

Projekt zaliczeniowy

# Wunderwaffe model shop

Autorzy:  
Sebastian Kosmala  
Igor Kiełpiński

## Opis projektu

WUNDERWAFFE Model SHOP to pseudo sklep internetowy, który oferuje miniaturowe modele niemieckiego Wunderwaffe. Użytkownicy, zarówno zalogowani, jak i nie zalogowani, mogą kupować produkty. Zalogowani klienci mają dodatkowe opcje personalizacji, takie jak zakup modeli pomalowanych lub złożonych za dodatkową opłatą. Administrator zarządza użytkownikami oraz produktami na stronie.

Funkcjonalności strony obejmują:

- Zakup produktów dla zalogowanych użytkowników.
- Administrator może edytować konta użytkowników.
- Wyświetlanie najnowszych produktów na stronie głównej.
- Historię zakupów dla zalogowanych klientów.
- Dodawanie i edycja produktów przez administratora.
- Koszyk w którym można zwiększać ilość produktów lub je usunąć z koszyka.






## Specyfikacja technologii

- **Backend:** ASP.NET Core 8 aplikacja MVC
- **Baza danych:** SQL Server 2019, SQL Express
- **Język programowania:** C#
- **Frontend:** HTML, CSS, JavaScript

## Instrukcja pierwszego uruchomienia

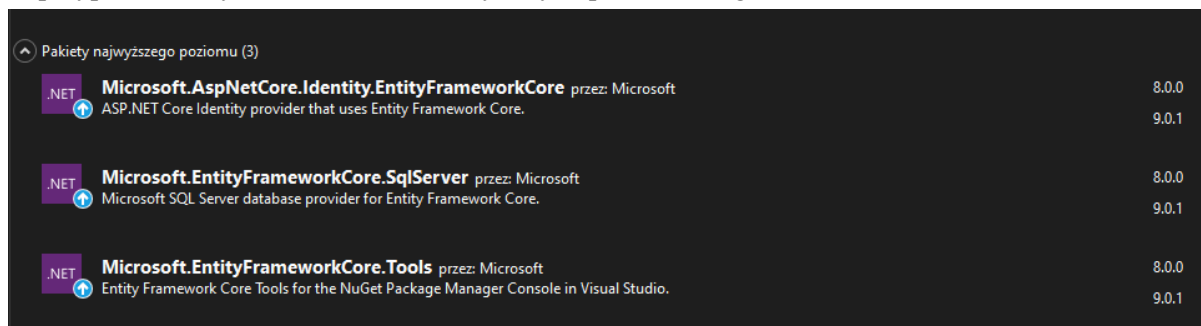
Najpierw należy pobrać cały projekt z githuba.

Aby uruchomić po raz pierwszy program należy zainstalować program Microsoft SQL Server 2019 Express(<https://www.microsoft.com/en-us/download/details.aspx?id=101064>). Po zainstalowaniu należy sprawdzić w konsoli cmd wpisując services.msc czy są te usługi

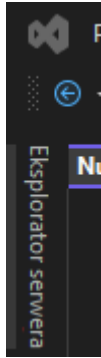
	SQL Server (SQLEXPRESS)	Provi...	Działa	Automatyczny	NT Service\MSS...
	SQL Server Agent (SQLEXPRESS)	Exec...		Wyłączony	Usługa sieciowa
	SQL Server Browser	Provi...		Wyłączony	Usługa lokalna
	SQL Server CEIP service (SQL...	CEIP ...	Działa	Automatyczny	NT Service\SQLT...
	SQL Server VSS Writer	Provi...	Działa	Automatyczny	System lokalny

W przypadku kiedy ich nie ma należy zainstalować jeszcze raz.

