (koszyka) lub zwiększa jego ilość.
- AddCommentToItem() - Pozwala dodać lub edytować komentarz do konkretnej pozycji zamówienia (np. „bez sosu”, „na ostro”).
- SplitOrder() - Pozwala rozdzielić wybrane produkty na osobne rachunki (np. dla kilku gości przy jednym stoliku).

User.master.cs
Szablon główny dla kelnera.

Najważniejsze funkcje:

- Zarządzanie wyglądem strony, obsługa logowania/wylogowania, przekierowania do profilu/rejestracji, aktualizacja licznika koszyka.

Admin.master.cs
Szablon główny dla panelu administratora.

Najważniejsze funkcje:

- Obsługa wylogowania administratora i przekierowanie do strony logowania użytkownika.

Przykładowy opis kodu na podstawie pliku Default.aspx oraz Default.aspc.cs

Przykładowe kaskady:

```css
.login-container {
    display: flex;
    min-height: 80vh;
    align-items: center;
    justify-content: center;
    background-color: var(--light-color);
}

.login-card {
    width: 100%;
    max-width: 900px;
    border-radius: 15px;
    overflow: hidden;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
    display: flex;
    border: 2px solid var(--dark-color);
}
```

```css
.login-image {
    flex: 1;
    background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.7)),
                url('https://bing.com/th/id/BCO.b82010e9-13f7-4260-95e3-9ec51b564f84.png');
    background-size: cover;
    background-position: center;
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 2rem;
    color: var(--primary-color);
}

.login-image h2 {
    font-size: 2.5rem;
    margin-bottom: 1rem;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

.login-container

- display: flex;
  Ustawia kontener jako elastyczny, umożliwiając łatwe centrowanie i układanie dzieci w linii.
- min-height: 80vh;
  Kontener zajmuje co najmniej 80% wysokości widoku przeglądarki.
- align-items: center; justify-content: center;
  Elementy wewnątrz kontenera są wyśrodkowane zarówno w pionie, jak i w poziomie.
- background-color: var(--light-color);
  Tło kontenera jest ustawione na kolor zdefiniowany w zmiennej CSS --light-color.

.login-card

- width: 100%; max-width: 980px;
  Karta logowania zajmuje całą szerokość kontenera, ale nie przekracza 980px.
- border-radius: 15px;
  Zaokrąglenie rogów karty.
- overflow: hidden;
  Wszystko wychodzące poza kartę jest ukryte.
- box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
  Dodaje cień pod kartą, co nadaje jej efekt unoszenia.
- display: flex;
  Wnętrze karty jest również elastyczne – umożliwia łatwe rozmieszczenie dzieci (np. obrazek i formularz).
- border: 2px solid var(--dark-color);
  Ramka karty w kolorze zdefiniowanym przez zmienną --dark-color.

.login-image

- flex: 1;
  Obrazek zajmuje tyle miejsca, ile może w elastycznym układzie.
- background:
  Ustawia tło jako połączenie półprzezroczystego gradientu (ciemnego) oraz obrazka z podanego URL.
  Gradient zapewnia lepszą czytelność tekstu na obrazku.
- background-size: cover; background-position: center;
  Obrazek tła jest rozciągnięty tak, by zakryć cały obszar i wyśrodkowany.
- display: flex; align-items: center; justify-content: center;
  Wszelkie elementy wewnątrz (np. tekst) są wyśrodkowane pionowo i poziomo.
- padding: 2rem;
  Dodaje wewnętrzny odstęp.
- color: var(--primary-color);
  Ustawia kolor tekstu na wartość zdefiniowaną w zmiennej --primary-color.

.login-image h2

- font-size: 2.5rem;
  Duży rozmiar czcionki dla nagłówka.
- margin-bottom: 1rem;
  Odstęp pod nagłówkiem.
- text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
  Dodaje cień do tekstu, poprawiając jego czytelność na tle obrazka.

Script:

```
function appendDigit(digit) {
    const pinInputs = document.querySelectorAll('.pin-digit');
    for (let i = 0; i < pinInputs.length; i++) {
        if (pinInputs[i].value === '') {
            pinInputs[i].value = digit;
            if (i < pinInputs.length - 1) pinInputs[i + 1].focus();
            break;
        }
    }
    updateHiddenField();
}

function clearPin() {
    const pinInputs = document.querySelectorAll('.pin-digit');
    pinInputs.forEach(input => {
        input.value = '';
    });
    pinInputs[0].focus();
    updateHiddenField();
}

function updateHiddenField() {
    const pinInputs = document.querySelectorAll('.pin-digit');
    let pin = '';
    pinInputs.forEach(input => {
        pin += input.value;
    });
    document.getElementById('<%= hfPassword.ClientID %>').value = pin;
}

window.onload = function () {
    var seconds = 5;
    setTimeout(function () {
        document.getElementById("<%=lblMsg.ClientID %>").style.display = "none";
    }, seconds * 1000);

    document.querySelector('.pin-digit').focus();
};

function moveToNext(current, nextId) {
    if (current.value.length === 1) {
        document.getElementById(nextId).focus();
        updateHiddenField();
    }
}
```

1. appendDigit(digit)

- Dodaje wciśniętą cyfrę do pierwszego pustego pola PIN-u.
- Po wpisaniu cyfry automatycznie przenosi fokus do następnego pola.
- Po każdej zmianie wywołuje updateHiddenField(), aby zaktualizować ukryte pole z pełnym PIN-em.

2. clearPin()

- Czyści wszystkie pola PIN-u.
- Ustawia fokus na pierwsze pole PIN-u.
- Aktualizuje ukryte pole z PIN-em.

3. updateHiddenField()

- Zbiera wartości ze wszystkich pól PIN-u i łączy je w jeden ciąg znaków.
- Wartość tę wpisuje do ukrytego pola (hfPassword.ClientID), które jest wysyłane na serwer.

4. window.onload

- Po załadowaniu strony ustawia timer, który po 5 sekundach ukrywa komunikat (lblMsg.ClientID).
- Ustawia fokus na pierwsze pole PIN-u.

5. moveToNext(current, nextId)

- Po wpisaniu cyfry w bieżące pole, jeśli jest ono pełne, automatycznie przenosi fokus do następnego pola PIN-u.
- Aktualizuje ukryte pole z PIN-em.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <div class="login-container">
        <div class="login-card">
            <div class="login-image">
                <div>
                    <h2>Witaj!</h2>
                    <p>Zaloguj się, aby aby przejść dalej</p>
                </div>
            </div>

            <div class="login-form">
                <div class="form-header">
                    <h2>Logowanie</h2>
                    <p>Wprowadź swój kod PIN</p>
                </div>

                <asp:Label ID="lblMsg" runat="server" Visible="false" CssClass="message-alert"></asp:Label>

                <div class="pin-container">
                    <asp:TextBox ID="txtPin1" runat="server" CssClass="pin-digit" MaxLength="1" onkeyup="moveToNext(this, 'txtPin2')" autocomplete="off" />
                    <asp:TextBox ID="txtPin2" runat="server" CssClass="pin-digit" MaxLength="1" onkeyup="moveToNext(this, 'txtPin3')" autocomplete="off" />
                    <asp:TextBox ID="txtPin3" runat="server" CssClass="pin-digit" MaxLength="1" onkeyup="moveToNext(this, 'txtPin4')" autocomplete="off" />
                    <asp:TextBox ID="txtPin4" runat="server" CssClass="pin-digit" MaxLength="1" autocomplete="off" />
                    <asp:HiddenField ID="hfPassword" runat="server" />
                </div>

                <div class="numpad">
                    <button type="button" class="numpad-btn" onclick="appendDigit('1')">1</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('2')">2</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('3')">3</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('4')">4</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('5')">5</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('6')">6</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('7')">7</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('8')">8</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('9')">9</button>
                    <button type="button" class="numpad-btn clear-btn" onclick="clearPin()">C</button>
                    <button type="button" class="numpad-btn" onclick="appendDigit('0')">0</button>
                    <asp:Button ID="btnLogin" runat="server" Text="ZALOGUJ SIĘ" CssClass="numpad-btn submit-btn" OnClick="btnLogin_Click" />
                </div>
            </div>
        </div>
    </div>
</asp:Content>
```

1. Układ ogólny

Całość znajduje się w głównym kontenerze , który odpowiada za wyśrodkowanie i stylizację karty logowania.

2. Karta logowania

- – główny panel logowania, podzielony na dwie sekcje:
    - login-image:
      Zawiera powitanie i instrukcję dla użytkownika
    - login-form:
      Sekcja z właściwym formularzem logowania.

3. Formularz logowania

- Nagłówek:
  Sekcja z tytułem i instrukcją
- Komunikat:
  <asp:Label ID="lblMsg" ... /> – etykieta do wyświetlania komunikatów (np. o błędnym PIN-ie).

