TechStore System

1.0

Wygenerowano za pomocą Doxygen 1.16.1

Rozdział 1

CHANGELOG

This file explains how Visual Studio created the project.

The following steps were used to generate this project:

- Create new ASP.NET Core Web API project.
- Update project file to add a reference to the frontend project and set SPA properties.
- Update `launchSettings.json` to register the SPA proxy as a startup assembly.
- Add `dockerfile` to set up docker build.
- Add project to the startup projects list.
- Write this file.

Rozdział 2

Instrukcja Konfiguracji Projektu (Docker & Migracje)

Ten dokument opisuje standardowe procedury pracy z projektem, w tym tworzenie nowych migracji bazy danych oraz codzienne uruchamianie środowiska deweloperskiego.

2.1 ## Jak Dodać Nową Migrację (np. dodać kolumnę do tabeli)

To jest proces, który wykonujesz **TY**, gdy zmieniasz strukturę bazy danych (np. dodajesz nową tabelę lub kolumnę).

1. **Zmień Modele C#:** Zanim zaczniesz, dokonaj zmian w swoich modelach C# (np. dodaj `public string NewProperty { get; set; }` do klasy `Product.cs`).
2. **Uruchom Kontenery:** W Visual Studio, upewnij się, że jako "Startup Project" wybrany jest `docker-compose`. Naciśnij **`Ctrl + F5`** (Start Without Debugging), aby uruchomić kontenery (backend i bazę danych).
3. **Zmień Connection String (Tymczasowo):** Otwórz plik `appsettings.json` (w projekcie `.Server`). Zmień `Server=db` na `Server=localhost`.
   - *(To jest "most", aby Visual Studio mogło zobaczyć bazę danych, która działa w Dockerze).*
4. **Zmień Ustawienia Projektu:**
   - W Visual Studio, zmień "Startup Project" (na górnym pasku obok "Play") na `Sklep_internetowy.Server`.
   - W konsoli `Package Manager Console` (na dole), w polu `Default project`, również wybierz `Sklep_internetowy.Server`.
5. **Dodaj Migrację:** W `Package Manager Console` wpisz: `powershell Add-Migration TwojaNazwaMigracji (np. AddIsAdminToUser)`
6. **Zaktualizuj Bazę Danych:** W `Package Manager Console` wpisz: `powershell Update-↩ Database` *(Poczekaj na komunikat `Done.`)*
7. ** PRZYWRÓĆ USTAWIENIA! (Najważniejsze):**
   - Wróć do `appsettings.json` i zmień `Server=localhost` z powrotem na `Server=db`.
   - Wróć do "Startup Project" (na górnym pasku) i zmień go z powrotem na `docker-compose`.
8. **Zrób Commit:** Teraz zrób `git commit` i `git push`. Twoi koledzy z zespołu otrzymają nowy plik migracji, który będą musieli tylko u siebie zastosować (patrz następna sekcja).

2.2 ## Jak Zastosować Nową Migrację (dla reszty zespołu)

To jest proces, który wykonuje **KAŻDY** członek zespołu, gdy pobierze z GitHuba nową migrację (stworzoną przez kogoś innego).

1. **Zrób `git pull`**, aby pobrać najnowsze zmiany (w tym nowy plik migracji).
2. Wykonaj kroki **2, 3, 4** oraz **6** z instrukcji "Jak Dodać Nową Migrację".
3. **Nie musisz** uruchamiać `Add-Migration`. Musisz tylko uruchomić `Update-Database`.
4. Pamiętaj, aby na końcu **przywrócić ustawienia** (krok 7).

2.3 ## Jak Uruchomić Cały Projekt (Backend + Baza + Frontend)

To jest proces, który wykonujesz **codziennie**, aby uruchomić projekt do pracy.

2.3.0.1 1. Uruchom Backend i Bazę Danych:

1. W Visual Studio upewnij się, że "Startup Project" (na górze) to `docker-compose`.
2. Naciśnij **`Ctrl + F5`** (Start Without Debugging).
3. Otworzy się przeglądarka ze Swaggerem (np. `localhost:8080`). **Zostaw to włączone.**

2.3.0.2 2. Uruchom Frontend:

1. Otwórz **nowy, osobny** terminal (np. `View -> Terminal` w Visual Studio).
2. Jeśli używasz **PowerShell**, wpisz tę komendę, aby zezwolić na skrypty (musisz to robić za każdym razem, gdy otwierasz nowy terminal): `powershell Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass`
3. Upewnij się, że jesteś w głównym folderze projektu (np. `D:\ProjectZespolowy\Sklep_internetowy`, tam gdzie jest `package.json`).
4. Wpisz komendę: `bash npm run dev`
5. Otwórz w przeglądarce adres, który poda Ci Vite (np. `http://localhost:5173`).

Rozdział 3

Sklep_internetowy

Rozdział 4

Indeks przestrzeni nazw

4.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie udokumentowane przestrzenie nazw wraz z ich krótkimi opisami:

Rozdział 5

Indeks klas

5.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 6

Indeks plików

6.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików wraz z ich krótkimi opisami:

Rozdział 7

Dokumentacja przestrzeni nazw

7.1 Dokumentacja przestrzeni nazw Sklep_internetowy

7.2 Dokumentacja przestrzeni nazw Sklep_internetowy.Server

7.3 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers

Komponenty

- class BidController

  Kontroler API odpowiedzialny za obsluge systemu aukcyjnego oraz licytacji. Umozliwia przegladanie aktywnych aukcji, skladanie ofert oraz wystawianie nowych przedmiotow na licytacje.
- class UsersController

  Kontroler API odpowiedzialny za udostepnianie informacji o uzytkownikach systemu. Umozliwia bezpieczne pobieranie danych profilowych przeksztalconych do formatu obiektow transferu danych (DTO).

7.4 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Account

Komponenty

- class AccountController

  Kontroler API odpowiedzialny za zarzadzanie danymi konta zalogowanego uzytkownika. Wymaga autoryzacji przy uzyciu schematu Bearer (token JWT).

7.5 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Admin

Komponenty

- class ProductCategoryController

  Kontroler API odpowiedzialny za zarzadzanie kategoriami produktow oraz filtrowanie ofert specjalnych. Umozliwia pobieranie list kategorii, szczegolowych danych o kategorii oraz powiazanych z nimi produktow.
- class ProductController

  Kontroler administracyjny odpowiedzialny za pelne zarzadzanie asortymentem produktow. Obsluguje operacje CRUD, przesyłanie plikow graficznych, wyszukiwanie oraz system sugestii.
- class PromotionController

  Kontroler administracyjny odpowiedzialny za zarzadzanie kampaniami promocyjnymi. Umozliwia identyfikacje produktow kwalifikujacych sie do obnizek oraz masowe nakladanie rabatow.

- class ApplyBulkDiscountRequest

  Klasa pomocnicza (DTO) reprezentujaca zadanie masowego nalozenia rabatow.
- class UserMessageController

  Kontroler API odpowiedzialny za zarzadzanie wiadomosciami kontaktowymi uzytkownikow. Umozliwia przegladanie, wyszukiwanie, tworzenie oraz edycje zgloszen w panelu administracyjnym.
- class UsersController

  Kontroler administracyjny odpowiedzialny za zarzadzanie kontami uzytkownikow i ich uprawnieniami (rolami). Umozliwia pelny cykl zycia uzytkownika, w tym tworzenie, edycje, usuwanie oraz resetowanie hasel.

7.6 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Auth

Komponenty

- class AuthController

  Kontroler API odpowiedzialny za procesy uwierzytelniania i autoryzacji uzytkownikow. Obsluguje operacje rejestracji nowych kont oraz logowania (generowania tokenow JWT).

7.7 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Message

Komponenty

- class MessageController

  Kontroler API odpowiedzialny za obsluge wiadomosci kontaktowych przesylanych przez uzytkownikow. Modul umozliwia zapisywanie nowych zgloszen oraz ich odczyt w panelu administracyjnym.

7.8 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Payment

Komponenty

- class PaymentController

  Kontroler API odpowiedzialny za integracje z systemem platnosci Stripe. Umozliwia generowanie zamiarow platnosci (PaymentIntents) dla transakcji elektronicznych.
- class PaymentRequest

  Klasa pomocnicza (DTO) reprezentujaca zadanie utworzenia platnosci.

7.9 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Controllers.Shop

Komponenty

- class ProductController

  Kontroler API odpowiedzialny za dostarczanie danych o produktach dla czesci sklepowej (ogolnodostepnej). Obsluguje operacje przegladania katalogu produktow oraz wyswietlania szczegolowych informacji o konkretnym towarze.

7.10 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Data

Komponenty

- class StoreDbContext

  Glowny kontekst bazy danych aplikacji TechStore. Klasa dziedziczy po IdentityDbContext, co integruje system ASP.↩ NET Core Identity z modelem danych aplikacji, umozliwiajac zarzadzanie uzytkownikami i rolami.

7.11 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.DTOs

Komponenty

- class AuctionDto

  Obiekt transferu danych (DTO) reprezentujący podstawowe informacje o aukcji. Klasa służy do przesyłania kluczowych parametrów licytacji pomiędzy warstwą serwerową a interfejsem użytkownika, minimalizując narzut sieciowy poprzez ograniczenie przesyłanych danych.
- class BidRequest

  Obiekt transferu danych (DTO) reprezentujący żądanie złożenia nowej oferty licytacyjnej. Przechowuje informacje o deklarowanej kwocie oraz tożsamości licytanta niezbędne do przetworzenia postąpienia przez silnik aukcyjny.
- class CreateAuctionDto

  Obiekt transferu danych (DTO) wykorzystywany do zainicjowania nowej aukcji w systemie. Zawiera niezbędne informacje o produkcie przeznaczonym do sprzedaży licytacyjnej oraz ustalonej dla niego cenie wywoławczej.
- class CreateOrderItemDto

  Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję w żądaniu utworzenia zamówienia. Zawiera informacje niezbędne do zidentyfikowania produktu oraz określenia zamawianej ilości sztuk.
- class CreateOrderRequestDto

  Obiekt transferu danych (DTO) reprezentujący kompletne żądanie utworzenia nowego zamówienia. Przesyłany z warstwy frontendu podczas finalizacji koszyka zakupowego w celu zainicjowania transakcji po stronie serwerowej.
- class CreateProductDto

  Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu do katalogu sklepu. Zawiera kompletny zestaw informacji o towarze, jego parametrach cenowych oraz logice promocyjnej.
- class CreateProductWithFilesDto

  Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu wraz z załącznikami multimedialnymi. Klasa obsługuje dane przesyłane w formacie multipart/form-data, umożliwiając jednoczesną definicję parametrów handlowych oraz przesyłanie plików graficznych do serwera.
- class LoginRequest

  Obiekt transferu danych (DTO) wykorzystywany podczas procesu uwierzytelniania użytkownika. Przechowuje poświadczenia (login i hasło) niezbędne do weryfikacji tożsamości i wygenerowania tokenu autoryzacyjnego w systemie TechStore.
- class OrderDetailsDto

  Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o zamówieniu klienta. Klasa agreguje dane identyfikacyjne użytkownika, aktualny status realizacji zamówienia oraz pełną listę produktów wraz z datą operacji, służąc do prezentacji danych w interfejsie użytkownika.
- class OrderProductDetailsDto

  Obiekt transferu danych (DTO) zawierający szczegółowe informacje o konkretnym produkcie przypisanym do zamówienia. Przechowuje dane o wolumenie zakupu, stanach magazynowych oraz historycznej cenie sprzedaży.
- class OrderUpdateItemDto

  Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję towarową w żądaniu aktualizacji zamówienia. Przechowuje informacje niezbędne do skorygowania ilości konkretnego produktu w ramach istniejącej transakcji.
- class OrderUpdateDto

  Obiekt transferu danych (DTO) wykorzystywany do kompleksowej modyfikacji istniejącego zamówienia. Klasa umożliwia jednoczesną zmianę statusu logistycznego zamówienia oraz aktualizację zestawienia produktów wchodzących w jego skład.
- class PlaceBidDto

  Obiekt transferu danych (DTO) reprezentujący ofertę złożoną przez użytkownika w ramach aukcji. Przechowuje informację o kwocie postąpienia, która jest przesyłana do serwera w celu walidacji i aktualizacji bieżącej ceny licytowanego przedmiotu.
- class ProductCategoryDto

  Obiekt transferu danych (DTO) reprezentujący kategorię produktów w systemie. Służy do przesyłania podstawowych informacji o grupach towarowych, wykorzystywanych w menu nawigacyjnym, filtrach oraz przy przypisywaniu produktów do katalogu.
- class ProductDto

  Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o produkcie. Zawiera kompletny zestaw danych o towarze, w tym parametry cenowe, logikę promocyjną, przynależność do kategorii oraz powiązane zasoby multimedialne.