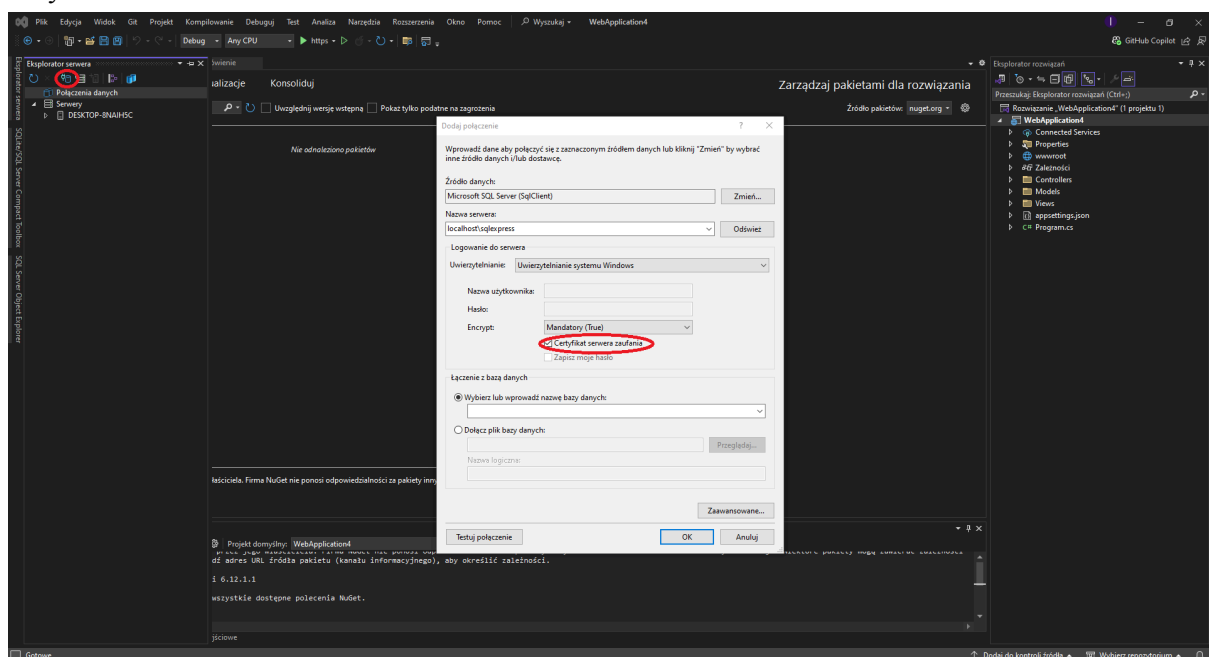
W przypadku kiedy nie ma zainstalowanych tych pakietów nuget w visual studio:



Należy je zainstalować.  
Następnie w eksploratorze serwerów



Należy nacisnąć łączenie z bazą danych w nazwie serwera wpisać localhost/sqlexpress i zaznaczyć certyfikat serwera zaufania.



W momencie kiedy wszystko już zainstalowane w konsoli pakietów nuget w visual studio należy wpisać Update-Database

## Opis struktury projektu

- **Kontrolery:** Kontrolery odpowiedzialne za obsługę żądań HTTP i zarządzanie widokami.
- **Modele:** Definicje danych, które są wykorzystywane w aplikacji (np. Product, User, Order).
- **Widoki:** Pliki HTML obsługujące interfejs użytkownika.

# Modele

## 1. ApplicationUser

**Opis:** Model zawiera dane o aktualnym użytkowniku. Wykorzystywane jest do tego aby w innych modelach nie powtarzać ciał

- FirstName (int): Imię.
- LastName (string): Nazwisko.
- Address (string): Adres zamieszkania.
- CreatedAt (DateTime): Data utworzenia konta.

## 2. CheckoutDto

**Opis:** Model służy do wykonania polecenia wysłania adresu i wyboru płatności.

- DeliveryAddress (string)[MaxLength(300)]: Adres wpisany przez użytkownika
- PaymentMethod (string): Email użytkownika.

## 3. ErrorViewModel

**Opis:** Automatycznie wygenerowany model przez asp.net do obsługi błędów.

- RequestId (string): Wymagany identyfikator
- ShowRequestId (ShowRequestedId): Żądany kod błędu

## 4. LoginDto

**Opis:** Model wykorzystywany podczas logowania.

- Email (string)[Required] - Adres e-mail osoby logującej się. Pole wymagane
- Password (string)[Required] - Hasło osoby logującej się. Pole wymagane
- RememberMe (string) - Pole wyboru odpowiedzialne za zapamiętanie użytkownika

## 5. Order

**Opis:** Model jest odpowiedzialny za zapisywanie szczegółów zamówienia wraz z jego statusem. Po wykonaniu zamówienia w bazie pojawia się zamówienie, które jest widoczne u administratora, który ręcznie aktualizuje status zamówienia. Wykorzystuje modele ApplicationUser oraz OrderItem.