4. Wprowadzanie PIN-u

- Sekcja zawiera cztery pola tekstowe (TextBoxy) do wprowadzania kolejnych cyfr PIN-u:
    - Każde pole ma MaxLength="1", więc można wpisać tylko jedną cyfrę.
    - onkeyup="moveToNext(this, 'txtPinX')" – po wpisaniu cyfry kursor automatycznie przechodzi do następnego pola.
    - Autouzupełnianie jest wyłączone (autocomplete="off").
- <asp:HiddenField ID="hfPassword" ... /> – ukryte pole, do którego JavaScript wpisuje pełny PIN z czterech pól, aby przesłać go na serwer.

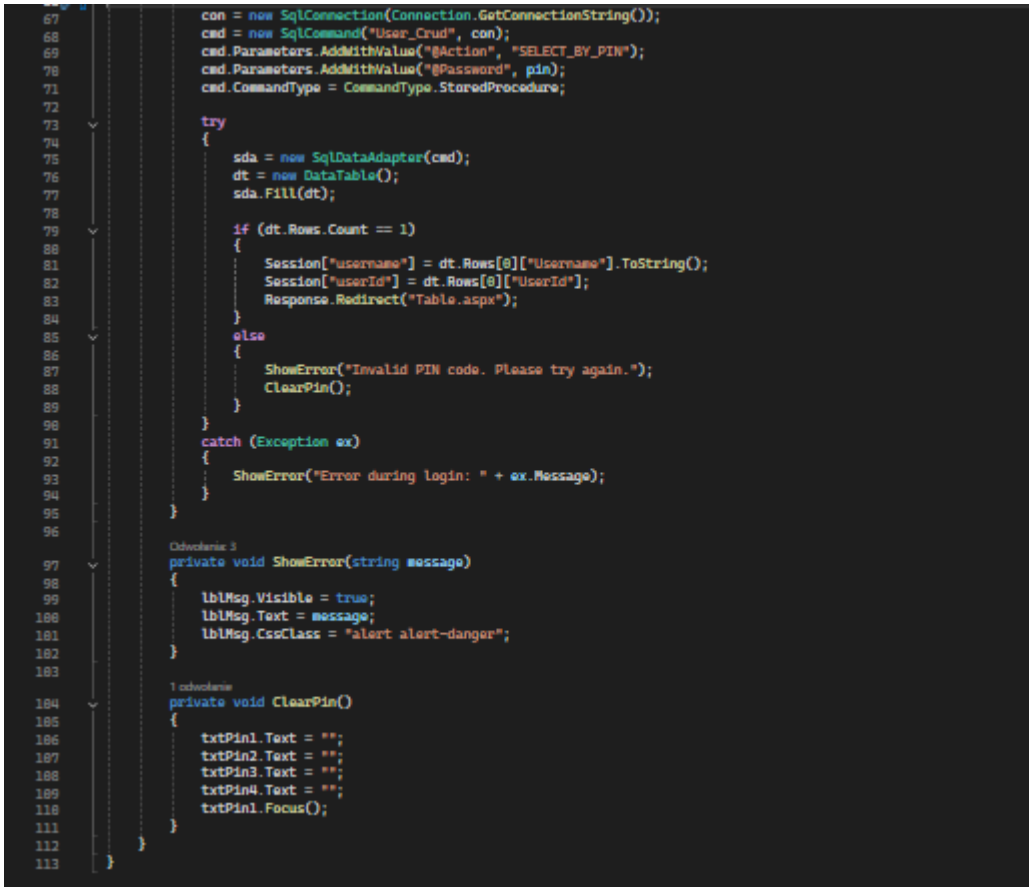
5. Klawiatura ekranowa

- – wirtualna klawiatura numeryczna z przyciskami 0–9 oraz przyciskiem "C" (czyści PIN).
    - Każdy przycisk wywołuje funkcję JavaScript appendDigit('X'), która wpisuje cyfrę do pierwszego wolnego pola PIN.
    - Przycisk "C" wywołuje funkcję clearPin(), która czyści wszystkie pola PIN.

