Projekt
Sklep internetowy

Projekt grupowy

Autorzy projektu:
Łukasz Skrzypczak,
Marcin Ratajczak

Spis treści

1. Opis działania projektu

Baza danych: Strona internetowa korzysta z bazy danych SQLite, która przechowuje dane dotyczące użytkowników, produktów, zamówień i ról (administrator, moderator, użytkownik). Baza umożliwia przechowywanie danych, takich jak informacje o kontach użytkowników, zamówieniach oraz produktach dostępnych w sklepie.

Rejestracja i logowanie użytkowników: Strona umożliwia rejestrację kont użytkowników. Każdy użytkownik może założyć konto, podając podstawowe dane (imię, nazwisko, e-mail, hasło). Proces rejestracji zapisuje dane użytkownika w bazie. Po założeniu konta użytkownik może się zalogować do systemu, uzyskując dostęp do funkcji związanych z zakupami i zamawianiem produktów. Istnieje także możliwość wylogowania się z systemu.

Role użytkowników: W systemie dostępne są trzy role:

- **Administrator:** ma pełny dostęp do zarządzania użytkownikami (tworzenie, edytowanie, usuwanie kont), produktami (dodawanie nowych produktów, aktualizacja informacji o produktach, usuwanie), a także przeglądanie wszystkich zamówień.
- **Moderator:** ma ograniczony dostęp, mogąc zarządzać produktami (dodawać, edytować, usuwać), ale nie ma możliwości zarządzania kontami użytkowników.
- **Użytkownik:** zwykły klient sklepu, który może przeglądać produkty, dodawać je do koszyka i składać zamówienia.

Podstrony publiczne i prywatne: Strona podzielona jest na:

- **Podstrony publiczne:** dostępne dla wszystkich użytkowników, np. strona główna sklepu, przeglądanie produktów.
- **Podstrony prywatne:** dostępne tylko po zalogowaniu tj. podgląd koszyka, składanie zamówień, zarządzanie zamówieniami

Koszyk zakupowy: Każdy zalogowany użytkownik ma możliwość dodawania produktów do koszyka. Informacje o produktach w koszyku są przechowywane w sesji. Użytkownik może przeglądać zawartość koszyka, modyfikować liczbę produktów lub usuwać produkty z koszyka.

Zamawianie produktów: Po złożeniu zamówienia dane dotyczące zamówienia (lista produktów, dane użytkownika) są zapisywane w bazie danych. Użytkownik otrzymuje potwierdzenie zamówienia. Administrator i moderator mają możliwość przeglądania wszystkich zamówień, a administrator może nimi zarządzać.

Indywidualna interakcja użytkownika: Użytkownik może przesyłać różne formularze (np. formularz zmiany danych osobowych). Te dane będą zapisywane w bazie i przypisane do konkretnego konta użytkownika. Każda interakcja użytkownika z systemem, taka jak zakupy, zmiany w profilu, jest przechowywana w bazie danych.

2. Specyfikacja wykorzystanych technologii

Framework MVC

- **Nazwa**: ASP.NET MVC
- **Wersja**: 6.0
- **Opis**: ASP.NET MVC to framework do budowy aplikacji webowych, który stosuje wzorzec projektowy Model-View-Controller. Umożliwia on organizację kodu, co ułatwia jego utrzymanie i testowanie. MVC oddziela logikę biznesową, logikę prezentacji i interakcje użytkownika.

Język programowania

- **Nazwa**: C#
- **Wersja**: 8.0
- **Opis**: C# jest nowoczesnym, obiektowym językiem programowania, który jest wykorzystywany do tworzenia aplikacji na platformie .NET. Oferuje silne typowanie, wsparcie dla programowania asynchronicznego oraz bogaty zestaw bibliotek.

Baza danych

- **Nazwa**: SQLite
- **Wersja**: 8.0.10
- **Aplikacja graficzna bazy:** DB Browser for SQLite
- **Plik bazy:** Główny katalog / Sklepinternetowy.db

ORM (Object-Relational Mapping)

- **Nazwa**: Entity Framework Core
- **Wersja**: 6.0
- **Opis**: Entity Framework Core to framework ORM, który umożliwia mapowanie obiektów .NET do relacyjnych baz danych. Ułatwia on pracę z danymi, pozwalając na wykonywanie operacji CRUD (Create, Read, Update, Delete) bez konieczności pisania skomplikowanych zapytań SQL.