- Id (int) - Numer zamówienia
- ClientId (string) - Numer klienta
- Client (ApplicationUser) - Dane użytkownika brane z modelu ApplicationUser
- Items (List<OrderItem>) - Przedmioty z koszyka są brane z modelu OrderItem
- ShippingFee (decimal) [Precision(16, 2)] - Cena za wysyłkę
- DeliveryAddress (string) - Adres wysyłki
- PaymentMethod (string) - Metoda płatności
- PaymentStatus (string) - Status płatności

- **PaymentDetails** (string) - Szczegóły zamówienia
- **OrderStatus** (string) - Status zamówienia
- **CreatedAt** (DateTime) - Data utworzenia

## 6. OrderItem

Opis: Model reprezentuje koszyk zakupowy

- **Id** (int) - Identyfikator koszyka.
- **Quantity** (int) - Ilość sztuk danego produktu w zamówieniu.
- **UnitPrice** (decimal)[Precision(16, 2)] - Cena produktu (lub łączna cena jednego produktu wybranego kilka razy) z dokładnością do dwóch miejsc po przecinku. Maksymalnie 16 cyfr w liczbie, w tym 2 po przecinku.
- **Product** (Product) - Produkt przypisany do zamówienia.

## 7. PasswordDto

Opis: Model wykorzystywany do zmiany hasła użytkownika. Przechowuje stare hasło, nowe hasło oraz jego potwierdzenie. Pokazuje komunikaty gdy wartości są niewłaściwe lub są brakujące

- **CurrentPassword** (string)[Required, MaxLength(100)] - Aktualne hasło użytkownika
- **NewPassword** (string)[Required, MaxLength(100)] - Nowe hasło
- **ConfirmPassword** (string)[Required, Compare("NewPassword")] - Pole do potwierdzenia nowego hasła, musiby być identyczne jak hasło powyżej

## 8. Product

Opis: Model odpowiedzialny za produkty na stronie. Dzięki niemu produkty są widoczne na stronie a także widoczne w koszyku.

- **Id** (int) -Identyfikator produktu.
- **Name** (string)[MaxLength(100)] - Nazwa produktu maksymalna do 100 znaków.
- **Category** (string)[MaxLength(100)] - Kategoria produktu maksymalna do 100 znaków.
- **Price** (decimal)[Precision(16, 2)] - Cena produktu z dokładnością do dwóch miejsc po przecinku. Maksymalnie 16 cyfr w liczbie, w tym 2 po przecinku.
- **Description** (string) - Opis produktu.
- **ImageFileName** (string)[MaxLength(100)] - Nazwa pliku obrazu produktudo maksymalnie 100 znaków.
- **CreatedAt** (DateTime) - Data utworzenia produktu.

## 9.ProductDto

Opis: Model wykorzystywany podczas przesyłania danych o produkcie w operacjach takich jak dodawanie lub edycja.

- **Name** (string)[Required, MaxLength(100)] - Nazwa produktu. Pole wymagane, maksymalnie 100 znaków.
- **Category** (string)[Required, MaxLength(100)] - Kategoria produktu. Pole wymagane, maksymalnie 100 znaków.

- **Price** (decimal)[Required] - Cena produktu. Pole wymagane.
- **Description** (string)[Required] - Opis produktu. Pole wymagane, bez ograniczeń długości.
- **ImageFile** (IFormFile?) - Plik obrazu produktu. Pole opcjonalne.

## 10. ProfileDto

**Opis:** Model wykorzystywany do przechowywania i edycji danych profilowych użytkownika.

- **FirstName** (string)[Required, MaxLength(100)] - Imię użytkownika. Pole wymagane, maksymalnie 100 znaków.
- **LastName** (string)[Required, MaxLength(100)] - Nazwisko użytkownika. Pole wymagane, maksymalnie 100 znaków.
- **Email** (string)[Required, EmailAddress, MaxLength(100)] - Adres e-mail użytkownika. Pole wymagane, musi mieć poprawny format e-mail, maksymalnie 100 znaków.
- **PhoneNumber** (string?)[Phone, MaxLength(11)] - Numer telefonu użytkownika. Pole opcjonalne, musi mieć poprawny format numeru telefonu, maksymalnie 11 znaków.
- **Address** (string)[Required, MaxLength(200)] - Adres użytkownika. Pole wymagane, maksymalnie 200 znaków.

•

## 11. RegisterDto

**Opis:** Model wykorzystywany podczas rejestracji nowego użytkownika.

- **FirstName** (string)[Required, MaxLength(100)] - Imię użytkownika. Pole wymagane, maksymalnie 100 znaków.
- **LastName** (string)[Required, MaxLength(100)] - Nazwisko użytkownika. Pole wymagane, maksymalnie 100 znaków.
- **Email** (string)[Required, EmailAddress, MaxLength(100)] - Adres e-mail użytkownika. Pole wymagane, musi mieć poprawny format e-mail, maksymalnie 100 znaków.
- **PhoneNumber** (string?)[Phone, MaxLength(11)] - Numer telefonu użytkownika. Pole opcjonalne, musi mieć poprawny format numeru telefonu, maksymalnie 11 znaków.
- **Address** (string)[Required, MaxLength(200)] - Adres użytkownika. Pole wymagane, maksymalnie 200 znaków.
- **Password** (string)[Required, MaxLength(100)] - Hasło użytkownika. Pole wymagane, maksymalnie 100 znaków.
- **ConfirmPassword** (string)[Required, Compare("Password")] - Potwierdzenie hasła użytkownika. Pole wymagane, musi być identyczne jak hasło.