6. Przycisk logowania

- <asp:Button ID="btnLogin" ... /> – przycisk do wysłania formularza i próby logowania, wywołuje metodę serwerową btnLogin_Click.

```
public partial class Default : System.Web.UI.Page
{
    SqlConnection con;
    SqlCommand cmd;
    SqlDataAdapter sda;
    DataTable dt;


    protected TextBox txtPin1;
    protected TextBox txtPin2;
    protected TextBox txtPin3;
    protected TextBox txtPin4;
    protected HiddenField hfPassword;
    protected Label lblMsg;

    Odwołania: 0
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["userId"] != null)
        {
            Response.Redirect("Table.aspx");
        }

        ContentPlaceHolder contentPlaceHolder = (ContentPlaceHolder)this.Master.FindControl("ContentPlaceHolder1");

        if (contentPlaceHolder != null)
        {
            txtPin1 = (TextBox)contentPlaceHolder.FindControl("txtPin1");
            txtPin2 = (TextBox)contentPlaceHolder.FindControl("txtPin2");
            txtPin3 = (TextBox)contentPlaceHolder.FindControl("txtPin3");
            txtPin4 = (TextBox)contentPlaceHolder.FindControl("txtPin4");
            hfPassword = (HiddenField)contentPlaceHolder.FindControl("hfPassword");
            lblMsg = (Label)contentPlaceHolder.FindControl("lblMsg");
        }
        else
        {
            throw new Exception("ContentPlaceHolder1 not found in the Master Page.");
        }
    }

    Odwołania: 0
    protected void btnLogin_Click(object sender, EventArgs e)
    {

        string pin = txtPin1.Text + txtPin2.Text + txtPin3.Text + txtPin4.Text;

        if (string.IsNullOrEmpty(pin) || pin.Length != 4)
        {
            ShowError("Please enter a 4-digit PIN code");
            return;
        }

        if (pin == "0000")
        {
            Session["admin"] = "Admin";
            Response.Redirect("../Admin/Dashboard.aspx");
            return;
        }
```



```
            con = new SqlConnection(Connection.GetConnectionString());
            cmd = new SqlCommand("User_Crud", con);
            cmd.Parameters.AddWithValue("@Action", "SELECT_BY_PIN");
            cmd.Parameters.AddWithValue("@Password", pin);
            cmd.CommandType = CommandType.StoredProcedure;

            try
            {
                sda = new SqlDataAdapter(cmd);
                dt = new DataTable();
                sda.Fill(dt);

                if (dt.Rows.Count == 1)
                {
                    Session["username"] = dt.Rows[0]["Username"].ToString();
                    Session["userId"] = dt.Rows[0]["UserId"];
                    Response.Redirect("Table.aspx");
                }
                else
                {
                    ShowError("Invalid PIN code. Please try again.");
                    ClearPin();
                }
            }
            catch (Exception ex)
            {
                ShowError("Error during login: " + ex.Message);
            }
        }

        Odwołania: 3
        private void ShowError(string message)
        {
            lblMsg.Visible = true;
            lblMsg.Text = message;
            lblMsg.CssClass = "alert alert-danger";
        }

        1 odwołanie
        private void ClearPin()
        {
            txtPin1.Text = "";
            txtPin2.Text = "";
            txtPin3.Text = "";
            txtPin4.Text = "";
            txtPin1.Focus();
        }
    }
}
```

1. Deklaracje pól i zmiennych

Na początku pliku zadeklarowane są zmienne do obsługi połączenia z bazą danych (SqlConnection, SqlCommand, SqlDataAdapter, DataTable) oraz kontrolki interfejsu użytkownika (cztery pola PIN, ukryte pole z PIN-em, etykieta na komunikaty).

2. Metoda Page_Load

- Sprawdza, czy użytkownik jest już zalogowany (czy istnieje sesja "userId"). Jeśli tak, przekierowuje go od razu na stronę stolików.
- Następnie pobiera z master page referencje do czterech pól PIN, ukrytego pola oraz etykiety na komunikaty.
  Dzięki temu kod-behind ma dostęp do kontrolek zdefiniowanych w szablonie strony.

3. Metoda btnLogin_Click

- Po kliknięciu przycisku logowania, kod zbiera wartości ze wszystkich czterech pól PIN i skleja je w jeden ciąg znaków.
- Sprawdza, czy PIN jest czterocyfrowy. Jeśli nie, wyświetla komunikat o błędzie.
- Jeżeli PIN to "0000", użytkownik jest traktowany jako admin – tworzona jest sesja "admin" i następuje przekierowanie do panelu administracyjnego.
- W przeciwnym razie kod łączy się z bazą danych i wywołuje procedurę składowaną, która sprawdza, czy taki PIN istnieje w bazie użytkowników.
    - Jeśli znaleziono użytkownika, tworzone są odpowiednie sesje i następuje przekierowanie do wyboru stolika.