Frontend

- **Technologie**: HTML5, CSS3, JavaScript

- **Opis**: HTML5 i CSS3 są standardowymi językami do tworzenia struktury i stylizacji stron internetowych. JavaScript, wspierany przez wybrane frameworki, umożliwia dodanie interaktywności do aplikacji.

Środowisko deweloperskie

- **Nazwa**: Visual Studio
- **Wersja**: 2022 17.11.2
- **Opis**: Visual Studio to zintegrowane środowisko deweloperskie (IDE), które wspiera programowanie w C# i inne języki. Oferuje narzędzia do debugowania, testowania oraz zarządzania projektami.

Serwer aplikacji

- **Nazwa**: Kestrel
- **Opis**: Kestrel to wydajny serwer HTTP, który jest częścią ASP.NET Core. Umożliwia uruchamianie aplikacji w trybie lokalnym oraz produkcyjnym.

Kontrola wersji

- **Narzędzie**: Git
- **Platforma**: GitHub
- **Link do zdalnego repozytorium**: https://github.com/S1Dek/Sklep-internetowy
- **Opis**: Git to system kontroli wersji, który umożliwia śledzenie zmian w kodzie źródłowym. GitHub lub GitLab służą jako platforma do współpracy nad kodem i zarządzania repozytoriami

Pakiety NuGet

- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.AspNetCore.Identity.UI
- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Sqlite
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.VisualStudio.Web.CodeGeneration.Design
- Microsoft.EntityFrameworkCore.Tools

3. Instrukcje pierwszego uruchomienia

Otwórz katalog główny, gdzie znajduje się plik sklep-internetowy.sln

Po załadowaniu projektu, przejdź do Konsoli Menedżera Pakietów, wpisz i wykonaj polecenie Update-Database.

Po pomyślnym zakończeniu migracji, kliknij przycisk "https" w Visual Studio lub naciśnij F5, aby uruchomić projekt.

Aplikacja zostanie uruchomiona w domyślnej przeglądarce internetowej.

Po otwarciu strony logowania, możesz zalogować się za pomocą wcześniej zdefiniowanych kont użytkowników (loginy oraz hasła znajdują się w głównym katalogu projektu) lub założyć nowe konto, aby przetestować funkcjonalność sklepu.

4. Opis struktury projektu

Główne katalogi i pliki

wwwroot: Katalog zawierający pliki statyczne aplikacji, takie jak skrypty JavaScript, style CSS.

Controllers

AccountController.cs: Kontroler odpowiedzialny za funkcje logowania, rejestracji i wylogowania.

AdminController.cs: Kontroler dla administratora odpowiedzialny za zarządzanie użytkownikami.

CartController.cs: Kontroler do zarządzania koszykiem użytkowników.

HomeController.cs: Kontroler odpowiedzialny za listowanie produktów z bazy.

OrdersController.cs: Kontroler zarządzający zamówieniami.

ProductsController.cs: Kontroler do zarządzania produktami w sklepie.

Data

ApplicationDbContext.cs: Klasa kontekstu bazy danych używana przez Entity Framework do zarządzania bazą danych aplikacji.

Migrations

Katalog zawierający pliki migracji bazy danych, które są używane do aktualizowania struktury bazy danych w czasie.

Models

CartItem.cs: Model dla przedmiotów w koszyku.

ErrorViewModel.cs: Model do wyświetlania błędów.

Order.cs: Model dla zamówień.

Product.cs: Model dla produktów w sklepie.

User.cs: Model dla użytkowników aplikacji.

Usługi:

FakeEmailSender.cs: Symulowanie przesyłania mail'a potwierdzającego

Views

Account: Widoki związane z funkcjami konta (logowanie, rejestracja).

Admin: Widoki dla panelu administracyjnego.

Cart: Widoki do wyświetlania zawartości koszyka.

Home: Widok strony głównej.

Orders: Widok do wyświetlania zamówień.

Products: Widoki dla produktów.

Program.cs

Główna klasa programu zawierająca konfigurację aplikacji oraz punkt wejścia do aplikacji.

