Maria Gierszon-Nikolayenko

Przedmiot: Programowanie zaawansowane

Grupa: K51

Cel

Celem aplikacji było stworzenie strony internetowej sklepu z autorską biżuterią tworzoną z naturalnych kamieni.

Strona Główna

Po wejściu w link do strony na platformie Azure (https://jewelrystore.azurewebsites.net/) lub kodu na platformie GitHub (https://github.com/jembotokocham/MyJewelryStore) ukazuje się strona główna "Home". Strona zawiera nagłówek z powitaniem oraz opisem sklepu, a także tabelę z obrazkami różnych produktów oferowanych w sklepie. W sekcji poniżej tabeli znajduje się lista opisów produktów, wraz z ich orientacyjnymi cenami.

Kolejna dostępna bez logowania zakładka to "Privacy" gdzie znajduje się Polityka prywatności.

Zakładki "Zamówienia", "Dodawanie nowych produktów", "Zmiana Statusu" - do nich wymagane jest logowanie użytkownika. Po wybraniu zakładki jest możliwość założenia konta lub zalogowania. Można użyć przykładowych danych do logowania użytkownika już dodanego do bazy:

Email: abc@gmail.com

Password: Maria1996@

Zamówienia

Widok

Widok główny wyświetla tabelę zamówień, w której każde zamówienie składa się z następujących kolumn:
→ Email - adres email użytkownika, który dokonał zamówienia
→ Zamówiony artykuł - identyfikator zamawianego produktu
→ Opcje - szczegóły zamówienia

Zamówienia są wyświetlane w pętli foreach, dla każdego zamówienia tworzona jest nowa linia tabeli z odpowiednimi wartościami. Użytkownik ma również możliwość stworzenia nowego zamówienia poprzez kliknięcie "Stwórz nowe zamówienie".

Po przejściu do strony tworzenia nowych zamówień wyświetlają się zdjęcia dostępnych produktów. Żeby złożyć zamówienie użytkownik musi wpisać swój adres e-mail oraz wybrać produkt z listy dostępnych produktów, a następnie nacisnąć przycisk "Zamów" aby dokonać zamówienia. Po zamówieniu produktu, użytkownik zostaje przeniesiony do widoku z listą swoich zamówień.

Na stronie znajduje się również opcja Szczegóły, która służy do wyświetlania szczegółów dotyczących danego zamówienia. Pobiera ona informacje o zamówieniu i wyświetla je na stronie. W widoku znajdują się informacje takie jak adres e-mail zamawiającego oraz numer produktu zamówionego. Na końcu znajduje się link, który pozwala na przejście do listy zamówień.

Model

Order to klasa reprezentująca zamówienie w systemie.

Pola:
→ Id - identyfikator zamówienia (typ int)
→ Email - adres email klienta (typ string)
→ NewProductId - identyfikator nowego produktu (typ int)
→ NewProduct - obiekt produktu, który jest przedmiotem zamówienia (typ NewProduct)
→ UserId - identyfikator użytkownika, który dokonał zamówienia (typ string)
→ User - obiekt użytkownika, który dokonał zamówienia (typ IdentityUser)
→ Name - obiekt opisujący status zamówienia (typ Status)

Kontroler

OrdersController to kontroler MVC, który udostępnia interfejs do zarządzania zamówieniami w aplikacji. Warto zwrócić uwagę, że kontroler jest oznaczony atrybutem Authorize, co oznacza, że dostęp do jego metod jest ograniczony do zalogowanych użytkowników.

Metody kontrolera:
→ Index - zwraca widok z listą zamówień danego użytkownika
→ Details - zwraca widok z szczegółami konkretnego zamówienia o danym id
→ Create - zwraca widok formularza tworzenia nowego zamówienia
→ Create (HttpPost) - przetwarza dane z formularza tworzenia nowego zamówienia i dodaje je do bazy danych
→ Edit - zwraca widok formularza edycji zamówienia o danym id
→ Edit (HttpPost) - przetwarza dane z formularza edycji zamówienia i aktualizuje je w bazie danych

→ Delete - zwraca widok z potwierdzeniem usunięcia zamówienia o danym id
→ DeleteConfirmed - usuwa zamówienie o danym id z bazy danych

Kontroler wstrzykuje obiekt klasy ApplicationDbContext do pola _context, który jest używany do komunikacji z bazą danych. Obiekt klasy UserManager<IdentityUser> jest wstrzykiwany do pola _userManager i służy do pobierania informacji o zalogowanym użytkowniku.