## 12. StoreSearchModel

**Opis:** Model wykorzystywany do filtrowania, wyszukiwania i sortowania produktów w sklepie.

- **Search** (string?) - Frazę wyszukiwaną w nazwach lub opisach produktów. Pole opcjonalne.
- **Category** (string?) - Wybrana kategoria produktów (np. czołgi, rakiety). Pole opcjonalne.
- **Sort** (string?) - Kryterium sortowania (np. cena rosnąco, cena malejąco, najnowsze produkty). Pole opcjonalne.
-

# Kontrolery

## 1. AccountController

**Opis:** Kontroler odpowiadający za zarządzanie kontem użytkownika (rejestracja, logowanie, profil, zmiana hasła).

### 1. Register

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wyświetla widok rejestracji użytkownika.
- **Zwracane dane:** Widok formularza rejestracji.

### 2. Register (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - RegisterDto registerDto - Obiekt zawierający dane rejestracyjne użytkownika.
- **Opis:** Tworzy nowego użytkownika na podstawie przesłanych danych. Przypisuje rolę "Klient" i loguje użytkownika po pomyślnej rejestracji.
- **Zwracane dane:**
  - Przekierowanie na stronę główną (Index w HomeController) w przypadku sukcesu.
  - Widok rejestracji z komunikatami błędów w przypadku niepowodzenia.

### 3. Logout

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wylogowuje aktualnie zalogowanego użytkownika.
- **Zwracane dane:** Przekierowanie na stronę główną (Index w HomeController).

### 4. Login

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wyświetla widok logowania użytkownika.
- **Zwracane dane:** Widok formularza logowania.

### 5. Login (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - LoginDto loginDto - Obiekt zawierający dane logowania użytkownika.
- **Opis:** Uwierzytelnia użytkownika na podstawie danych logowania.
- **Zwracane dane:**
  - Przekierowanie na stronę główną (Index w HomeController) w przypadku sukcesu.
  - Widok logowania z komunikatami błędów w przypadku niepowodzenia.

## 6. ProfileAsync

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wyświetla widok profilu zalogowanego użytkownika z wypełnionymi danymi użytkownika.
- **Zwracane dane:** Widok z formularzem danych użytkownika.

## 7. Profile (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - ProfileDto profileDto - Obiekt zawierający dane profilu użytkownika.
- **Opis:** Aktualizuje dane profilu użytkownika.
- **Zwracane dane:**
  - Widok profilu z komunikatami sukcesu lub błędów.

## 8. Password

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wyświetla widok zmiany hasła.
- **Zwracane dane:** Widok formularza zmiany hasła.

## 9. Password (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - PasswordDto passwordDto - Obiekt zawierający dane do zmiany hasła.
- **Opis:** Zmienia hasło zalogowanego użytkownika.
- **Zwracane dane:**
  - Widok zmiany hasła z komunikatami sukcesu lub błędów.

## 10. AccessDenied

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Obsługuje sytuację, gdy użytkownik próbuje uzyskać dostęp do zabronionej akcji.
- **Zwracane dane:** Przekierowanie na stronę główną (Index w HomeController).

## 11. ForgotPassword

- **HTTP Methods:** GET
- **Parametry:** Brak
- **Opis:** Wyświetla widok umożliwiający rozpoczęcie procedury odzyskiwania hasła.
- **Zwracane dane:** Widok formularza odzyskiwania hasła.



## 12. ForgotPassword (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - [Required, EmailAddress] string email - Adres e-mail użytkownika.
- **Opis:** Obsługuje zgłoszenie odzyskiwania hasła i informuje użytkownika o wysłaniu wiadomości e-mail.
- **Zwracane dane:** Widok formularza z komunikatami.

## 2. AdminOrdersController

**Opis:** Kontroler odpowiadający za zarządzanie zamówieniami, dostępny wyłącznie dla użytkowników z rolą "admin".

### 1. Index

- **HTTP Methods:** GET
- **Parametry:**
  - int pageIndex - Numer strony (domyślnie 1, jeśli wartość  $\leq 0$ ).
- **Opis:** Wyświetla listę zamówień w systemie w sposób stronicowany.
- **Zwracane dane:**
  - Widok z listą zamówień (ViewBag.Orders).
  - Bieżący numer strony (ViewBag.PageIndex).
  - Łączna liczba stron (ViewBag.TotalPages).