- class RegisterUserRequest

  Obiekt transferu danych (DTO) reprezentujący żądanie rejestracji nowego użytkownika w systemie TechStore. Klasa służy do przesyłania kompletu informacji niezbędnych do zainicjowania procesu tworzenia konta użytkownika, obejmując dane identyfikacyjne, uwierzytelniające oraz definicję ról dostępu.
- class RemoveProductDto

  Obiekt transferu danych (DTO) wykorzystywany w procesie usuwania produktu z systemu. Zawiera minimalny zestaw danych niezbędny do jednoznacznej identyfikacji zasobu przeznaczonego do usunięcia z bazy danych.
- class UpdateProductDto

  Obiekt transferu danych (DTO) służący do aktualizacji parametrów istniejącego produktu. Klasa zawiera kompletny zestaw właściwości niezbędnych do modyfikacji danych technicznych, logiki cenowej oraz przypisań kategoryzacyjnych towaru w systemie TechStore.
- class UpdateUserRequest

  Obiekt transferu danych (DTO) służący do aktualizacji podstawowych informacji o koncie użytkownika. Klasa ta jest wykorzystywana w procesie modyfikacji danych profilowych, umożliwiając synchronizację zmian nazwy użytkownika oraz adresu e-mail pomiędzy warstwą prezentacji a bazą danych.
- class UserDto

  Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o profilu użytkownika. Klasa agreguje dane identyfikacyjne, personalne oraz przypisane role systemowe, służąc do bezpiecznego przesyłania informacji o koncie z warstwy serwerowej do interfejsu klienta.
- class UserUpdateDto

  Obiekt transferu danych (DTO) wykorzystywany przez administratora do aktualizacji kluczowych parametrów konta użytkownika. Klasa umożliwia synchronizację zmian w adresie e-mail oraz modyfikację zestawu ról przypisanych do tożsamości użytkownika.

7.12 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Migrations

Komponenty

- class InitialCreate

7.13 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Models

Komponenty

- class Auction

  Klasa reprezentująca aukcję (licytację) konkretnego produktu w systemie. Przechowuje informacje o cenach, czasie trwania oraz uczestnikach biorących udział w licytacji.
- class Bid

  Klasa reprezentująca pojedynczą ofertę licytacji (postąpienie) złożoną przez użytkownika. Przechowuje informacje o kwocie, czasie złożenia oraz powiązaniach z konkretną aukcją i licytantem.
- class MailSettings

  Klasa konfiguracyjna przechowująca parametry serwera poczty elektronicznej (SMTP). Wykorzystywana przez serwis e-mail do autoryzacji i wysyłania powiadomień systemowych.
- class Order

  Klasa reprezentująca zamówienie klienta w systemie TechStore. Przechowuje informacje o dacie zakupu, statusie oraz powiązaniach z użytkownikiem i produktami.
- class OrderProduct

  Klasa pośrednicząca (tabela łącząca) reprezentująca relację wiele-do-wielu pomiędzy zamówieniami a produktami. Przechowuje informacje o tym, jakie produkty i w jakiej ilości wchodzą w skład danego zamówienia.
- class Product

  Klasa reprezentująca produkt w systemie sklepu internetowego. Zawiera szczegółowe informacje o towarze, stanach magazynowych oraz logikę zarządzania cenami promocyjnymi i powiązaniami z kategoriami.
- class ProductCategory

Klasa reprezentująca kategorię produktów w systemie TechStore. Służy do logicznego grupowania towarów, co ułatwia zarządzanie asortymentem oraz nawigację użytkownika w części frontendowej sklepu.

- class ProductImage

  Klasa reprezentująca zdjęcie przypisane do konkretnego produktu w systemie TechStore. Przechowuje informacje o lokalizacji pliku graficznego oraz relację z produktem, co umożliwia renderowanie galerii zdjęć w części frontendowej sklepu.
- class User

  Rozszerzona klasa użytkownika systemu, dziedzicząca po IdentityUser. Przechowuje dodatkowe dane profilowe, takie jak imię i nazwisko, oraz zarządza relacją z historią zamówień klienta.
- class UserMessage

  Klasa reprezentująca wiadomość przesłaną przez użytkownika poprzez formularz kontaktowy. Przechowuje dane teleadresowe nadawcy oraz treść zapytania, umożliwiając obsługę zgłoszeń klientów.

7.14 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Services

Komponenty

- class EmailService

  Serwis odpowiedzialny za wysyłanie wiadomości e-mail.

7.15 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Services.Auth

Komponenty

- class AccountService

  Serwis odpowiedzialny za obsługę kont użytkowników.
- class AuthExtensions

  Klasa rozszerzeń konfiguracji uwierzytelniania JWT.
- class AuthSettings

  Klasa przechowująca ustawienia uwierzytelniania JWT.
- class JwtService

  Serwis odpowiedzialny za generowanie tokenów JWT dla użytkowników.

7.16 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Services.Bidding

Komponenty

- class AuctionBackgroundService

  Usługa działająca w tle odpowiedzialna za automatyczne kończenie aukcji.
- class AuctionHub

  Hub SignalR odpowiedzialny za komunikację w czasie rzeczywistym dla aukcji.
- class AuctionService

  Serwis odpowiedzialny za pełną obsługę logiki aukcji.

7.17 Dokumentacja przestrzeni nazw Sklep_internetowy.Server.Services.Promotion

Komponenty

- class PromotionService

Serwis biznesowy odpowiedzialny za zaawansowane zarządzanie akcjami promocyjnymi w systemie TechStore. Klasa dostarcza metody do identyfikacji produktów o niskiej rotacji, masowego nakładania rabatów oraz automatycznego czyszczenia bazy danych z wygasłych ofert cenowych.

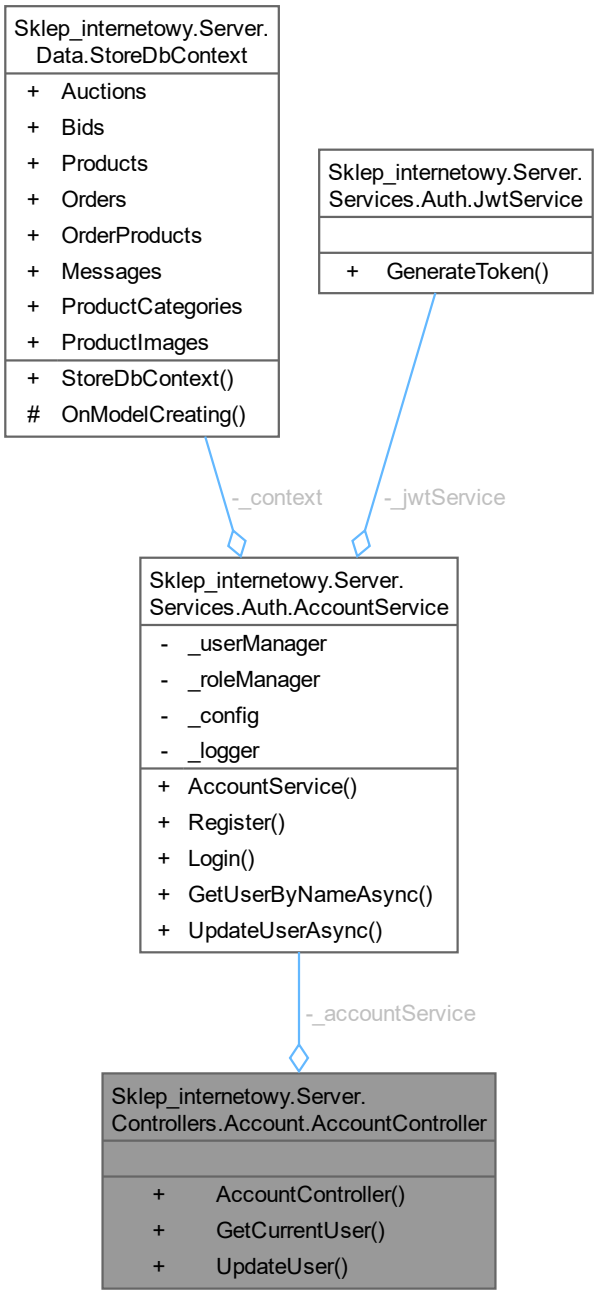
Rozdział 8

Dokumentacja klas

8.1 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Account.AccountController

Kontroler API odpowiedzialny za zarzadzanie danymi konta zalogowanego uzytkownika. Wymaga autoryzacji przy uzyciu schematu Bearer (token JWT).

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Account.AccountController:



Metody publiczne

- AccountController (AccountService accountService)

  Inicjalizuje nowa instancje klasy AccountController.
- async Task< IActionResult > GetCurrentUser ()

  Pobiera szczegolowe dane profilowe aktualnie uwierzytelnionego uzytkownika.
- async Task< IActionResult > UpdateUser ([FromBody] UpdateUserRequest request)

  Aktualizuje informacje o profilu zalogowanego uzytkownika (np. zmiana adresu e-mail).

8.1.1 Opis szczegółowy

Kontroler API odpowiedzialny za zarzadzanie danymi konta zalogowanego uzytkownika. Wymaga autoryzacji przy uzyciu schematu Bearer (token JWT).

8.1.2 Dokumentacja konstruktora i destruktora

8.1.2.1 AccountController()

```
Sklep_internetowy.Server.Controllers.Account.AccountController.AccountController (
            AccountService accountService)  [inline]
```

Inicjalizuje nowa instancje klasy AccountController.

Parametry

accountService	Serwis obslugujacy logike biznesowa kont uzytkownikow.

8.1.3 Dokumentacja funkcji składowych

8.1.3.1 GetCurrentUser()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Account.AccountController.↩
GetCurrentUser ()  [inline]
```

Pobiera szczegolowe dane profilowe aktualnie uwierzytelnionego uzytkownika.

Zwraca

Obiekt zawierajacy nazwe uzytkownika oraz adres e-mail.

<response code="200">Zwraca dane profilowe uzytkownika.</response> <response code="401">Gdy struktura tokenu jest nieprawidlowa lub uzytkownik nie jest zalogowany.</response> <response code="404">Gdy uzytkownik nie figuruje w bazie danych.</response>

8.1.3.2 UpdateUser()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Account.AccountController.↩
UpdateUser (
            [FromBody] UpdateUserRequest request)  [inline]
```

Aktualizuje informacje o profilu zalogowanego uzytkownika (np. zmiana adresu e-mail).

