1. Tytuł projektu - „Sklep internetowy w oparciu o MVC”
2. Krótki opis projektu – W projekcie mam zamiar stworzyć sklep internetowy w oparciu o poznane na zajęciach zagadnienia z C# asp.net oraz MVC. Główne założenia to stworzenie katalogu produktów które są widoczne dla wszystkich(niezalogowani), ale dodawanie do koszyka jest funkcją dostępną tylko dla zalogowanych klientów. Istnieć mają 3 role użytkowników: klienci, sprzedawcy, oraz administratorzy. Klienci, tak jak wyżej dodają do koszyka, edytują go lub usuwają, śledzą też swoje zamówienia. Sprzedawcy zatwierdzają koszyk oraz obsługują utworzone z koszyka zamówienie, edytują też bazę produktów. Administratorzy mają wszystkie uprawnienia.
3. Autor – Paweł Porożyński
4. Wykorzystane technologie: .NET core 8,MS SQL Server (zintegrowane z Visual Studio u mnie wersja 15.0.4382)
5. Instrukcja pierwszego uruchomienia:
   1. Wykonanie update-database
   2. W celu poznania wszystkich funkcjonalności najlepiej zalogować się na konto admina które powstanie po kompilacji, email: [admin@admin.admin](mailto:admin@admin.admin) Hasło: Admin123!
6. Opis struktury projektu: Projekt powstał zgodnie z założeniami MVC oferowanymi w Visual Studio 2022, tzn. mamy podział na widoki, kontrolery, oraz modele. Oprócz tego w folderze wwwroot znajdziemy plik jpg dla braku zdjęcia produktu, a w folderze Data klasę DataSeeder służąca do utworzenia ról oraz konta admina. Oprócz standardowych modeli użyłem też modeli pomocniczych łączących dane z wielu modeli np. do pokazywania tabel.
7. Modele:
   1. Product- model dla produktów w sklepie
      1. Id czyli unikalny identyfikator produktu
      2. Name nazwa produktu, jest wymagana
      3. Description opis produktu
      4. Price cena, wartość domyślna to 0,00
      5. StockQuantity stan magazynowy, wartość domyślna 0
      6. ImageUrl link do obrazka
   2. Cart – model dla koszyka klienta
      1. Id unikalny identyfikator koszyka
      2. UserId połaczenie z Id usera, wymagane
      3. CreatedDate data utworzenia koszyka, wartość domyślna to aktualna data
      4. TotalPrice cena produktów z koszyka
      5. IsOpen kontrolka bool do obsługi aktualnie wybranego koszyka
      6. CartItems połaczenia z CartItem
      7. Name nazwa koszyka
      8. IsVisible kontrolka bool do sterowanie widocznością (w finalnej wersji nie używana)
   3. CartItem – model dla łączenia produktów z koszykami
      1. Id unikalny identyfikator
      2. CartId połączenie z Cart
      3. ProductId połączenie z Product
      4. Quantitiy ilość produktu, od 1 do maxymalnego inta, domyślnie 1
   4. Order – model dla zamówienia
      1. Id unikalny identyfikator
      2. UserId łączenie zamówienia z userem
      3. CreatedDate data utworzenia, domyślnie aktualna data, wymagane
      4. DeliveryDate data dostarczenia
      5. Status status zamówienia z listy wyborów, domyślnie pending
      6. OrderItems połączenie z OrderItems
   5. OrderItem model dla łączenie zamówień z produktami
      1. Id unikalny identyfikator
      2. OrderId połączenie z zamówieniem, wymagane
      3. ProductId połączenie z produktami, wymagane
      4. Quantity ilość, wymagane
      5. Price cena, wymagane
   6. RoleEditorModel model pomocniczy do nadawania ról użytkownikom
      1. UserId identyfikator użytkownika
      2. UserName nazwa użytkownika równa jego email
      3. Roles połączenie z RoleAssignmentModel
      4. SelectedRole wybrana rola
   7. RoleAssignmentModel model pomocniczy zawarty w RoleEditorModel
      1. RoleName nazwa roli
      2. IsSelected kontrolka bool do wyboru aktualnej roli
   8. CartViewModel model pomocniczy do wyświetlania koszyków
      1. Carts lista koszyków
      2. OpenCart aktualnie otwarty koszyk
      3. Products lista produktów
   9. OrderListViewModel model pomocniczy do wyświetlania listy zamówień
      1. Orders lista zamówień
   10. OrderEditViewModel model pomocniczy do edytowania zamówień
       1. OrderId id aktualnie edytowanego zamówienia
       2. Status status aktualnie edytowanego zamówienia, wymagane
       3. DeliveryDate data dostarczenia, wymagane
       4. Items lista produktów z OrderItemEditViewModel
   11. OrderItemEditViewModel model pomocniczy do edytowania produktów w zamówieniu(w wersji finalnej tylko do wyświetlania tych przedmiotów)
       1. OrderItemId id aktualnie używane id
       2. ProductId id przedmiotu
       3. ProductName aktualna nazwa przedmiotu
       4. Quantity aktualna ilość przedmiotu, wymagane od 1 do max int
       5. Price aktualna cena, wymagane od 0 do max double
8. Kontrolery z metodami
   1. ProductsController podstawowy kontroler utworzony z elementu szkieletowego
      1. Index, GET, wyświetla przedmioty