- Jeśli nie – wyświetlany jest komunikat o błędnym PIN-ie.

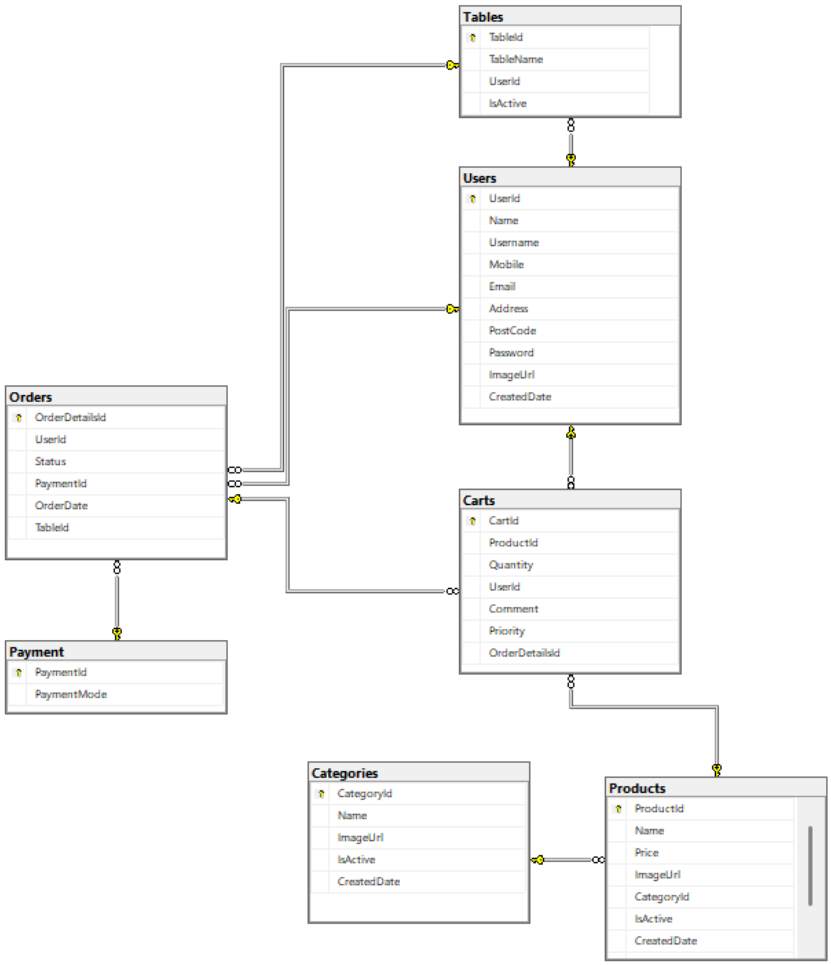
4. Metoda ShowError

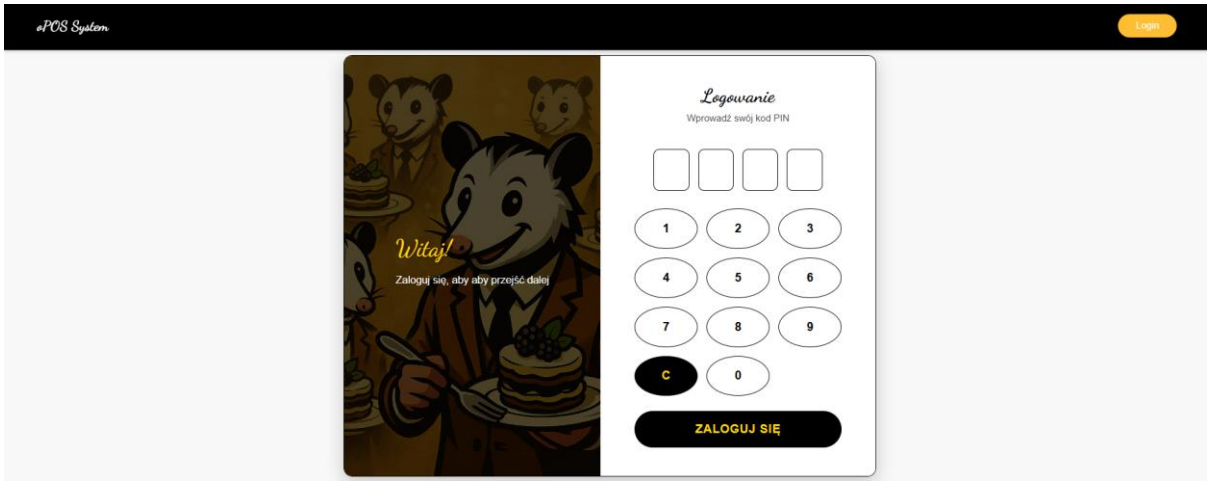
- Ustawia etykietę z komunikatem na widoczną i przypisuje jej tekst oraz odpowiednią klasę CSS dla alertu błędu.