Parametry

request	Obiekt DTO zawierajacy nowe dane do zapisu.

Zwraca

Komunikat o powodzeniu operacji lub opis bledu.

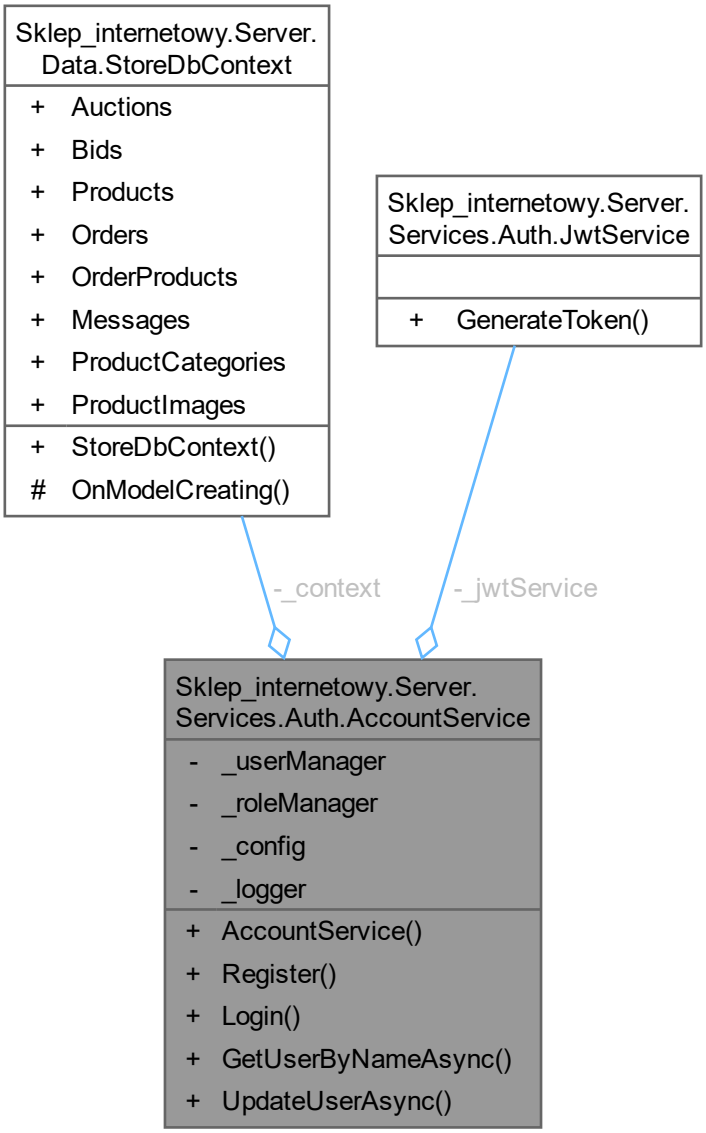
<response code="200">Profil zostal pomyslnie zaktualizowany.</response> <response code="400">Gdy format zadania jest bledny lub wystapil blad podczas zapisu danych.</response> <response code="401">Gdy uzytkownik nie posiada uprawnien do wykonania akcji.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Account/AccountController.cs

8.2 Dokumentacja klasy Sklep_internetowy.Server.Services.Auth.AccountService

Serwis odpowiedzialny za obsługę kont użytkowników.

Diagram współpracy dla Sklep_internetowy.Server.Services.Auth.AccountService:



Metody publiczne

- AccountService (UserManager< User > userManager, RoleManager< IdentityRole > roleManager, JwtService jwtService, IConfiguration config, ILogger< AccountService > logger)

  Konstruktor serwisu AccountService.
- async Task Register (string username, string email, string password)

  Rejestruje nowego użytkownika w systemie.
- async Task<(string Token, User User)> Login (string username, string password)

  Loguje użytkownika do systemu.
- async Task< User > GetUserByNameAsync (string username)

  Pobiera użytkownika po nazwie.

- async Task UpdateUserAsync (string currentUsername, UpdateUserRequest request)

  Aktualizuje dane użytkownika.

Atrybuty prywatne

- readonly UserManager< User > **_userManager**

  Menedżer użytkowników ASP.NET Identity.
- readonly RoleManager< IdentityRole > **_roleManager**

  Menedżer ról ASP.NET Identity.
- readonly StoreDbContext **_context**

  Kontekst bazy danych.
- readonly JwtService **_jwtService**

  Serwis generowania tokenów JWT.
- readonly IConfiguration **_config**

  Konfiguracja aplikacji.
- readonly ILogger **_logger**

  Logger serwisu.

8.2.1 Opis szczegółowy

Serwis odpowiedzialny za obsługę kont użytkowników.
Realizuje rejestrację, logowanie, pobieranie oraz aktualizację danych użytkownika.

8.2.2 Dokumentacja konstruktora i destruktora

8.2.2.1 AccountService()

```
Sklep_internetowy.Server.Services.Auth.AccountService.AccountService (
            UserManager< User > userManager,
            RoleManager< IdentityRole > roleManager,
            JwtService jwtService,
            IConfiguration config,
            ILogger< AccountService > logger)  [inline]
```

Konstruktor serwisu AccountService.

Parametry

userManager	Menedżer użytkowników.
roleManager	Menedżer ról.
jwtService	Serwis JWT.
config	Konfiguracja aplikacji.
logger	Logger.

8.2.3 Dokumentacja funkcji składowych

8.2.3.1 GetUserByNameAsync()

```
async Task< User > Sklep_internetowy.Server.Services.Auth.AccountService.GetUserByNameAsync (
            string username)  [inline]
```

Pobiera użytkownika po nazwie.

Parametry

username	Nazwa użytkownika.

Zwraca

Obiekt użytkownika lub null jeśli nie istnieje.

8.2.3.2 Login()

```
async Task<(string Token, User User)> Sklep_internetowy.Server.Services.Auth.AccountService.↩
Login (
            string username,
            string password)  [inline]
```

Loguje użytkownika do systemu.

Parametry

username	Nazwa użytkownika.
password	Hasło.

Zwraca

Krotka zawierająca token JWT oraz obiekt użytkownika.

Wyjątki

Exception	Gdy użytkownik nie istnieje lub hasło jest błędne.

8.2.3.3 Register()

```
async Task Sklep_internetowy.Server.Services.Auth.AccountService.Register (
            string username,
            string email,
            string password)  [inline]
```

Rejestruje nowego użytkownika w systemie.
Tworzy role Admin i User jeśli nie istnieją. Pierwszy użytkownik dostaje rolę Admin.

Parametry

username	Nazwa użytkownika.
email	Adres e-mail.
password	Hasło.

Wyjątki

Exception	Gdy rejestracja się nie powiedzie.

8.2.3.4 UpdateUserAsync()

```
async Task Sklep_internetowy.Server.Services.Auth.AccountService.UpdateUserAsync (
            string currentUsername,
            UpdateUserRequest request)  [inline]
```

Aktualizuje dane użytkownika.

Parametry

currentUsername	Aktualna nazwa użytkownika.
request	Obiekt DTO z nowymi danymi.

Wyjątki

Exception	Gdy użytkownik nie istnieje lub aktualizacja się nie powiedzie.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Auth/AccountService.cs

8.3 Dokumentacja klasy AdminDashboard

Diagram współpracy dla AdminDashboard:

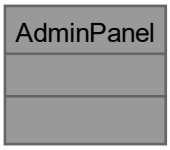
8.3.1 Opis szczegółowy

@description Renderuje responsywny interfejs kafelkowy (grid kart), gdzie kazda karta reprezentuje osobny obszar zarzadzania systemem e-commerce.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/AdminDashboard/AdminDashboard.jsx

8.4 Dokumentacja klasy AdminPanel

Diagram współpracy dla AdminPanel:



8.4.1 Opis szczegółowy

@description Glowny komponent widoku administracyjnego. Zarzadza stanem listy uzytkownikow oraz koordynuje operacje CRUD wykonywane na bazie danych uzytkownikow systemu TechStore.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/AdminDashboard/Users/UsersManage.jsx

8.5 Dokumentacja klasy App

Diagram współpracy dla App:

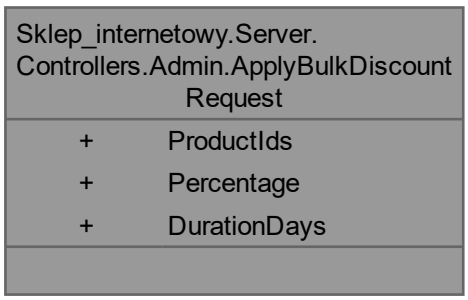
8.5.1 Opis szczegółowy

@description Glowna funkcja aplikacji React. Odpowiada za renderowanie odpowiednich widokow na podstawie sciezki URL oraz przekazywanie stanu porownywarki do komponentow potomnych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/App.jsx

8.6 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.↩ ApplyBulkDiscountRequest

Klasa pomocnicza (DTO) reprezentujaca zadanie masowego nalozenia rabatow.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.ApplyBulkDiscountRequest:



Właściwości

- List< int > **ProductIds** `[get, set]`

  Lista identyfikatorow produktow objetych akcja promocyjna.
- int **Percentage** `[get, set]`

  Wartosc procentowa znizki (np. 20 dla 20% rabatu).
- int **DurationDays** `[get, set]`

  Liczba dni, przez ktore promocja bedzie aktywna od momentu nalozenia.

8.6.1 Opis szczegółowy

Klasa pomocnicza (DTO) reprezentujaca zadanie masowego nalozenia rabatow.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/PromotionController.cs

8.7 Dokumentacja klasy Sklep_internetowy.Server.Models.Auction

Klasa reprezentująca aukcję (licytację) konkretnego produktu w systemie. Przechowuje informacje o cenach, czasie trwania oraz uczestnikach biorących udział w licytacji.
Diagram współpracy dla Sklep_internetowy.Server.Models.Auction:

Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator aukcji.
- int **ProductId** `[get, set]`

  Identyfikator powiązanego produktu, który jest przedmiotem licytacji.

- Product **Product** = null! `[get, set]`

  Obiekt nawigacyjny do szczegółowych danych produktu wystawionego na aukcję.
- decimal **StartingPrice** `[get, set]`

  Cena początkowa (wywoławcza), od której zaczyna się licytacja.
- decimal **CurrentPrice** `[get, set]`

  Aktualna najwyższa cena zaoferowana przez licytujących.
- DateTime **CreatedAt** = DateTime.UtcNow `[get, set]`

  Data i godzina utworzenia aukcji (domyślnie czas UTC).
- DateTime **EndTime** `[get, set]`

  Data i godzina planowanego zakończenia aukcji.
- string? **LastBidderId** `[get, set]`

  Identyfikator użytkownika (GUID), który złożył ostatnią (najwyższą) ofertę.
- User? **LastBidder** `[get, set]`

  Obiekt użytkownika będącego aktualnym liderem licytacji.
- bool **IsFinished** `[get, set]`

  Flaga określająca, czy aukcja została już sfinalizowana.
- string? **WinnerId** `[get, set]`

  Identyfikator użytkownika, który wygrał aukcję (został zwycięzcą).
- List< Bid > **Bids** = new() `[get, set]`

  Kolekcja wszystkich ofert (postąpień) złożonych w ramach danej aukcji.