5. Wylistowane wszystkie modele

EditUserViewModel

- Id: Unikalny identyfikator użytkownika, typ string. Klucz do odwołania się do konkretnego użytkownika.
- User: Obiekt User, który przechowuje informacje o użytkowniku, takich jak imię i nazwisko, adres e-mail, oraz rolę.
- AssignedRoles: Przydzielona rola do której obecnie przypisany jest użytkownik.

- AllRoles: Lista wszystkich dostępnych ról w systemie.

Opis: Model EditUserViewModel jest używany do zarządzania rolami użytkownika w aplikacji. Pozwala na edycję przypisanych ról oraz wybór nowych ról podczas edycji profilu użytkownika. Obejmuje walidację ról i ich integrację z systemem zarządzania tożsamością ASP.NET Identity.

LoginViewModel

- Email: Adres e-mail użytkownika. Wymagane, walidowane jako adres e-mail (EmailAddress).
- Password: Hasło użytkownika. Wymagane, walidowane jako pole typu Password.
- RememberMe: Checkbox umożliwiający użytkownikowi wybranie opcji „Zapamiętaj mnie”, co powoduje dłuższe utrzymanie sesji logowania.

Opis: Model LoginViewModel jest używany do zbierania danych potrzebnych do logowania użytkownika. Pole Email i Password są niezbędne do uwierzytelnienia, a RememberMe umożliwia użytkownikowi wybór opcji zapamiętywania sesji.

RegisterViewModel

- Email: Adres e-mail użytkownika, wymagany do rejestracji.
- Password: Hasło użytkownika, wymagane do rejestracji. Walidowane jako hasło.
- ConfirmPassword: Potwierdzenie hasła, które musi zgadzać się z polem Password.
- FirstName: Imię użytkownika, wymagane.
- LastName: Nazwisko użytkownika, wymagane.

Opis: Model RegisterViewModel jest używany do zbierania danych rejestracyjnych użytkownika. Zawiera pola wymagane do utworzenia nowego konta, takie jak adres e-mail, hasło, imię i nazwisko. Walidacja obejmuje porównanie haseł (ConfirmPassword).

CartItem

- Id: Unikalny identyfikator elementu w koszyku.
- ProductId: Identyfikator produktu w koszyku.
- Product: Obiekt Product związany z produktem w koszyku.
- Quantity: Ilość zakupionego produktu.
- UserId: Identyfikator użytkownika, do którego należy koszyk.
- User: Obiekt User, który reprezentuje właściciela koszyka.

Opis: Model CartItem reprezentuje pojedynczy przedmiot w koszyku użytkownika. Zawiera informacje o produkcie, ilości oraz powiązaniu z użytkownikiem. Umożliwia zarządzanie koszykiem zakupowym użytkowników.

Order

- Id: Unikalny identyfikator zamówienia.

- UserId: Identyfikator użytkownika, który złożył zamówienie.
- User: Obiekt User reprezentujący właściciela zamówienia.
- OrderDate: Data złożenia zamówienia.
- TotalAmount: Łączna kwota zamówienia.
- OrderDetails: Szczegóły dotyczące zamówienia.

Opis: Model Order jest używany do przechowywania danych o zamówieniach użytkowników, w tym o produktach, ich ilości i cenie, oraz powiązaniach z użytkownikiem. Jest niezbędny do zarządzania procesem zakupów na stronie.

OrderDetail

- Id: Unikalny identyfikator szczegółu zamówienia.
- OrderId: Identyfikator zamówienia, do którego należą dane.
- Order: Obiekt Order, który reprezentuje zamówienie.
- ProductId: Identyfikator produktu w szczególe zamówienia.
- Product: Obiekt Product, który reprezentuje produkt w szczególe zamówienia.
- Quantity: Ilość zamówionego produktu.
- Price: Cena produktu w ramach zamówienia.

Opis: Model OrderDetail jest używany do szczegółowego przedstawienia elementów w zamówieniu, takich jak produkt, ilość i cena. Służy do przechowywania informacji o zakupionych produktach przez użytkownika.

Product

- Id: Unikalny identyfikator produktu.
- Name: Nazwa produktu, wyświetlana na stronie.
- Image: Ścieżka do obrazu produktu.
- Price: Cena produktu.
- Stock: Ilość dostępna w magazynie.

Opis: Model Product przechowuje dane o produkcie, takie jak nazwa, cena, dostępność oraz obraz. Jest potrzebny do zarządzania ofertą produktową w sklepie internetowym.