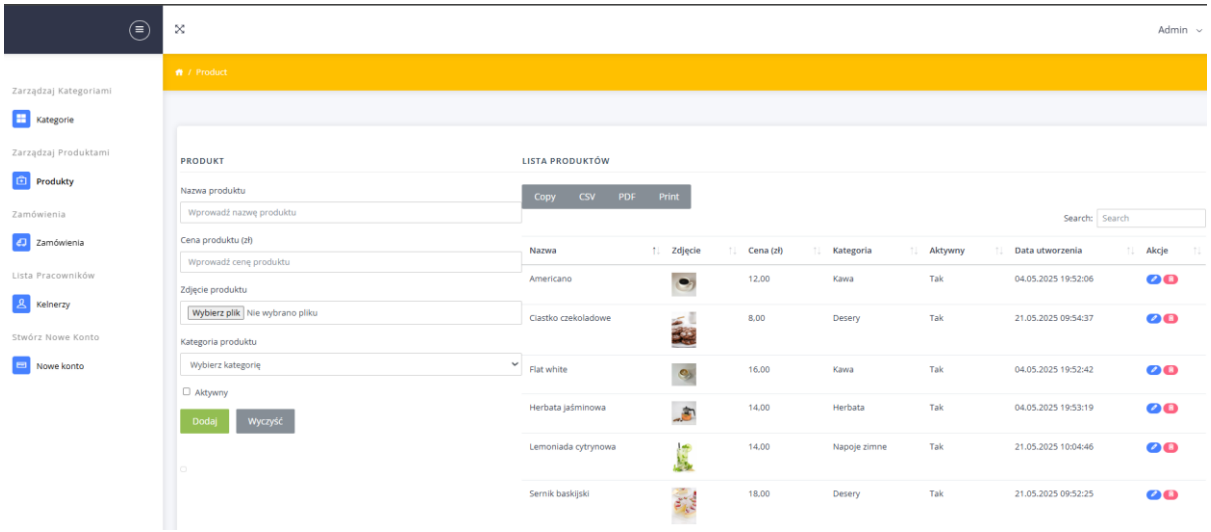
5. Metoda ClearPin

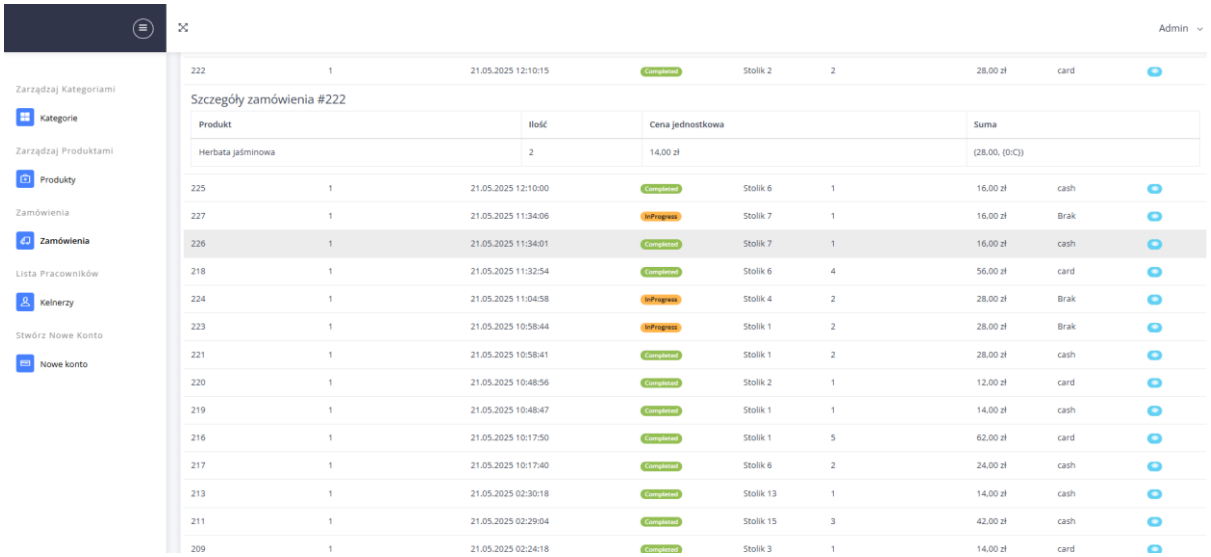
- Czyści wszystkie cztery pola PIN i ustawia fokus na pierwsze z nich, umożliwiając szybkie ponowne wpisanie PIN-u.