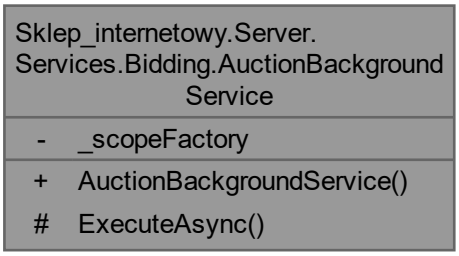
8.7.1 Opis szczegółowy

Klasa reprezentująca aukcję (licytację) konkretnego produktu w systemie. Przechowuje informacje o cenach, czasie trwania oraz uczestnikach biorących udział w licytacji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/Auction.cs

8.8 Dokumentacja klasy Sklep_internetowy.Server.Services.Bidding.↩ AuctionBackgroundService

Usługa działająca w tle odpowiedzialna za automatyczne kończenie aukcji.
Diagram współpracy dla Sklep_internetowy.Server.Services.Bidding.AuctionBackgroundService:



Metody publiczne

- AuctionBackgroundService (IServiceScopeFactory scopeFactory)

  Konstruktor usługi AuctionBackgroundService.

Metody chronione

- override async Task ExecuteAsync (CancellationToken ct)

  Główna pętla wykonywana w tle przez usługę.

Atrybuty prywatne

- readonly IServiceScopeFactory **_scopeFactory**

  Fabryka zakresów usług do tworzenia scope dla zależności.

8.8.1 Opis szczegółowy

Usługa działająca w tle odpowiedzialna za automatyczne kończenie aukcji.
Serwis cyklicznie sprawdza, czy istnieją aukcje, których czas zakończenia już minął i automatycznie je finalizuje przy użyciu AuctionService.

8.8.2 Dokumentacja konstruktora i destruktora

8.8.2.1 AuctionBackgroundService()

```
Sklep_internetowy.Server.Services.Bidding.AuctionBackgroundService.AuctionBackgroundService (
            IServiceScopeFactory scopeFactory)  [inline]
```

Konstruktor usługi AuctionBackgroundService.

Parametry

scopeFactory	Fabryka zakresów serwisów do tworzenia kontekstu zależności.

8.8.3 Dokumentacja funkcji składowych

8.8.3.1 ExecuteAsync()

```
override async Task Sklep_internetowy.Server.Services.Bidding.AuctionBackgroundService.↩
ExecuteAsync (
            CancellationToken ct)  [inline], [protected]
```

Główna pętla wykonywana w tle przez usługę.

Parametry

ct	Token anulowania umożliwiający bezpieczne zatrzymanie usługi.

Zwraca

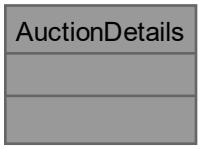
Zadanie asynchroniczne.

Pobranie listy aukcji, które powinny zostać zakończone.
Wybierane są aukcje, które nie są jeszcze zakończone i których czas dobiegł końca.
Finalizacja każdej wygasłej aukcji.
Opóźnienie pomiędzy kolejnymi sprawdzeniami (30 sekund).
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Bidding/AuctionBackgroundService .cs

8.9 Dokumentacja klasy AuctionDetails

Diagram współpracy dla AuctionDetails:



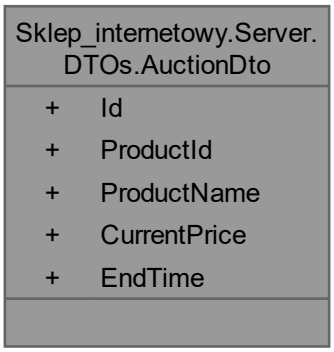
8.9.1 Opis szczegółowy

@description Wyswietla pelne dane aukcji, zarzadza cyklem zycia polaczenia SignalR oraz umozliwia zalogowanym uzytkownikom branie udzialu w licytacji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Auction/AuctionDetails.jsx

8.10 Dokumentacja klasy Sklep_internetowy.Server.DTOs.AuctionDto

Obiekt transferu danych (DTO) reprezentujący podstawowe informacje o aukcji. Klasa służy do przesyłania kluczowych parametrów licytacji pomiędzy warstwą serwerową a interfejsem użytkownika, minimalizując narzut sieciowy poprzez ograniczenie przesyłanych danych.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.AuctionDto:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator aukcji w systemie.
- int **ProductId** `[get, set]`

  Identyfikator techniczny produktu przypisanego do danej aukcji.

- string **ProductName** = string.Empty `[get, set]`

  Nazwa handlowa produktu wystawionego na licytację.
- decimal **CurrentPrice** `[get, set]`

  Aktualna najwyższa cena zaoferowana w ramach aukcji (bieżące przebicie).
- DateTime **EndTime** `[get, set]`

  Data i godzina planowanego zakończenia licytacji w formacie UTC.

8.10.1 Opis szczegółowy

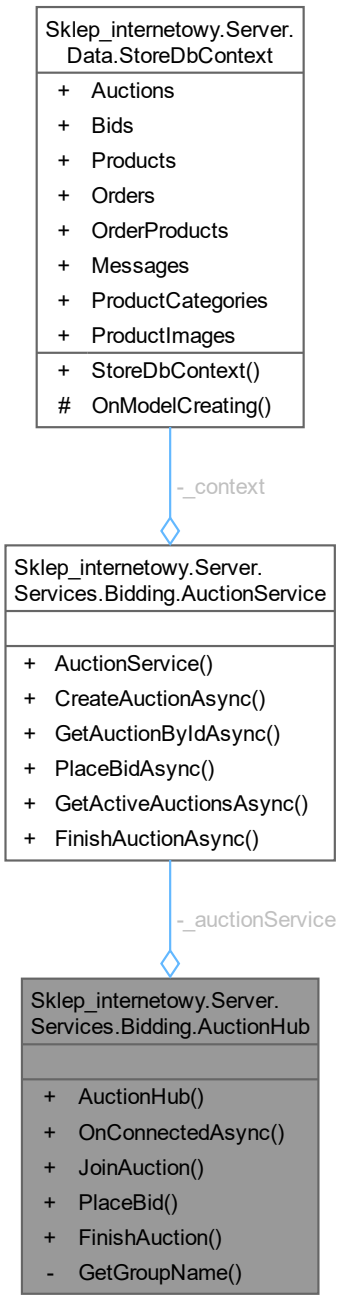
Obiekt transferu danych (DTO) reprezentujący podstawowe informacje o aukcji. Klasa służy do przesyłania kluczowych parametrów licytacji pomiędzy warstwą serwerową a interfejsem użytkownika, minimalizując narzut sieciowy poprzez ograniczenie przesyłanych danych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/AuctionDto.cs

8.11 Dokumentacja klasy Sklep_internetowy.Server.Services.Bidding.AuctionHub

Hub SignalR odpowiedzialny za komunikację w czasie rzeczywistym dla aukcji.

Diagram współpracy dla Sklep_internetowy.Server.Services.Bidding.AuctionHub:



Metody publiczne

- AuctionHub (AuctionService auctionService)

  Konstruktor huba AuctionHub.
- override async Task OnConnectedAsync ()

  Metoda wywoływana po połączeniu klienta z hubem.
- async Task JoinAuction (int auctionId)

  Dodaje użytkownika do grupy SignalR powiązanej z daną aukcją.

- async Task PlaceBid (int auctionId, decimal amount)

  Składa ofertę w wybranej aukcji.
- async Task FinishAuction (int auctionId)

  Wysyła do klientów informację o zakończeniu aukcji.

Metody prywatne

- string GetGroupName (int auctionId)

  Generuje nazwę grupy SignalR dla danej aukcji.

Atrybuty prywatne

- readonly AuctionService **_auctionService**

  Serwis obsługujący logikę aukcji.

8.11.1 Opis szczegółowy

Hub SignalR odpowiedzialny za komunikację w czasie rzeczywistym dla aukcji.
Umożliwia dołączanie użytkowników do aukcji, składanie ofert oraz przesyłanie aktualizacji ceny i czasu zakończenia aukcji do wszystkich uczestników.

8.11.2 Dokumentacja konstruktora i destruktora

8.11.2.1 AuctionHub()

```
Sklep_internetowy.Server.Services.Bidding.AuctionHub.AuctionHub (
            AuctionService auctionService)  [inline]
```

Konstruktor huba AuctionHub.

Parametry

auctionService	Serwis odpowiedzialny za obsługę aukcji.

8.11.3 Dokumentacja funkcji składowych

8.11.3.1 FinishAuction()

```
async Task Sklep_internetowy.Server.Services.Bidding.AuctionHub.FinishAuction (
            int auctionId)  [inline]
```

Wysyła do klientów informację o zakończeniu aukcji.

Parametry

auction↩ Id	Identyfikator aukcji.

Zwraca

Zadanie asynchroniczne.

8.11.3.2 GetGroupName()

```
string Sklep_internetowy.Server.Services.Bidding.AuctionHub.GetGroupName (
            int auctionId)  [inline], [private]
```

Generuje nazwę grupy SignalR dla danej aukcji.

Parametry

auction↩ Id	Identyfikator aukcji.

Zwraca

Nazwa grupy SignalR.

8.11.3.3 JoinAuction()

```
async Task Sklep_internetowy.Server.Services.Bidding.AuctionHub.JoinAuction (
            int auctionId)  [inline]
```

Dodaje użytkownika do grupy SignalR powiązanej z daną aukcją.

Parametry

auction↩ Id	Identyfikator aukcji.

Zwraca

Zadanie asynchroniczne.

8.11.3.4 OnConnectedAsync()

```
override async Task Sklep_internetowy.Server.Services.Bidding.AuctionHub.OnConnectedAsync ()
[inline]
```

Metoda wywoływana po połączeniu klienta z hubem.
Służy do logowania informacji o nowym połączeniu użytkownika.

Zwraca

Zadanie asynchroniczne.

8.11.3.5 PlaceBid()

```
async Task Sklep_internetowy.Server.Services.Bidding.AuctionHub.PlaceBid (
            int auctionId,
            decimal amount)  [inline]
```

Składa ofertę w wybranej aukcji.
Metoda weryfikuje użytkownika, a następnie próbuje zapisać ofertę w systemie. W przypadku sukcesu informuje wszystkich uczestników aukcji o nowej cenie.

Parametry

auction↩ Id	Identyfikator aukcji.
amount	Kwota oferty.

Zwraca

Zadanie asynchroniczne.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Bidding/AuctionHub .cs

8.12 Dokumentacja klasy AuctionList

Diagram współpracy dla AuctionList:

8.12.1 Opis szczegółowy

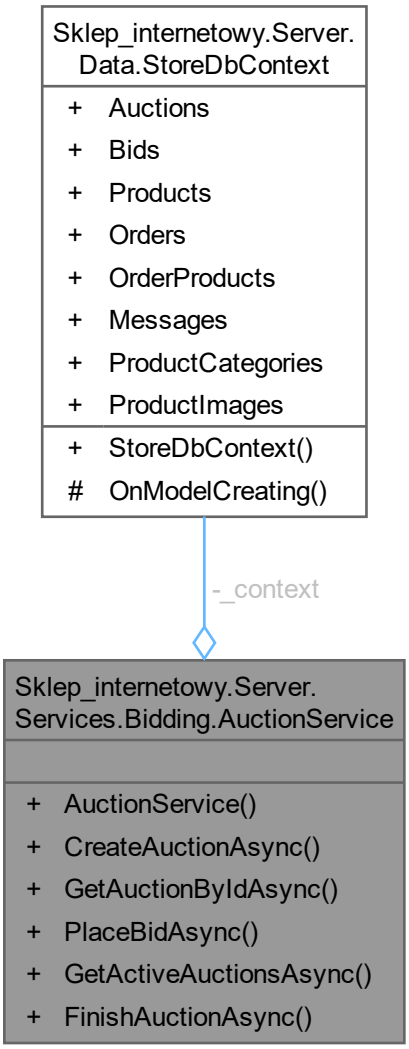
@description Glowny widok publiczny dla modulu aukcji. Zarzadza stanem ladowania danych, obsluga bledow oraz renderowaniem listy licytacji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Auction/AuctionList.jsx

8.13 Dokumentacja klasy Sklep_internetowy.Server.Services.Bidding.AuctionService

Serwis odpowiedzialny za pełną obsługę logiki aukcji.