Dodawanie nowych produktów

Widok

Zakładka ta jest przeznaczona do funkcjonalności administracyjnych, w przypadku utworzenia roli Administrator. Znajduję się w niej widok, który wyświetla tabelę z dostępnymi produktami. Widok ten zawiera również elementy nawigacji umożliwiające edycję lub usunięcie istniejących oraz wyświetlenie szczegółów dotyczących poszczególnych produktów. Użytkownik ma również możliwość stworzenia nowego produktu poprzez kliknięcie linku "Nowy produkt".

Po kliknięciu pokazuje się widok dodawania nowych produktów do sklepu. Widok ten składa się z formularza, w którym użytkownik może podać nazwę produktu, cenę oraz opis produktu. Po wprowadzeniu tych danych i kliknięciu przycisku "Dodaj" produkt jest dodawany do sklepu.

Model

NewProduct to klasa reprezentująca produkt w systemie.

Pola:
→ ID - identyfikator produktu (typ int)
→ Name - nazwa produktu (typ string)
→ Price - cena produktu (typ int)
→ Description - opis produktu (typ string)

Kontroler

NewProductsController to kontroler MVC, który udostępnia interfejs do zarządzania produktami w aplikacji. Warto zwrócić uwagę, że kontroler jest oznaczony atrybutem Authorize co oznacza, że dostęp do jego metod jest ograniczony do zalogowanych użytkowników. Jednak jeżeli zostałaby dodana rola Administratora (której tworzenie nie jest dostępne w projekcie) wystarczyłoby zamienić atrybut na [Authorize]//(Roles = "Administrator")].

Metody kontrolera:
→ Index - zwraca widok z listą wszystkich produktów
→ Details - zwraca widok z szczegółami konkretnego produktu o danym id
→ Create - zwraca widok formularza tworzenia nowego produktu
→ Create (HttpPost) - przetwarza dane z formularza tworzenia nowego produktu i dodaje go do bazy danych
→ Edit - zwraca widok formularza edycji produktu o danym id
→ Edit (HttpPost) - przetwarza dane z formularza edycji produktu i aktualizuje go w bazie danych

→ Delete - zwraca widok z potwierdzeniem usunięcia produktu o danym id
→ DeleteConfirmed - usuwa produkt o danym id z bazy danych.

Kontroler wstrzykuje obiekt klasy ApplicationDbContext do pola _context, który jest używany do komunikacji z bazą danych.

Zmiana statusu

Widok

Zakładka ta jest przeznaczona do funkcjonalności administracyjnych, w przypadku utworzenia roli Administrator. Strona przedstawia widok dla listy zamówień. W widoku znajduje się tabela zawierająca kolumny:

→ "Numer zamówienia" - przedstawiająca zamówienia dodane przez klientów strony,
→ "Status" - status zamówienia dodany przed użytkownika
→ "Zamówiony artykuł" - artykuł który wybrał klient sklepu w danym zamówieniu
→ "Opcje" - gdzie znajdują się linki do akcji "Edytuj" i "Szczegóły".

Na górze strony znajduje się link do akcji "Dodaj nowy status". Po kliknięciu w ten link, użytkownik zostanie przekierowany do formularza umożliwiającego dodanie nowego statusu do listy. Formularz ten składa się z pola, w którym użytkownik może wybrać status "Nowe", "W realizacji" lub "Zrealizowane" z listy rozwijanej, oraz pola za pomocą którego użytkownik może wybrać również numer zamówienia, do którego chce przypisać nowy status. Po wprowadzeniu danych i naciśnięciu przycisku "Dodaj" nowy status zostanie dodany do bazy danych.

Model

Status to klasa reprezentująca status zamówienia w systemie.

Pola:
→ Id - identyfikator statusu (typ int)
→ Name - nazwa statusu (typ string)
→ OrderId - identyfikator zamówienia, dla którego status dotyczy (typ int)
→ Order - obiekt zamówienia, dla którego status dotyczy (typ Order)
→ Wartość typu Order jest opcjonalna i może mieć wartość null.

Kontroler

StatusController to kontroler MVC odpowiedzialny za obsługę akcji dotyczących statusów zamówień w systemie. Warto zwrócić uwagę, że kontroler jest oznaczony atrybutem Authorize co oznacza, że dostęp do jego metod jest ograniczony do zalogowanych użytkowników. Jednak jeżeli zostałaby dodana rola Administratora (której tworzenie nie jest dostępne w projekcie) wystarczyłoby zamienić atrybut na [Authorize]//(Roles = "Administrator")].

Akcje:
→ Index - wyświetla listę wszystkich statusów
→ Details - wyświetla szczegóły wybranego statusu
→ Create - tworzy nowy status
→ Edit - edytuje istniejący status
→ Delete - usuwa istniejący status

Dodatkowo, dla każdej akcji zostały zaimplementowane metody obsługi żądań HTTP GET oraz POST.