      2. Details, GET, po id przedmiotu wyświetla jego szegóły
      3. Create, GET & POST, w gecie przekierowuje do widoku, w poście obsługuje parametry Id, name, description, price, stockquantity, imageurl. Które to trafią jako nowy rekord w tabeli produktów
      4. Edit, GET & POST, w poscie przekierowuje do widoku edycji, w gecie obsługuje parametry do aktualizacji przedmiotu
      5. Delete, GET & POST, w poscie przekierowuje do widoku usunięcia, w poscie usuwa rekord przedmiotu
   2. CartsController kontroler do obsługi koszyka, wyświetlania, tworzenia, usuwania, oraz tworzenia zamówienia z koszyka
      1. Index, GET, metoda wyświetli wszystkie koszyki przypisane do id usera, jeśli nie ma koszyka dla usera to powstanie nowy, pokaże aktualnie używany koszyk, zwraca nowy obiekt do modelu CartListViewModel
      2. Create, POST, metoda do tworzenia nowego koszyka. Przyjmuje parametr nazwy, pobiera id usera, zapisuje koszyk w bazie i przekierowuje do indexu
      3. Switch, POST, metoda pomocnicza do przełączania między koszykami usera i ustawiania aktualnie używanego. Przyjmuje parametr cartId z formularza, przełącza aktywny koszyk, przekierowuje do indexu
      4. AddToCart, POST, metoda dodawania przedmiotu do koszyka. Przyjmuje parametry productId oraz quantity z formularza, sprawdza logikę dostępności magazynowej, sprawdza czy koszyk jest otwarty, dodaje lub aktualizuje produkt w CartItem, przekierowuje do indexu
      5. RemoveFromCart, POST, metoda do usuwania przedmiotu z koszyka, Przyjmuje parametr cartItemId z formularza, usuwa przedmiot z aktualne koszyka, przekierowuje do index
      6. Delete, POST, metoda do usuwania koszyka, Przyjmuje parametr cartId z formularza, usuwa koszyk, przekierowuje do index
      7. CreateOrder, POST, metoda do tworzenia zamówienia z aktualnego koszyka. Przyjmuje parametry przekazywane przez aktualny koszyk, pobiera aktualny koszyk oraz usera, tworzy nowy obiekt zamówienia, zimniejsza stany magazynowe w oparciu o aktualny koszyk, zmienia kontrolerkę koszyka na zamknięty, przekierowuje do indexu zamówień
   3. OrdersController kontroler do obsługiwania zamówień, wyświetlania, edytowania
      1. Index, GET, metoda zależna od roli usera, dla klienta pokazuje jego zamówienia; dla sprzedawcy pokazuje wszystkie zamówienia. Sprawdza rolę zalogowanego usera, pobiera unikalne id userów oraz ich adresy email do viewbag-a, zwraca nowy obiekt OrderListViewModel i przekierowuje do jego wyświetlenia
      2. Edit, GET & POST, metoda do edycji danych zamówienia. W gecie przyjmuje parametr id zamówienia z którego tworzy nowy obiekt OrderEditViewModel do którego przekierowuje. W poscie przyjmuje parametr modelu z OrderEditViewModel, aktualizuje status oraz datę dostarczenia, przekierowuje do indexu
      3. Details, GET, metoda do wyświetlania szczegółów zamówienia. Sprawdza czy zalogowany jest sprzedawca, jeżeli nie to wyświetlanie jest możliwe tylko do własnych zamówień. Tworzy nowy obiekt OrderEditViewModel i go pokazuje
   4. HomeController kontroler do wyświetlania katalogu produktów
      1. Index, GET, metoda zwraca widok modelu Products
   5. AdminController kontroler do obsługi nadawania ról userom
      1. Index, GET, metoda do pokazania listy userów oraz ich ról. Pobiera userów z userManagera a role z roleManager, wyświetla userów oraz przypisuje do ViewBag role.
      2. EditUserRoles, GET & POST, metoda w gecie towrzy nowy obiekt RoleEditorModel z danym wybranego usera. W poscie odbywa się edycja roli wybranego usera, w finalnej wersji można zmieniać lub nadawać rolę ale nie można wyzerować roli.
9. Opis systemu użytkowników
   1. Niezalogowany – użytkownik niezalogowany ma dostęp tylko do strony Home na której może zobaczyć katalog produktów, zdjęcia, opis itp.
   2. Client – klienci mają dostęp do strony Home, Carts, oraz Orders. Mogą więc przeglądać produkty, tworzyć koszyki i składać zamówienia, mogą też przeglądać swoje zamówienia.
   3. Seller – sprzedawcy mają dodatkowo możliwość dostępu do strony Products gdzie obsługują bazę produktów, mają też możliwość edycji zamówień (status oraz termin dostawy)
   4. Admin – admini mają wszystkie uprawnienia oraz dostęp do strony edycji ról użytkowników
10. Opis najciekawszych funkcjonalności
    1. Role użytkowników – w momencie kompilacji następuje utworzenie 3 ról dla użytkowników (Admin, Seller, Client), powstaje też konto admina dzięki któremu możemy nadawać role pozostałym użytkownikom w specjalnym panelu.
    2. Funkcjonalne koszyki – każdy użytkownik ma swoją bazę koszyków co ułatwia powtarzalne zamawianie przedmiotów ze sklepu, eliminuje konieczność każdorazowego dodawania przedmiotów do koszyka jeżeli koszyk ma być taki sam.