Diagram współpracy dla Sklep_internetowy.Server.Services.Bidding.AuctionService:



Metody publiczne

- AuctionService (StoreDbContext context, ILogger< AuctionService > logger)

  Konstruktor serwisu AuctionService.
- async Task< Auction > CreateAuctionAsync (int productId, decimal startingPrice)

  Tworzy nową aukcję dla wybranego produktu.
- async Task< Auction?> GetAuctionByIdAsync (int auctionId)

  Pobiera aukcję na podstawie jej identyfikatora.
- async Task< bool > PlaceBidAsync (int auctionId, decimal amount, string userId)

  Składa ofertę w aukcji.
- async Task< List< AuctionDto > > GetActiveAuctionsAsync ()

  Pobiera listę wszystkich aktywnych (niezakończonych) aukcji.
- async Task FinishAuctionAsync (Auction auction)

  Finalizuje aukcję po jej zakończeniu.

8.13.1 Opis szczegółowy

Serwis odpowiedzialny za pełną obsługę logiki aukcji.
Umożliwia tworzenie aukcji, składanie ofert, pobieranie aktywnych aukcji oraz finalizowanie zakończonych licytacji.

8.13.2 Dokumentacja konstruktora i destruktora

8.13.2.1 AuctionService()

```
Sklep_internetowy.Server.Services.Bidding.AuctionService.AuctionService (
            StoreDbContext context,
            ILogger< AuctionService > logger)  [inline]
```

Konstruktor serwisu AuctionService.

Parametry

context	Kontekst bazy danych.
logger	Logger do zapisywania informacji diagnostycznych.

8.13.3 Dokumentacja funkcji składowych

8.13.3.1 CreateAuctionAsync()

```
async Task< Auction > Sklep_internetowy.Server.Services.Bidding.AuctionService.CreateAuction↩
Async (
            int productId,
            decimal startingPrice)  [inline]
```

Tworzy nową aukcję dla wybranego produktu.
Sprawdza, czy produkt istnieje oraz czy nie jest już wystawiony na aukcję.

Parametry

productId	Identyfikator produktu.
startingPrice	Cena początkowa aukcji.

Zwraca

Utworzona aukcja.

8.13.3.2 FinishAuctionAsync()

```
async Task Sklep_internetowy.Server.Services.Bidding.AuctionService.FinishAuctionAsync (
            Auction auction)  [inline]
```

Finalizuje aukcję po jej zakończeniu.
Ustawia zwycięzcę, oznacza aukcję jako zakończoną oraz aktualizuje właściciela produktu.

Parametry

auction	Aukcja do zakończenia.

8.13.3.3 GetActiveAuctionsAsync()

```
async Task< List< AuctionDto > > Sklep_internetowy.Server.Services.Bidding.AuctionService.↩
GetActiveAuctionsAsync ()  [inline]
```

Pobiera listę wszystkich aktywnych (niezakończonych) aukcji.

Zwraca

Lista obiektów AuctionDto.

8.13.3.4 GetAuctionByIdAsync()

```
async Task< Auction?> Sklep_internetowy.Server.Services.Bidding.AuctionService.GetAuctionBy↩
IdAsync (
            int auctionId)  [inline]
```

Pobiera aukcję na podstawie jej identyfikatora.

Parametry

auction↩ Id	Identyfikator aukcji.

Zwraca

Obiekt aukcji lub null, jeśli nie istnieje.

8.13.3.5 PlaceBidAsync()

```
async Task< bool > Sklep_internetowy.Server.Services.Bidding.AuctionService.PlaceBidAsync (
            int auctionId,
            decimal amount,
            string userId)  [inline]
```

Składa ofertę w aukcji.
Sprawdza poprawność aukcji, wysokość oferty oraz aktualizuje dane w bazie.

Parametry

auction↩ Id	Identyfikator aukcji.
amount	Kwota oferty.
userId	Identyfikator użytkownika składającego ofertę.

Zwraca

True jeśli oferta została przyjęta, w przeciwnym wypadku false.

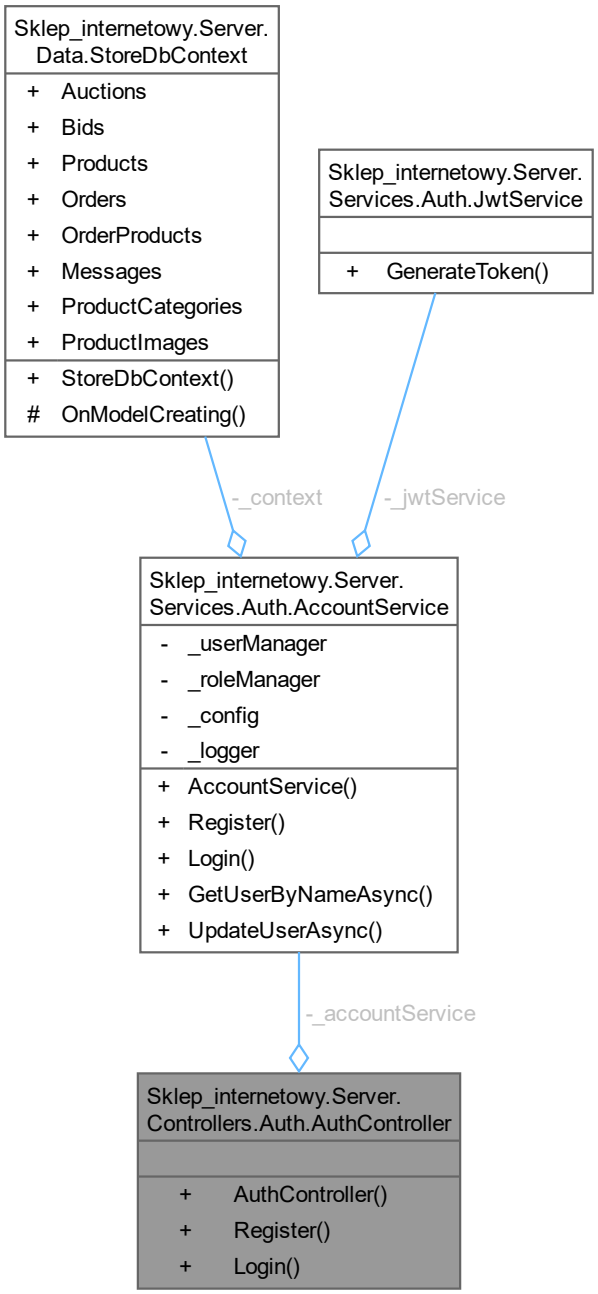
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Bidding/AuctionService.cs

8.14 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Auth.AuthController

Kontroler API odpowiedzialny za procesy uwierzytelniania i autoryzacji uzytkownikow. Obsluguje operacje rejestracji nowych kont oraz logowania (generowania tokenow JWT).

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Auth.AuthController:



Metody publiczne

- AuthController (AccountService AccountServcice, IConfiguration configuration)

  Inicjalizuje nowa instancje klasy AuthController.
- async Task< IActionResult > Register ([FromBody] RegisterUserRequest request)

  Rejestruje nowego uzytkownika w systemie.
- async Task< IActionResult > Login ([FromBody] LoginRequest request)

  Loguje uzytkownika do systemu i zwraca token autoryzacyjny.

8.14.1 Opis szczegółowy

Kontroler API odpowiedzialny za procesy uwierzytelniania i autoryzacji uzytkownikow. Obsluguje operacje rejestracji nowych kont oraz logowania (generowania tokenow JWT).

8.14.2 Dokumentacja konstruktora i destruktora

8.14.2.1 AuthController()

```
Sklep_internetowy.Server.Controllers.Auth.AuthController.AuthController (
            AccountService AccountServcice,
            IConfiguration configuration)  [inline]
```

Inicjalizuje nowa instancje klasy AuthController.

Parametry

AccountServcice	Serwis zarzadzajacy kontami uzytkownikow i logika uwierzytelniania.
configuration	Interfejs dostepu do konfiguracji aplikacji (np. ustawienia JWT).

8.14.3 Dokumentacja funkcji składowych

8.14.3.1 Login()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Auth.AuthController.Login (
            [FromBody] LoginRequest request)  [inline]
```

Loguje uzytkownika do systemu i zwraca token autoryzacyjny.

Parametry

request	Obiekt DTO zawierajacy poswiadczenia (login i haslo).

Zwraca

Token JWT oraz podstawowe dane uzytkownika (ID, e-mail).

<response code="200">Zwraca token dostępowy oraz dane zalogowanego profilu.</response> <response code="400">Gdy poswiadczenia sa bledne lub wystapil blad serwera.</response>

8.14.3.2 Register()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Auth.AuthController.Register
(
            [FromBody] RegisterUserRequest request)  [inline]
```

Rejestruje nowego uzytkownika w systemie.

Parametry

request	Obiekt DTO zawierajacy nazwe uzytkownika, e-mail oraz haslo.

Zwraca

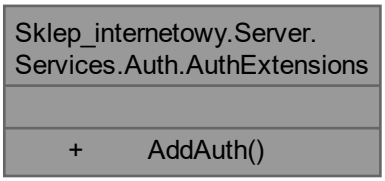
Komunikat o pomyslnej rejestracji lub blad walidacji.

<response code="200">Uzytkownik zostal pomyslnie zarejestrowany.</response> <response code="400">Gdy dane wejsciowe sa nieprawidlowe lub proces rejestracji sie nie powiodl.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Auth/AuthController.cs

8.15 Dokumentacja klasy Sklep_internetowy.Server.Services.Auth.AuthExtensions

Klasa rozszerzeń konfiguracji uwierzytelniania JWT.
Diagram współpracy dla Sklep_internetowy.Server.Services.Auth.AuthExtensions:



Statyczne metody publiczne

- static IServiceCollection AddAuth (this IServiceCollection serviceCollection, IConfiguration configuration)

  Dodaje konfigurację uwierzytelniania JWT do kontenera usług.

8.15.1 Opis szczegółowy

Klasa rozszerzeń konfiguracji uwierzytelniania JWT.
Zawiera metodę rozszerzającą do konfiguracji mechanizmu autoryzacji w aplikacji.

8.15.2 Dokumentacja funkcji składowych

8.15.2.1 AddAuth()

```
IServiceCollection Sklep_internetowy.Server.Services.Auth.AuthExtensions.AddAuth (
            this IServiceCollection serviceCollection,
            IConfiguration configuration)  [inline], [static]
```

Dodaje konfigurację uwierzytelniania JWT do kontenera usług.
Konfiguruje walidację tokena JWT oraz klucz szyfrujący na podstawie ustawień aplikacji.

Parametry

serviceCollection	Kolekcja usług aplikacji.
configuration	Konfiguracja aplikacji.

Zwraca

Zmieniona kolekcja usług z dodaną obsługą JWT.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Auth/AuthExtensions.cs

8.16 Dokumentacja klasy Sklep_internetowy.Server.Services.Auth.AuthSettings

Klasa przechowująca ustawienia uwierzytelniania JWT.

Diagram współpracy dla Sklep_internetowy.Server.Services.Auth.AuthSettings:

Właściwości

- TimeSpan Expires `[get, set]`

  Czas ważności tokena JWT.
- string SecretKey `[get, set]`

  Tajny klucz używany do podpisywania tokenów JWT.