### 2. Details

- **HTTP Methods:** GET
- **Parametry:**
  - int id - Identyfikator zamówienia.
- **Opis:** Wyświetla szczegóły wybranego zamówienia, w tym dane klienta i produkty w zamówieniu.
- **Zwracane dane:**
  - Widok ze szczegółami zamówienia.
  - Liczba zamówień klienta (ViewBag.NumOrders).
  - Przekierowanie na stronę główną zamówień w przypadku niezalezienia zamówienia.

### 3. Edit

- **HTTP Methods:** GET
- **Parametry:**
  - int id - Identyfikator zamówienia.
  - string? payment\_status - Nowy status płatności (opcjonalny).
  - string? order\_status - Nowy status zamówienia (opcjonalny).
- **Opis:** Aktualizuje status płatności i/lub status zamówienia. Jeśli żaden z parametrów nie zostanie przekazany, przekierowuje do szczegółów zamówienia.
- **Zwracane dane:**

- Przekierowanie do szczegółów zamówienia po udanej aktualizacji.
- Przekierowanie na stronę główną zamówień w przypadku niezalezienia zamówienia.

### 3. CartController

**Opis:** Kontroler odpowiedzialny za zarządzanie koszykiem użytkownika oraz procesem składania zamówienia.

#### 1. Index

- **HTTP Methods:** GET
- **Parametry:** Brak.
- **Opis:** Wyświetla zawartość koszyka użytkownika, w tym listę produktów, podsumowanie (kwotę zamówienia, opłatę za wysyłkę i całkowitą kwotę).
- **Zwracane dane:**
  - Widok koszyka z następującymi danymi w ViewBag:
    - CartItems: lista pozycji w koszyku.
    - ShippingFee: opłata za wysyłkę.
    - Subtotal: suma wartości zamówienia bez wysyłki.
    - Total: suma całkowita (wartość zamówienia + wysyłka).

#### 2. Confirm (GET)

- **HTTP Methods:** GET
- **Parametry:** Brak.
- **Opis:** Wyświetla podsumowanie zamówienia, w tym adres dostawy, metodę płatności i liczbę produktów w koszyku.
- **Zwracane dane:**
  - Widok podsumowania zamówienia z następującymi danymi w ViewBag:
    - DeliveryAddress: adres dostawy.
    - PaymentMethod: metoda płatności.
    - Total: całkowita wartość zamówienia.
    - CartSize: liczba produktów w koszyku.
  - Przekierowanie na stronę główną, jeśli koszyk jest pusty lub brakuje wymaganych danych (adresu, metody płatności).

#### 3. Confirm (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - int any: sztuczny parametr potrzebny do rozróżnienia akcji.
- **Opis:** Tworzy zamówienie na podstawie zawartości koszyka i danych użytkownika. Zapisuje zamówienie w bazie danych, czyści koszyk i wyświetla potwierdzenie.
- **Zwracane dane:**
  - Widok z potwierdzeniem zamówienia i komunikatem o sukcesie.
  - Przekierowanie na stronę główną w przypadku braku danych lub pustego koszyka.

## 4. Index (POST)

- **HTTP Methods:** POST
- **Parametry:**
  - CheckoutDto model: model zawierający dane do realizacji zamówienia (adres dostawy i metoda płatności).
- **Opis:** Waliduje dane użytkownika, zapisuje informacje o zamówieniu w TempData i przekierowuje użytkownika do odpowiedniego widoku w zależności od metody płatności (PayPal, karta kredytowa lub inne).
- **Zwracane dane:**
  - Widok koszyka z błędami walidacyjnymi, jeśli dane są niepoprawne.
  - Przekierowanie do widoku podsumowania lub do kontrolera Checkout w zależności od metody płatności.

## 4. ClientOrdersController

**Opis:** Kontroler umożliwiający klientom przeglądanie własnych zamówień.

### 1. Atrybuty

- **[Authorize(Roles = "klient")]:**
  - Ogranicza dostęp do kontrolera wyłącznie użytkownikom przypisanym do roli "klient".
- **[Route("/Client/Orders/{action=Index}/{id?}")]:**
  - Konfiguruje domyślną ścieżkę dla akcji kontrolera.

### 2. Konstruktor

- **Parametry:**
  - ApplicationDbContext context: kontekst bazy danych.
  - UserManager<ApplicationUser> userManager: zarządzanie użytkownikami.
- **Opis:** Inicjalizuje pola context i userManager.