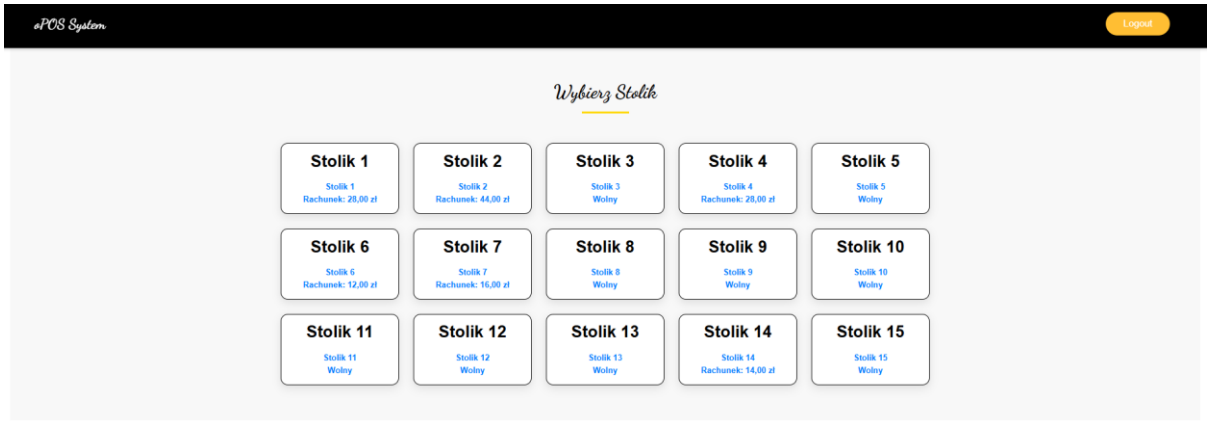
Baza Danych:

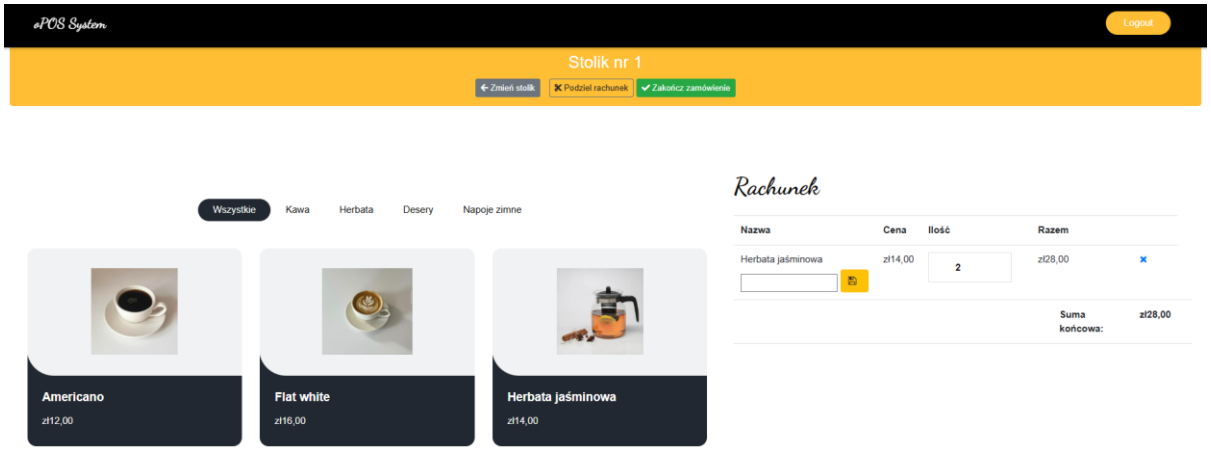



aPOS System
Login
Witaj!
Zaloguj się, aby aby przejść dalej
Logowanie
Wprowadź swój kod PIN
1
2
3
4
5
6
7
8
9
C
0
ZALOGUJ SIĘ