8.16.1 Opis szczegółowy

Klasa przechowująca ustawienia uwierzytelniania JWT.
Obiekt tej klasy jest mapowany z konfiguracji aplikacji (np. appsettings.json) i zawiera parametry potrzebne do generowania oraz weryfikacji tokenów JWT.

8.16.2 Dokumentacja właściwości

8.16.2.1 Expires

`TimeSpan Sklep_internetowy.Server.Services.Auth.AuthSettings.Expires [get], [set]`
Czas ważności tokena JWT.
Określa jak długo token pozostaje aktywny od momentu jego wygenerowania.

8.16.2.2 SecretKey

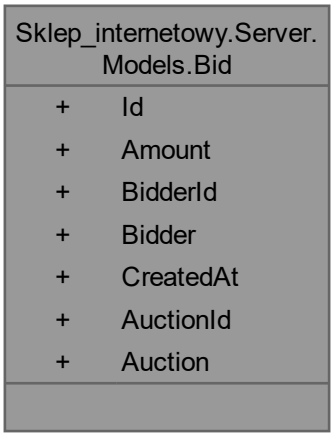
`string Sklep_internetowy.Server.Services.Auth.AuthSettings.SecretKey [get], [set]`
Tajny klucz używany do podpisywania tokenów JWT.
Klucz ten musi być odpowiednio długi i przechowywany w bezpiecznym miejscu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Auth/AuthSettings.cs

8.17 Dokumentacja klasy Sklep_internetowy.Server.Models.Bid

Klasa reprezentująca pojedynczą ofertę licytacji (postąpienie) złożoną przez użytkownika. Przechowuje informacje o kwocie, czasie złożenia oraz powiązaniach z konkretną aukcją i licytantem.

Diagram współpracy dla Sklep_internetowy.Server.Models.Bid:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator rekordu oferty.
- decimal **Amount** `[get, set]`

  Kwota zaoferowana przez licytanta w ramach tego postąpienia.
- string **BidderId** = null! `[get, set]`

  Identyfikator użytkownika (GUID), który jest autorem oferty.
- User **Bidder** = null! `[get, set]`

  Obiekt użytkownika składającego ofertę, umożliwiający dostęp do jego danych profilowych.
- DateTime **CreatedAt** = DateTime.UtcNow `[get, set]`

  Data i godzina zarejestrowania oferty w systemie (domyślnie czas UTC).
- int **AuctionId** `[get, set]`

  Identyfikator aukcji, której dotyczy dana oferta cenowa.
- Auction **Auction** = null! `[get, set]`

  Obiekt aukcji powiązanej z tą ofertą, umożliwiający nawigację do szczegółów licytacji.

8.17.1 Opis szczegółowy

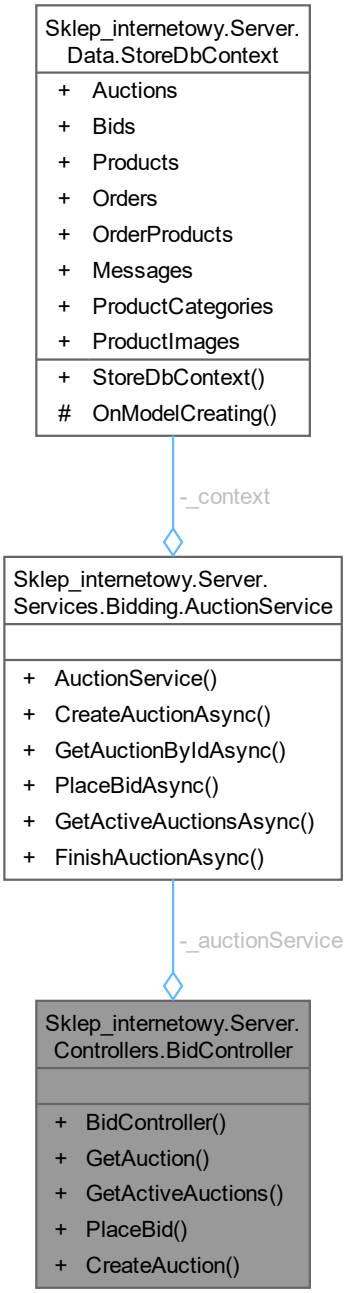
Klasa reprezentująca pojedynczą ofertę licytacji (postąpienie) złożoną przez użytkownika. Przechowuje informacje o kwocie, czasie złożenia oraz powiązaniach z konkretną aukcją i licytantem.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/Bid.cs

8.18 Dokumentacja klasy Sklep_internetowy.Server.Controllers.BidController

Kontroler API odpowiedzialny za obsluge systemu aukcyjnego oraz licytacji. Umozliwia przegladanie aktywnych aukcji, skladanie ofert oraz wystawianie nowych przedmiotow na licytacje.

Diagram współpracy dla Sklep_internetowy.Server.Controllers.BidController:



Metody publiczne

- BidController (AuctionService auctionService)

  Inicjalizuje nowa instancje klasy BidController.
- async Task< IActionResult > GetAuction (int id)

  Pobiera szczegolowe informacje o konkretnej aukcji na podstawie jej identyfikatora.
- async Task< IActionResult > GetActiveAuctions ()

  Pobiera liste wszystkich obecnie trwajacych (aktywnych) aukcji.

- async Task< IActionResult > PlaceBid (int id, [FromBody] PlaceBidDto dto)

  Pozwala zalogowanemu uzytkownikowi na zlozenie oferty (podbicie ceny) w wybranej aukcji.
- async Task< IActionResult > CreateAuction ([FromBody] CreateAuctionDto dto)

  Tworzy nowa aukcje dla wskazanego produktu z okreslona cena wywolawcza.

8.18.1 Opis szczegółowy

Kontroler API odpowiedzialny za obsluge systemu aukcyjnego oraz licytacji. Umozliwia przegladanie aktywnych aukcji, skladanie ofert oraz wystawianie nowych przedmiotow na licytacje.

8.18.2 Dokumentacja konstruktora i destruktora

8.18.2.1 BidController()

```
Sklep_internetowy.Server.Controllers.BidController.BidController (
            AuctionService auctionService)  [inline]
```

Inicjalizuje nowa instancje klasy BidController.

Parametry

auctionService	Serwis obslugujacy logike biznesowa aukcji i licytacji.

8.18.3 Dokumentacja funkcji składowych

8.18.3.1 CreateAuction()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.BidController.CreateAuction (
            [FromBody] CreateAuctionDto dto)  [inline]
```

Tworzy nowa aukcje dla wskazanego produktu z okreslona cena wywolawcza.

Parametry

dto	Dane niezbedne do utworzenia aukcji (ID produktu, cena startowa).

Zwraca

Dane nowo utworzonej aukcji.

<response code="200">Aukcja zostala pomyslnie utworzona.</response>

8.18.3.2 GetActiveAuctions()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.BidController.GetActive↩
Auctions ()  [inline]
```

Pobiera liste wszystkich obecnie trwajacych (aktywnych) aukcji.

Zwraca

Kolekcja aktywnych aukcji.

<response code="200">Zwraca liste aktywnych licytacji.</response>

8.18.3.3 GetAuction()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.BidController.GetAuction (
            int id)  [inline]
```

Pobiera szczegolowe informacje o konkretnej aukcji na podstawie jej identyfikatora.

Parametry

id	Unikalny identyfikator aukcji.

Zwraca

Obiekt aukcji lub blad 404 w przypadku braku znalezienia.

<response code="200">Zwraca dane wybranej aukcji.</response> <response code="404">Gdy aukcja o podanym ID nie istnieje.</response>

8.18.3.4 PlaceBid()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.BidController.PlaceBid (
            int id,
            [FromBody] PlaceBidDto dto)  [inline]
```

Pozwala zalogowanemu uzytkownikowi na zlozenie oferty (podbicie ceny) w wybranej aukcji.

Parametry

id	ID aukcji, w ktorej skladana jest oferta.
dto	Obiekt DTO zawierajacy deklarowana kwote licytacji.

Zwraca

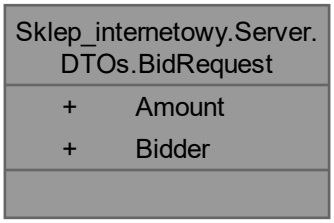
Informacja o powodzeniu operacji lub blad walidacji licytacji.

<response code="200">Oferta zostala pomyslnie przyjeta.</response> <response code="400">Gdy oferta jest zbyt niska, aukcja sie zakonczyla lub wystapil blad autoryzacji.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Bidding/BidController.cs

8.19 Dokumentacja klasy Sklep_internetowy.Server.DTOs.BidRequest

Obiekt transferu danych (DTO) reprezentujący żądanie złożenia nowej oferty licytacyjnej. Przechowuje informacje o deklarowanej kwocie oraz tożsamości licytanta niezbędne do przetworzenia postąpienia przez silnik aukcyjny.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.BidRequest:



Właściwości

- decimal **Amount** `[get, set]`

Kwota zaoferowana przez użytkownika w ramach licytacji.

- string **Bidder** = string.Empty `[get, set]`

  Nazwa wyświetlana lub unikalny identyfikator użytkownika składającego ofertę.

8.19.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący żądanie złożenia nowej oferty licytacyjnej. Przechowuje informacje o deklarowanej kwocie oraz tożsamości licytanta niezbędne do przetworzenia postąpienia przez silnik aukcyjny.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/BidRequest.cs

8.20 Dokumentacja klasy Carousel

Diagram współpracy dla Carousel:

8.20.1 Opis szczegółowy

@description Wyswietla interaktywny suwak obrazow (carousel) z kontrolkami nawigacji. Wykorzystuje klasy Bootstrapa (data-bs-ride="carousel") do zapewnienia animacji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/carousel/Carousel.jsx

8.21 Dokumentacja klasy Cart

Diagram współpracy dla Cart:

8.21.1 Opis szczegółowy

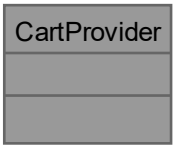
@description Renderuje interfejs koszyka. Wykorzystuje dane z CartContext do wyswietlania dynamicznej listy produktow oraz obliczania sumy zamowienia.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Cart/Cart.jsx

8.22 Dokumentacja klasy CartProvider

Provider kontekstu koszyka dla calej aplikacji.
Diagram współpracy dla CartProvider:



8.22.1 Opis szczegółowy

Provider kontekstu koszyka dla calej aplikacji.

Parametry

{Object}	children Komponenty potomne aplikacji.

Zwraca

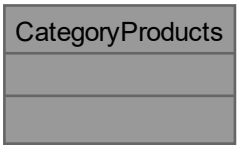
JSX.Element Provider kontekstu koszyka.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/context/CartContext.jsx

8.23 Dokumentacja klasy CategoryProducts

Diagram współpracy dla CategoryProducts:



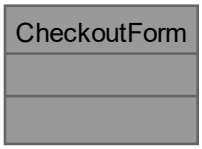
8.23.1 Opis szczegółowy

@description Renderuje widok kategorii produktow. Zarzadza stanem ladowania, bledami polaczenia oraz logika wyswietlania produktow w ukladzie wierszowym.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/CategoryProducts.jsx

8.24 Dokumentacja klasy CheckoutForm

Diagram współpracy dla CheckoutForm:



8.24.1 Opis szczegółowy

@description Podkomponent renderujacy formularz platnosci (PaymentElement). Zarzadza komunikacja z API Stripe w celu potwierdzenia srodkow.