### 3. Index

- **HTTP Methods:** GET
- **Parametry:**
  1. int pageIndex: indeks strony (domyślnie 1).
- **Opis:** Wyświetla listę zamówień użytkownika zalogowanego w systemie.
- **Działanie:**
  1. Pobiera zalogowanego użytkownika.
  2. Pobiera zamówienia przypisane do użytkownika, sortując je malejąco według ID.
  3. Oblicza paginację:
    - Liczba zamówień.
    - Liczba stron (na podstawie pageSize = 5).
    - Pobiera zamówienia dla bieżącej strony.
  4. Przekazuje dane do widoku:
    - Orders: lista zamówień.
    - pageIndex: bieżąca strona.

- TotalPages: liczba stron.
- **Zwracane dane:** Widok listy zamówień użytkownika.
- **Przekierowanie:** Do strony głównej, jeśli użytkownik nie jest zalogowany.

#### 4. Details

- **HTTP Methods:** GET
- **Parametry:**
  1. int id: identyfikator zamówienia.
- **Opis:** Wyświetla szczegóły zamówienia na podstawie identyfikatora.
- **Działanie:**
  1. Pobiera zalogowanego użytkownika.
  2. Wyszukuje zamówienie przypisane do użytkownika na podstawie id.
  3. Włącza dane produktów powiązanych z zamówieniem.
  4. Przekazuje zamówienie do widoku.
- **Zwracane dane:** Widok szczegółów zamówienia.
- **Przekierowanie:** Do listy zamówień, jeśli zamówienie nie istnieje lub użytkownik nie jest zalogowany.

#### Inne informacje

- **Paginacja w Index:**
  - pageSize: liczba zamówień na stronę (stała wartość = 5).
  - Automatyczne obliczanie stron na podstawie liczby zamówień użytkownika.
- **Bezpieczeństwo:**
  - Użycie Authorize zapewnia dostęp wyłącznie klientom.
  - Każda akcja sprawdza, czy użytkownik jest zalogowany, aby uniknąć nieautoryzowanego dostępu.

### 5. HomeController

**Opis:** Główny kontroler aplikacji odpowiedzialny za stronę główną oraz inne podstawowe widoki.

#### 1. Konstruktor

- **Parametry:**
  - ApplicationDbContext context: kontekst bazy danych.
- **Opis:** Inicjalizuje pole context, pozwalające na operacje na bazie danych.

#### 2. Akcje

##### Index

- **HTTP Methods:** GET
- **Opis:** Wyświetla stronę główną aplikacji, prezentując najnowsze produkty.
- **Działanie:**
  1. Pobiera z bazy danych cztery najnowsze produkty (sortowane malejąco po Id).
  2. Przekazuje produkty do widoku.
- **Zwracane dane:** Widok strony głównej z listą produktów.

### 3. Privacy

- **HTTP Methods:** GET
- **Opis:** Wyświetla stronę z polityką prywatności.
- **Działanie:** Renderuje statyczny widok Privacy.
- **Zwracane dane:** Widok polityki prywatności.

### 4. Error

- **HTTP Methods:** Dowolna (wywoływana automatycznie w przypadku błędów).
- **Atrybuty:**
  1. [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]: Wyłącza buforowanie odpowiedzi dla tego widoku.
- **Opis:** Wyświetla stronę błędu z informacjami diagnostycznymi.
- **Działanie:**
  1. Tworzy obiekt ErrorViewModel, wypełniając pole RequestId identyfikatorem bieżącego żądania.
  2. Przekazuje model do widoku.
- **Zwracane dane:** Widok strony błędu.

### 5. Inne informacje

- **Index:**
  - Pobiera maksymalnie 4 produkty z bazy danych, co pozwala ograniczyć ilość danych renderowanych na stronie głównej.
  - Sortowanie malejące po Id zakłada, że najnowsze produkty mają największe ID.
- **Error:**
  - Wykorzystywany do debugowania i obsługi błędów aplikacji.
  - Przechwytuje identyfikator żądania dla celów diagnostycznych.

## 6. ProductsController

Kontroler administracyjny odpowiedzialny za zarządzanie produktami w aplikacji. Obejmuje funkcjonalność tworzenia, edycji, usuwania oraz wyświetlania produktów.

#### Atrybuty kontrolera

- [Authorize(Roles = "admin")]  
Dostęp do akcji mają wyłącznie użytkownicy posiadający rolę admin.
- [Route("/Admin/{controller}/{action=Index}/{id?}")]  
Ścieżka URL dla akcji kontrolera to /Admin/Products/{action}/{id?}.

#### 1. Konstruktor

- **Parametry:**
  - ApplicationDbContext context: kontekst bazy danych.
  - IWebHostEnvironment environment: środowisko aplikacji (używane do operacji na plikach).
- **Opis:** Inicjalizuje zależności wymagane do operacji na bazie danych i plikach.