User

- FirstName: Imię użytkownika.
- LastName: Nazwisko użytkownika.
- Role: Rola użytkownika, domyślnie „user”.

Opis: Model User rozszerza podstawowy typ użytkownika ASP.NET Identity, umożliwiając dodanie dodatkowego pola rola. Jest używany do zarządzania profilami użytkowników i ich tożsamością w systemie.

6. Wylistowane kontrolery

AccountController

1.Login

- Metody HTTP: GET, POST
- Parametry: LoginViewModel model
- Opis: Wyświetla formularz logowania i przetwarza zgłoszenia logowania. Na poprawnym zgłoszeniu POST z odpowiednimi danymi, użytkownik jest logowany i przekierowany na stronę główną.
- Zwraca: Przekierowanie na akcję "Index" w przypadku udanego logowania, lub odświeżenie formularza logowania z błędami na nieudanym logowaniu.

2.Register

- Metody HTTP: GET, POST
- Parametry: RegisterViewModel model
- Opis: Wyświetla formularz rejestracji i przetwarza zgłoszenia rejestracji. Na poprawnym zgłoszeniu POST, nowy użytkownik jest utworzony i zalogowany, po czym przekierowany na stronę główną.
- Zwraca: Przekierowanie na akcję "Index" w przypadku udanej rejestracji, lub odświeżenie formularza rejestracji z błędami na nieudanej rejestracji.

3.Logout

- Metody HTTP: GET
- Opis: Wylogowuje aktualnie zalogowanego użytkownika i przekierowuje na stronę główną.
- Zwraca: Przekierowanie na "Index" po wylogowaniu.

AdminController

1.ManageUsers

- Metody HTTP: GET
- Opis: Wyświetla listę wszystkich użytkowników w systemie.
- Zwraca: Widok pokazujący listę użytkowników.

2.Edit

- Metody HTTP: GET, POST

- Parametry: int id, EditUserViewModel model
- Opis: Wyświetla formularz edycji użytkownika i przetwarza zmiany. Na żądanie GET pobiera dane użytkownika o podanym ID. Na żądanie POST aktualizuje dane użytkownika oraz jego role.
- Zwraca: Przekierowanie na "ManageUsers" w przypadku udanej aktualizacji, lub odświeżenie formularza edycji z błędami na nieudanej aktualizacji.

3.Details

- Metody HTTP: GET
- Parametry: int id
- Opis: Wyświetla szczegóły określonego użytkownika.

4.Delete

- Metody HTTP: GET, POST
- Parametry: int id
- Opis: Usuwa użytkownika o określonym ID. Żądanie POST realizuje usunięcie użytkownika.
- Zwraca: Przekierowanie na "ManageUsers" po usunięciu, lub widok z błędem, jeśli użytkownik nie mógł być usunięty.

CartController

1.Index

- Metody HTTP: GET
- Opis: Wyświetla koszyk zakupowy aktualnie zalogowanego użytkownika z wszystkimi dodanymi produktami.
- Zwraca: Widok z produktami w koszyku.

2.AddToCart

- Metody HTTP: POST
- Parametry: int productId
- Opis: Dodaje produkt do koszyka aktualnego użytkownika. Zwiększa ilość, jeśli produkt już jest w koszyku.
- Zwraca: Przekierowanie na "Index" pokazującą zaktualizowany koszyk.

3.RemoveFromCart

- Metody HTTP: GET
- Parametry: int id
- Opis: Usuwa określony produkt z koszyka użytkownika.
- Zwraca: Przekierowanie na "Index" po usunięciu produktu.

4.Checkout

- Metody HTTP: GET, POST
- Opis: Wyświetla stronę do finalizacji zamówienia i przetwarza zamówienie. Żądanie POST tworzy zamówienie i czyści koszyk.
- Zwraca: Przekierowanie na "Index" po udanym finalizowaniu zamówienia, lub odświeżenie strony z błędami na nieudanym finalizowaniu zamówienia.

HomeController

1.Index

- Metody HTTP: GET
- Opis: Wyświetla stronę główną z listą produktów dostępnych w sklepie.

OrdersController

1.Index

- Metody HTTP: GET
- Opis: Wyświetla listę zamówień wykonanych przez aktualnie zalogowanego użytkownika.
- Zwraca: Widok pokazujący listę zamówień.