Admin
Product
Zarządzaj Kategoriami
Kategorie
Zarządzaj Produktami
Produkty
Zamówienia
Zamówienia
Lista Pracowników
Kelnerzy
Stwórz Nowe Konto
Nowe konto
PRODUKT
Nazwa produktu
Wprowadź nazwę produktu
Cena produktu (zł)
Wprowadź cenę produktu
Zdjęcie produktu
Wybierz plik
Nie wybrano pliku
Kategoria produktu
Wybierz kategorię
Aktywny
Dodaj
Wyczyść
LISTA PRODUKTÓW
Copy
CSV
PDF
Print
Search:
Nazwa
Zdjęcie
Cena (zł)
Kategoria
Aktywny
Data utworzenia
Akcje
Americano
12,00
Kawa
Tak
04.05.2025 19:52:06
Ciastko czekoladowe
8,00
Desery
Tak
21.05.2025 09:54:37
Flat white
16,00
Kawa
Tak
04.05.2025 19:52:42
Herbata jaśminowa
14,00
Herbata
Tak
04.05.2025 19:53:19
Lemoniada cytrynowa
14,00
Napoje zimne
Tak
21.05.2025 10:04:46
Sernik baskijski
18,00
Desery
Tak
21.05.2025 09:52:25


Admin
Zarządzaj Kategoriami
Kategorie
Zarządzaj Produktami
Produkty
Zamówienia
Zamówienia
Lista Pracowników
Kelnerzy
Stwórz Nowe Konto
Nowe konto
222 1 21.05.2025 12:10:15 Completed Stolik 2 2 28,00 zł card
Szczegóły zamówienia #222
Produkt
Ilość
Cena jednostkowa
Suma
Herbata jaśminowa
2
14,00 zł
(28,00, (0:C))
225 1 21.05.2025 12:10:00 Completed Stolik 6 1 16,00 zł cash
227 1 21.05.2025 11:34:06 InProgress Stolik 7 1 16,00 zł Brak
226 1 21.05.2025 11:34:01 Completed Stolik 7 1 16,00 zł cash
218 1 21.05.2025 11:32:54 Completed Stolik 6 4 56,00 zł card
224 1 21.05.2025 11:04:58 InProgress Stolik 4 2 28,00 zł Brak
223 1 21.05.2025 10:58:44 InProgress Stolik 1 2 28,00 zł Brak
221 1 21.05.2025 10:58:41 Completed Stolik 1 2 28,00 zł cash
220 1 21.05.2025 10:48:56 Completed Stolik 2 1 12,00 zł card
219 1 21.05.2025 10:48:47 Completed Stolik 1 1 14,00 zł cash
216 1 21.05.2025 10:17:50 Completed Stolik 1 5 62,00 zł card
217 1 21.05.2025 10:17:40 Completed Stolik 6 2 24,00 zł cash
213 1 21.05.2025 02:30:18 Completed Stolik 13 1 14,00 zł cash
211 1 21.05.2025 02:29:04 Completed Stolik 15 3 42,00 zł cash
209 1 21.05.2025 02:24:18 Completed Stolik 3 1 14,00 zł card


ePOS System
Logout
Wybierz Stolik
Stolik 1
Stolik 1
Rachunek: 28,00 zł
Stolik 2
Stolik 2
Rachunek: 44,00 zł
Stolik 3
Stolik 3
Wolny
Stolik 4
Stolik 4
Rachunek: 28,00 zł
Stolik 5
Stolik 5
Wolny
Stolik 6
Stolik 6
Rachunek: 12,00 zł
Stolik 7
Stolik 7
Rachunek: 16,00 zł
Stolik 8
Stolik 8
Wolny
Stolik 9
Stolik 9
Wolny
Stolik 10
Stolik 10
Wolny
Stolik 11
Stolik 11
Wolny
Stolik 12
Stolik 12
Wolny
Stolik 13
Stolik 13
Wolny
Stolik 14
Stolik 14
Rachunek: 14,00 zł
Stolik 15
Stolik 15
Wolny


ePOS System
Logout
Stolik nr 1
Zmień stolik
Podziel rachunek
Zakończ zamówienie
Wszystkie
Kawa
Herbata
Desery
Napoje zimne
Americano
zł12,00
Flat white
zł16,00
Herbata jaśminowa
zł14,00
Rachunek
Nazwa
Cena
Ilość
Razem
Herbata jaśminowa
zł14,00
2
zł28,00
Suma końcowa:
zł28,00