## 2. Akcje

### Index

- **HTTP Methods:** GET
- **Parametry:**
  1. int pageIndex: Numer bieżącej strony.
  2. string? search: Fraza do wyszukiwania produktów po nazwie.
  3. string? column: Kolumna do sortowania (Id, Name, Category, Price, CreatedAt).
  4. string? orderBy: Kolejność sortowania (asc, desc).
- **Opis:** Wyświetla listę produktów z możliwością paginacji, sortowania i wyszukiwania.
- **Działanie:**
  1. Filtruje produkty na podstawie frazy search (jeśli podano).
  2. Sortuje produkty na podstawie kolumny i kolejności.
  3. Implementuje paginację.
  4. Przekazuje dane do widoku.
- **Zwracane dane:** Widok listy produktów z metadanymi paginacji.

### 3. Create

- **HTTP Methods:**
  - GET: Wyświetla formularz tworzenia produktu.
  - POST: Obsługuje zapis nowego produktu.
- **Parametry (POST):**
  - ProductDto productDto: Dane produktu wprowadzone przez użytkownika.
- **Opis:** Tworzy nowy produkt i zapisuje go w bazie danych.
- **Działanie (POST):**
  - Waliduje formularz, w tym obecność pliku obrazu.
  - Zapisuje plik obrazu na serwerze.
  - Tworzy i zapisuje nowy obiekt Product w bazie danych.
- **Zwracane dane:**
  - GET: Formularz tworzenia produktu.
  - POST: Przekierowanie do listy produktów.

### 4. Edit

- **HTTP Methods:**
  - GET: Wyświetla formularz edycji produktu.
  - POST: Obsługuje zapis zmian w produkcie.
- **Parametry:**
  - int id: ID edytowanego produktu.
  - ProductDto productDto: Zaktualizowane dane produktu.
- **Opis:** Edytuje istniejący produkt w bazie danych.
- **Działanie (POST):**
  - Znajduje produkt w bazie danych.
  - Aktualizuje dane produktu oraz zapisuje nowy obraz, jeśli dostarczono.
  - Usuwa stary obraz, jeśli został zaktualizowany.
- **Zwracane dane:**
  - GET: Formularz edycji produktu.

- POST: Przekierowanie do listy produktów.

## 5. Delete

- **HTTP Methods:** GET
- **Parametry:**
  1. int id: ID usuwanego produktu.
- **Opis:** Usuwa produkt z bazy danych.
- **Działanie:**
  1. Znajduje produkt w bazie danych.
  2. Usuwa produkt.
- **Zwracane dane:** Przekierowanie do listy produktów.

## 7. StoreController

Kontroler odpowiedzialny za obsługę widoków sklepu w aplikacji. Udostępnia funkcje przeglądania listy produktów, filtrowania, sortowania oraz wyświetlania szczegółów wybranego produktu.

### 1. Konstruktor

- **Parametry:**
  - ApplicationDbContext context: Kontekst bazy danych.
- **Opis:** Inicjalizuje zależności wymagane do operacji na bazie danych.

### 2. Akcje

#### Index

- **HTTP Methods:** GET
- **Parametry:**
  1. int pageIndex: Numer bieżącej strony (paginacja).
  2. string? search: Fraza wyszukiwania produktów (szukanie po nazwie).
  3. string? brand: Filtr produktów po marce (obecnie niewykorzystane w kodzie).
  4. string? category: Filtr produktów po kategorii.
  5. string? sort: Parametr określający sposób sortowania (price\_asc, price\_desc).
- **Opis:** Wyświetla listę produktów z funkcjami filtrowania, sortowania i paginacji.
- **Działanie:**
  1. Filtruje produkty na podstawie:
    - Frazy wyszukiwania (search).
    - Kategorii (category).
  2. Sortuje produkty:
    - price\_asc: Cena rosnąco.
    - price\_desc: Cena malejąco.
    - Domyślnie: Po ID malejąco.
  3. Implementuje paginację przy użyciu parametrów pageIndex i pageSize.
  4. Przygotowuje dane do widoku:
    - Produkty (ViewBag.Products).
    - Informacje o paginacji (ViewBag.PageIndex, ViewBag.TotalPages).
    - Model wyszukiwania i filtrów (StoreSearchModel).

- **Zwracane dane:** Widok listy produktów z uwzględnieniem wyszukiwania, sortowania i paginacji.