Parametry

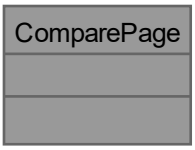
{Object}	props - Wlasciwosci komponentu.
{Function}	props.onSuccess - Callback wywolywany po pomyslnej autoryzacji platnosci.
{number}	props.amount - Calkowita kwota do zaplaty.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Payment/PaymentPage.jsx

8.25 Dokumentacja klasy ComparePage

Diagram współpracy dla ComparePage:



8.25.1 Opis szczegółowy

@description Renderuje interfejs porownawczy. Obsluguje wyswietlanie zdjec, cen, dostepnosci oraz opisow produktow.

Parametry

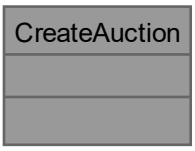
{Object}	props - Wlasciwosci komponentu.
{Object}	props.comparison - Obiekt zawierajacy stan i funkcje porownywarki.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/ComparePage.jsx

8.26 Dokumentacja klasy CreateAuction

Diagram współpracy dla CreateAuction:



8.26.1 Opis szczegółowy

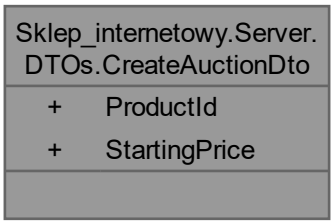
@description Renderuje formularz do zakladania aukcji. Zawiera walidacje uprawnien oraz pol wejsciowych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Auction/CreateAuction.jsx

8.27 Dokumentacja klasy Sklep_internetowy.Server.DTOs.CreateAuctionDto

Obiekt transferu danych (DTO) wykorzystywany do zainicjowania nowej aukcji w systemie. Zawiera niezbędne informacje o produkcie przeznaczonym do sprzedaży licytacyjnej oraz ustalonej dla niego cenie wywoławczej.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.CreateAuctionDto:



Właściwości

- int **ProductId** `[get, set]`

  Unikalny identyfikator produktu, który ma zostać wystawiony na aukcję.
- decimal **StartingPrice** `[get, set]`

  Cena początkowa (wywoławcza), od której rozpocznie się proces licytacji produktu.

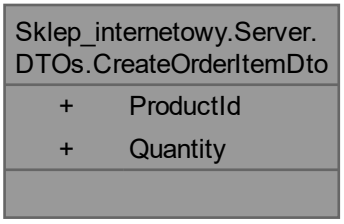
8.27.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany do zainicjowania nowej aukcji w systemie. Zawiera niezbędne informacje o produkcie przeznaczonym do sprzedaży licytacyjnej oraz ustalonej dla niego cenie wywoławczej.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/CreateAuctionDto.cs

8.28 Dokumentacja klasy Sklep_internetowy.Server.DTOs.CreateOrderItemDto

Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję w żądaniu utworzenia zamówienia. Zawiera informacje niezbędne do zidentyfikowania produktu oraz określenia zamawianej ilości sztuk.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.CreateOrderItemDto:



Właściwości

- int **ProductId** `[get, set]`

Unikalny identyfikator techniczny produktu (ID).

- int **Quantity** `[get, set]`

  Liczba sztuk danego produktu, którą użytkownik zamierza nabyć.

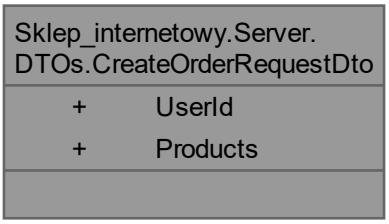
8.28.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję w żądaniu utworzenia zamówienia. Zawiera informacje niezbędne do zidentyfikowania produktu oraz określenia zamawianej ilości sztuk.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/CreateOrderDto.cs

8.29 Dokumentacja klasy Sklep_internetowy.Server.DTOs.CreateOrderRequestDto

Obiekt transferu danych (DTO) reprezentujący kompletne żądanie utworzenia nowego zamówienia. Przesyłany z warstwy frontendu podczas finalizacji koszyka zakupowego w celu zainicjowania transakcji po stronie serwerowej.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.CreateOrderRequestDto:



Właściwości

- string **UserId** = null! `[get, set]`

  Unikalny identyfikator użytkownika (GUID) składającego zamówienie.

- List< CreateOrderItemDto > **Products** = new List<CreateOrderItemDto>() `[get, set]`

  Lista pozycji wchodzących w skład zamówienia (kolekcja produktów wraz z ilościami).

8.29.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący kompletne żądanie utworzenia nowego zamówienia. Przesyłany z warstwy frontendu podczas finalizacji koszyka zakupowego w celu zainicjowania transakcji po stronie serwerowej.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/CreateOrderDto.cs

8.30 Dokumentacja klasy Sklep_internetowy.Server.DTOs.CreateProductDto

Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu do katalogu sklepu. Zawiera kompletny zestaw informacji o towarze, jego parametrach cenowych oraz logice promocyjnej.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.CreateProductDto:

Właściwości

- string **Name** = null! `[get, set]`

  Nazwa handlowa produktu (pole wymagane).
- decimal **Price** `[get, set]`

  Podstawowa cena jednostkowa towaru.
- int **Quantity** `[get, set]`

  Początkowa liczba sztuk wprowadzana na stan magazynowy.
- string? **Description** `[get, set]`

  Szczegółowy opis techniczny lub marketingowy produktu.
- decimal? **DiscountPercentage** `[get, set]`

  Wartość procentowa rabatu w przedziale od 0 do 100.
- DateTime? **DiscountStartDate** `[get, set]`

  Data i godzina rozpoczęcia okresu obowiązywania promocji.
- DateTime? **DiscountEndDate** `[get, set]`

  Data i godzina zakończenia okresu obowiązywania promocji.
- int **ProductCategoryId** `[get, set]`

  Identyfikator kategorii nadrzędnej, do której zostanie przypisany produkt.

8.30.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu do katalogu sklepu. Zawiera kompletny zestaw informacji o towarze, jego parametrach cenowych oraz logice promocyjnej.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/CreateProductDto.cs

8.31 Dokumentacja klasy Sklep_internetowy.Server.DTOs.CreateProductWithFilesDto

Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu wraz z załącznikami multimedialnymi. Klasa obsługuje dane przesyłane w formacie multipart/form-data, umożliwiając jednoczesną definicję parametrów handlowych oraz przesyłanie plików graficznych do serwera.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.CreateProductWithFilesDto:

Właściwości

- string **Name** = null! `[get, set]`

  Nazwa handlowa produktu.
- decimal **Price** `[get, set]`

  Podstawowa cena jednostkowa towaru brutto.
- int **Quantity** `[get, set]`

  Początkowa liczba sztuk wprowadzana na stan magazynowy.
- string? **Description** `[get, set]`

  Szczegółowy opis charakterystyki, specyfikacji lub przeznaczenia produktu.
- int **ProductCategoryId** `[get, set]`

  Identyfikator kategorii nadrzędnej, do której zostanie przypisany produkt w katalogu.
- decimal? **DiscountPercentage** `[get, set]`

  Wartość procentowa rabatu (np. 10.50 dla 10.5% zniżki).
- DateTime? **DiscountStartDate** `[get, set]`

  Data i godzina rozpoczęcia okresu obowiązywania ceny promocyjnej (UTC).
- DateTime? **DiscountEndDate** `[get, set]`

  Data i godzina zakończenia okresu obowiązywania ceny promocyjnej (UTC).
- string **Brand** = null! `[get, set]`

  Marka, producent lub linia produktowa, do której należy towar.
- List< IFormFile >? **Images** `[get, set]`

  Kolekcja plików graficznych (zdjęć) przesyłanych w ramach żądania HTTP. Mapowana z formularza binarnego typu IFormFile.

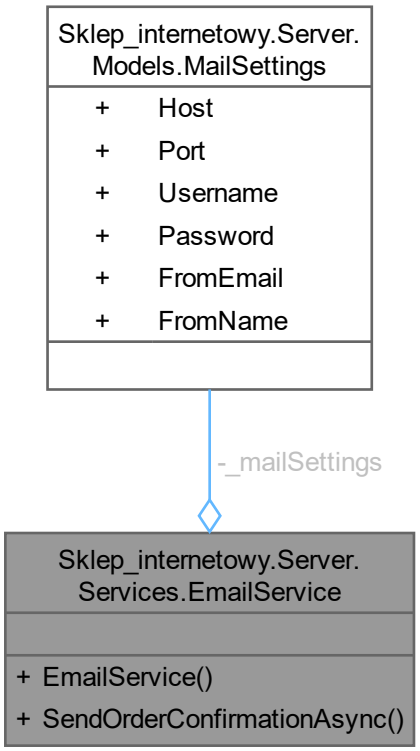
8.31.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany podczas procesu dodawania nowego produktu wraz z załącznikami multimedialnymi. Klasa obsługuje dane przesyłane w formacie multipart/form-data, umożliwiając jednoczesną definicję parametrów handlowych oraz przesyłanie plików graficznych do serwera.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/CreateProductWithFilesDto.cs

8.32 Dokumentacja klasy Sklep_internetowy.Server.Services.EmailService

Serwis odpowiedzialny za wysyłanie wiadomości e-mail.
Diagram współpracy dla Sklep_internetowy.Server.Services.EmailService:



Metody publiczne

- EmailService (IOptions< MailSettings > mailSettings)

  Konstruktor serwisu EmailService.
- async Task SendOrderConfirmationAsync (string toEmail, int orderId, decimal totalAmount, string orderDate)

  Wysyła e-mail z potwierdzeniem złożenia zamówienia.

8.32.1 Opis szczegółowy

Serwis odpowiedzialny za wysyłanie wiadomości e-mail.
Wykorzystywany głównie do wysyłania potwierdzeń zamówień oraz komunikacji systemowej z użytkownikiem.

8.32.2 Dokumentacja konstruktora i destruktora

8.32.2.1 EmailService()

```
Sklep_internetowy.Server.Services.EmailService.EmailService (
            IOptions< MailSettings > mailSettings)  [inline]
```

Konstruktor serwisu EmailService.

Parametry

mailSettings	Ustawienia serwera pocztowego wczytane z konfiguracji aplikacji.

8.32.3 Dokumentacja funkcji składowych

8.32.3.1 SendOrderConfirmationAsync()

```
async Task Sklep_internetowy.Server.Services.EmailService.SendOrderConfirmationAsync (
            string toEmail,
            int orderId,
            decimal totalAmount,
            string orderDate)  [inline]
```

Wysyła e-mail z potwierdzeniem złożenia zamówienia.
Generuje wiadomość HTML zawierającą szczegóły zamówienia i wysyła ją do klienta.

Parametry

toEmail	Adres e-mail odbiorcy.
orderId	Identyfikator zamówienia.
totalAmount	Łączna kwota do zapłaty.
orderDate	Data złożenia zamówienia.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Email/EmailService.cs

8.33 Dokumentacja klasy FeaturedProductCard

Diagram współpracy dla FeaturedProductCard:

8.33.1 Opis szczegółowy

@description Wyswietla atrakcyjna wizualnie karte produktu z efektem najechania (shadow-lg) i skalowaniem, dedykowana dla sekcji "Wyroznione" na stronie glownej.

Parametry

{Object}	props - Wlasciwosci komponentu.
{Object}	props.product - Obiekt danych produktu (ID, nazwa, opis, ceny, znizki).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/featuredProductCard/FeaturedProductCard.jsx

8.34 Dokumentacja klasy Footer

Glowny komponent stopki strony.
Diagram współpracy dla Footer:

8.34.1 Opis szczegółowy

Glowny komponent stopki strony.
Zawiera formularz kontaktowy, obsluge jego stanu, wysylke danych do API oraz sekcje z informacja o lokalizacji firmy.