2.Details

- Metody HTTP: GET
- Parametry: int id
- Opis: Wyświetla szczegóły określonego zamówienia wykonane przez aktualnego użytkownika.
- Zwraca: Widok pokazujący szczegóły zamówienia.

ProductsController

1.Index

- Metody HTTP: GET
- Opis: Wyświetla listę wszystkich produktów w sklepie.

2.Details

- Metody HTTP: GET
- Parametry: int id
- Opis: Wyświetla szczegóły określonego produktu.

3.Create

- Metody HTTP: GET, POST
- Parametry: Product product
- Opis: Wyświetla formularz dodawania produktu i przetwarza zgłoszenia. Na poprawnym zgłoszeniu POST produkt jest dodany do bazy.
- Zwraca: Przekierowanie na "Index" po udanym utworzeniu, lub odświeżenie formularza z błędami na nieudanym utworzeniu.

4.Edit

- Metody HTTP: GET, POST
- Parametry: int id, Product product
- Opis: Wyświetla formularz edycji produktu i przetwarza zmiany. Na żądanie GET pobiera dane produktu. Na żądanie POST aktualizuje dane produktu.
- Zwraca: Przekierowanie na "Index" po udanej aktualizacji, lub odświeżenie formularza z błędami na nieudanej aktualizacji.

5.Delete

- Metody HTTP: GET, POST
- Parametry: int id
- Opis: Usuwa określony produkt. Żądanie POST realizuje usunięcie produktu.
- Zwraca: Przekierowanie na "Index" po usunięciu.

7. Opis systemu użytkowników

Role użytkowników: W systemie dostępne są trzy role:

- **Administrator:** ma pełny dostęp do zarządzania użytkownikami (tworzenie, edytowanie, usuwanie kont), produktami (dodawanie nowych produktów, aktualizacja informacji o produktach, usuwanie), a także przeglądanie wszystkich zamówień.
- **Moderator:** ma ograniczony dostęp, mogąc zarządzać produktami (dodawać, edytować, usuwać), ale nie ma możliwości zarządzania kontami użytkowników.
- **Użytkownik:** zwykły klient sklepu, który może przeglądać produkty, dodawać je do koszyka i składać zamówienia.

Podstrony publiczne i prywatne: Strona podzielona jest na:

- **Podstrony publiczne:** dostępne dla wszystkich użytkowników, np. strona główna sklepu, przeglądanie produktów.
- **Podstrony prywatne:** dostępne tylko po zalogowaniu tj. podgląd koszyka, składanie zamówień, zarządzanie zamówieniami

Nadanie roli:

- **Admin:** W zakładce “Zarządzaj użytkownikami” następnie “Edytuj” może zmieniać role każdego użytkownika
- **Moderator:** Brak możliwości
- **User:** Brak możliwości

8. Charakterystyka najciekawszych funkcjonalności

Łączna liczba zamówień i całkowity dochód

Podsumowanie zamówień pozwala w szybki sposób ocenić skalę działalności sklepu internetowego. Łączna liczba zamówień prezentowana w panelu administratora daje pogląd na popularność sklepu w danym okresie.

Całkowity dochód przedstawiony w formie liczbowej i walutowej ułatwia analizę wyników finansowych bez konieczności ręcznego obliczania tych danych.

Najczęściej zamawiany produkt

Panel wyświetla produkt, który był najczęściej zamawiany przez klientów. Wskazuje nie tylko nazwę produktu, ale także łączną liczbę zamówionych sztuk, co jest szczególnie użyteczne w planowaniu magazynowym oraz strategiach marketingowych (np. promocja bestsellerów).

Analiza trendów takich produktów pozwala podejmować świadome decyzje o wprowadzaniu nowych asortymentów lub zwiększeniu zapasów najbardziej popularnych towarów.

Dlaczego warto to wyróżnić?

Funkcjonalność statystyk łączy prostotę w prezentacji z bogatymi możliwościami analitycznymi, które pomagają zarządzać sklepem i wspierać jego rozwój. Przykłady zastosowań obejmują planowanie kampanii promocyjnych, monitorowanie wyników finansowych czy optymalizację stanów magazynowych na podstawie danych o najpopularniejszych produktach.

9. Diagram przypadków użycia