#### Details

- **HTTP Methods:** GET
- **Parametry:**
  1. int id: ID produktu, którego szczegóły mają zostać wyświetlone.
- **Opis:** Wyświetla szczegóły wybranego produktu.
- **Działanie:**
  1. Wyszukuje produkt w bazie danych na podstawie ID.
  2. Jeśli produkt nie istnieje, przekierowuje do listy produktów (Index).
  3. Jeśli produkt istnieje, przekazuje go do widoku.
- **Zwracane dane:** Widok szczegółów produktu.

## 8. UsersController

Kontroler odpowiedzialny za zarządzanie użytkownikami w aplikacji. Udostępnia funkcje przeglądania listy użytkowników, szczegółów kont, zmiany ról oraz usuwania kont.

### 1. Konstruktor

- **Parametry:**
  - UserManager<ApplicationUser> userManager: Zarządzanie użytkownikami.
  - RoleManager<IdentityRole> roleManager: Zarządzanie rolami użytkowników.
- **Opis:** Inicjalizuje wymagane usługi do operacji na użytkownikach i rolach.

### 2. Akcje

#### Index

- **HTTP Methods:** GET
- **Parametry:**
  1. int? pageIndex: Numer bieżącej strony (paginacja).
- **Opis:** Wyświetla listę użytkowników z funkcją paginacji.
- **Działanie:**
  1. Pobiera użytkowników z bazy danych, sortując ich według daty utworzenia (CreatedAt) malejąco.
  2. Implementuje paginację przy użyciu pageIndex i pageSize.
  3. Przygotowuje dane do widoku:
    - Użytkownicy (users).
    - Informacje o paginacji (ViewBag.PageIndex, ViewBag.TotalPages).
- **Zwracane dane:** Widok listy użytkowników.



## Details

- **HTTP Methods:** GET
- **Parametry:**
  1. string? id: ID użytkownika, którego szczegóły mają zostać wyświetlone.
- **Opis:** Wyświetla szczegóły konta użytkownika, w tym jego rolę.
- **Działanie:**
  1. Wyszukuje użytkownika w bazie danych na podstawie ID.
  2. Jeśli użytkownik nie istnieje, przekierowuje do strony Index.
  3. Pobiera przypisane role użytkownika oraz listę wszystkich dostępnych ról.
  4. Tworzy listę wyboru (SelectListItem) z zaznaczeniem przypisanych ról.
- **Zwracane dane:** Widok szczegółów użytkownika.

## EditRole

- **HTTP Methods:** POST
- **Parametry:**
  1. string? id: ID użytkownika, którego rola ma zostać zmieniona.
  2. string? newRole: Nowa rola, która ma zostać przypisana.
- **Opis:** Zmienia rolę użytkownika na nową.
- **Działanie:**
  1. Weryfikuje poprawność danych wejściowych (id, newRole).
  2. Sprawdza, czy użytkownik i rola istnieją.
  3. Użytkownik nie może zmienić własnej roli.
  4. Usuwa bieżące role użytkownika i przypisuje nową rolę.
  5. Informuje o sukcesie lub błędzie operacji.
- **Zwracane dane:** Przekierowanie do szczegółów użytkownika.

## DeleteAccount

- **HTTP Methods:** POST
- **Parametry:**
  1. string? id: ID użytkownika, który ma zostać usunięty.
- **Opis:** Usuwa konto użytkownika.
- **Działanie:**
  1. Weryfikuje poprawność ID.
  2. Wyszukuje użytkownika w bazie danych.
  3. Użytkownik nie może usunąć własnego konta.
  4. Próbuje usunąć użytkownika i informuje o wyniku operacji.
- **Zwracane dane:** Przekierowanie do listy użytkowników lub szczegółów użytkownika.

## System użytkowników

- **Administrator** - Operuje całą witryną Dodając usuwając modyfikując produkty na stronie. Ma podgląd na wszystkie złożone zamówienia przez wszystkich użytkowników jak i może ustawić czy zamówienie zostało dostarczone lub wysłane, również ustawia w zamówieniu stan płatności. Administrator również może nadawać innym użytkownikom rolę administratora.
- **Klient** - Na stronie ma możliwość przeglądania produktów, kupna ich i przeglądania własnych zamówień. Również jest w stanie zobaczyć szczegóły danych produktów.

## Ciekawe funkcjonalności

- **System sortowania:** System sortowania produktów pod względem najniższej do najwyższej ceny i odwrotnie jak i sortowanie względem daty dodania lub kategorii .
- **Wyszukiwarka produktów:** Pozwala wyszukiwać produktów po nazwie
- **Produkty na stronie głównej:** Wyświetlanie najnowszych produktów na stronie głównej
- **Podstrony produktów, użytkowników :** Strony są przewijane dzięki czemu strona szybciej się ładuje z racji zmniejszonej ilości produktów która musi się załadować