Zwraca

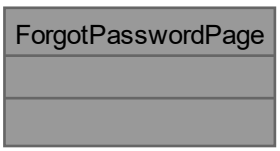
JSX.Element Element stopki strony.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/footer/Footer.jsx

8.35 Dokumentacja klasy ForgotPasswordPage

Diagram współpracy dla ForgotPasswordPage:



8.35.1 Opis szczegółowy

@description Renderuje minimalistyczny formularz zapytania o reset hasla. Zarzadza lokalnym stanem adresu e-mail i obsluguje logike wysylania danych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/ForgotPassword/ForgotPasswordPage.jsx

8.36 Dokumentacja klasy Home

Diagram współpracy dla Home:

8.36.1 Opis szczegółowy

@description Renderuje uklad strony glownej. Wykorzystuje fragmenty React do zwrocenia komponentow Carousel oraz ProductGrid bez dodawania zbednych wezlow DOM.

Zwraca

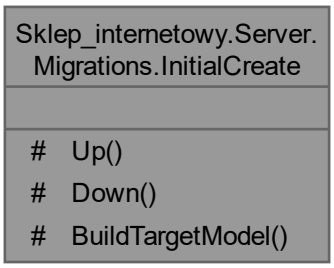
{JSX.Element} Widok strony glownej.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Home/Home.jsx

8.37 Dokumentacja klasy Sklep_internetowy.Server.Migrations.InitialCreate

Diagram współpracy dla Sklep_internetowy.Server.Migrations.InitialCreate:



Metody chronione

- override void **Up** (MigrationBuilder migrationBuilder)
- override void **Down** (MigrationBuilder migrationBuilder)
- override void **BuildTargetModel** (ModelBuilder modelBuilder)

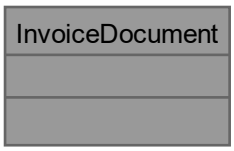
8.37.1 Opis szczegółowy

Dokumentacja dla tej klasy została wygenerowana z plików:

- Sklep_internetowy.Server/Migrations/20260121215558_InitialCreate.cs
- Sklep_internetowy.Server/Migrations/20260121215558_InitialCreate.Designer.cs

8.38 Dokumentacja klasy InvoiceDocument

Komponent renderujacy pelny dokument faktury VAT.
Diagram współpracy dla InvoiceDocument:



8.38.1 Opis szczegółowy

Komponent renderujacy pelny dokument faktury VAT.

Parametry

{Object}	props Wlasciwosci komponentu.
{Object}	props.order Obiekt zamowienia zawierajacy dane klienta, status oraz liste produktow.

Zwraca

{JSX.Element} Obiekt dokumentu PDF.

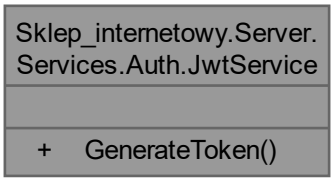
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/admin/InvoiceGenerator/InvoiceDocument.jsx

8.39 Dokumentacja klasy Sklep_internetowy.Server.Services.Auth.JwtService(IOptions< AuthSettings > options, UserManager< User > userManager)

Serwis odpowiedzialny za generowanie tokenów JWT dla użytkowników.

Diagram współpracy dla Sklep_internetowy.Server.Services.Auth.JwtService:



Metody publiczne

- async Task< string > GenerateToken (User user)

  Generuje token JWT dla podanego użytkownika.

8.39.1 Opis szczegółowy

Serwis odpowiedzialny za generowanie tokenów JWT dla użytkowników.
Tworzy token JWT zawierający dane użytkownika oraz jego role na podstawie ustawień z klasy AuthSettings.

8.39.2 Dokumentacja funkcji składowych

8.39.2.1 GenerateToken()

```
async Task< string > Sklep_internetowy.Server.Services.Auth.JwtService.GenerateToken (
            User user)  [inline]
```

Generuje token JWT dla podanego użytkownika.

Parametry

user	Obiekt użytkownika, dla którego generowany jest token.

Zwraca

Ciąg znaków reprezentujący token JWT.

Lista claimów umieszczanych w tokenie JWT.
Zawiera podstawowe informacje identyfikujące użytkownika.
Pobranie ról przypisanych do użytkownika.
Dodanie ról użytkownika do listy claimów.
Utworzenie obiektu tokena JWT.
Serializacja tokena JWT do postaci tekstowej.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Auth/JwtService.cs

8.40 Dokumentacja klasy LoginPage

Diagram współpracy dla LoginPage:

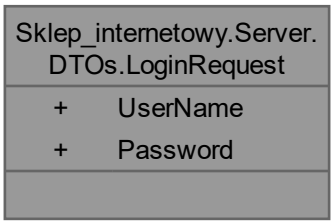
8.40.1 Opis szczegółowy

@description Renderuje formularz logowania i zarzadza stanem uwierzytelniania. Po pomyslnym logowaniu zapisuje dane sesji w localStorage i przekierowuje uzytkownika.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/LoginPage/LoginPage.jsx

8.41 Dokumentacja klasy Sklep_internetowy.Server.DTOs.LoginRequest

Obiekt transferu danych (DTO) wykorzystywany podczas procesu uwierzytelniania użytkownika. Przechowuje poświadczenia (login i hasło) niezbędne do weryfikacji tożsamości i wygenerowania tokenu autoryzacyjnego w systemie TechStore.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.LoginRequest:



Właściwości

- string **UserName** = null! `[get, set]`

  Nazwa użytkownika (login) przypisana do konta w bazie danych.
- string **Password** = null! `[get, set]`

  Hasło użytkownika przesyłane w celu sprawdzenia poprawności autoryzacji.

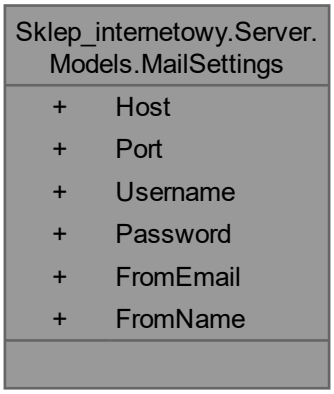
8.41.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany podczas procesu uwierzytelniania użytkownika. Przechowuje poświadczenia (login i hasło) niezbędne do weryfikacji tożsamości i wygenerowania tokenu autoryzacyjnego w systemie TechStore.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/LoginRequest.cs

8.42 Dokumentacja klasy Sklep_internetowy.Server.Models.MailSettings

Klasa konfiguracyjna przechowująca parametry serwera poczty elektronicznej (SMTP). Wykorzystywana przez serwis e-mail do autoryzacji i wysyłania powiadomień systemowych.
Diagram współpracy dla Sklep_internetowy.Server.Models.MailSettings:



Właściwości

- string **Host** = string.Empty `[get, set]`

  Adres serwera SMTP (np. smtp.gmail.com).
- int **Port** `[get, set]`

  Numer portu wykorzystywanego do połączenia z serwerem poczty (np. 587 dla TLS).
- string **Username** = string.Empty `[get, set]`

  Nazwa użytkownika (login) do autoryzacji na serwerze pocztowym.
- string **Password** = string.Empty `[get, set]`

  Hasło lub token aplikacji używany do uwierzytelnienia nadawcy.
- string **FromEmail** = string.Empty `[get, set]`

  Adres e-mail, który będzie wyświetlany jako nadawca wiadomości.
- string **FromName** = string.Empty `[get, set]`

  Nazwa wyświetlana nadawcy (np. "TechStore - Powiadomienia").

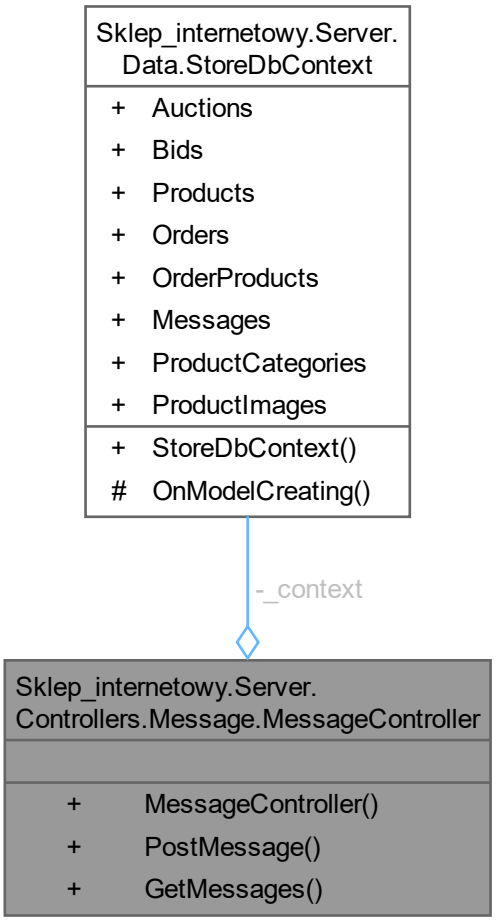
8.42.1 Opis szczegółowy

Klasa konfiguracyjna przechowująca parametry serwera poczty elektronicznej (SMTP). Wykorzystywana przez serwis e-mail do autoryzacji i wysyłania powiadomień systemowych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/MailSettings.cs

8.43 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Message.MessageController

Kontroler API odpowiedzialny za obsluge wiadomosci kontaktowych przesylanych przez uzytkownikow. Modul umozliwia zapisywanie nowych zgloszen oraz ich odczyt w panelu administracyjnym.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Message.MessageController:



Metody publiczne

- MessageController (StoreDbContext context)

  Inicjalizuje nowa instancje klasy MessageController.
- async Task< IActionResult > PostMessage ([FromBody] UserMessage message)

  Przyjmuje nowa wiadomosc uzytkownika i zapisuje ja w bazie danych.
- async Task< ActionResult< IEnumerable< UserMessage > > > GetMessages ()

  Pobiera pelna liste wiadomosci uzytkownikow zapisanych w systemie. Wyniki sa sortowane malejaco wedlug daty utworzenia.

8.43.1 Opis szczegółowy

Kontroler API odpowiedzialny za obsluge wiadomosci kontaktowych przesylanych przez uzytkownikow. Modul umozliwia zapisywanie nowych zgloszen oraz ich odczyt w panelu administracyjnym.

8.43.2 Dokumentacja konstruktora i destruktora

8.43.2.1 MessageController()

```
Sklep_internetowy.Server.Controllers.Message.MessageController.MessageController (
            StoreDbContext context)  [inline]
```

Inicjalizuje nowa instancje klasy MessageController.

Parametry

context	Kontekst bazy danych StoreDbContext do obslugi operacji na wiadomosciach.

8.43.3 Dokumentacja funkcji składowych

8.43.3.1 GetMessages()

```
async Task< ActionResult< IEnumerable< UserMessage > > > Sklep_internetowy.Server.Controllers.↩
Message.MessageController.GetMessages ()  [inline]
```

Pobiera pelna liste wiadomosci uzytkownikow zapisanych w systemie. Wyniki sa sortowane malejaco wedlug daty utworzenia.

Zwraca

Kolekcja obiektow UserMessage reprezentujacych zgloszenia.

<response code="200">Zwraca liste wiadomosci.</response>

8.43.3.2 PostMessage()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Message.MessageController.↩
PostMessage (
            [FromBody] UserMessage message)  [inline]
```

Przyjmuje nowa wiadomosc uzytkownika i zapisuje ja w bazie danych.

Parametry

message	Obiekt UserMessage przekazany w korpusie zadania.

Zwraca

Status powodzenia operacji wraz z komunikatem potwierdzajacym.

<response code="200">Wiadomosc zostala pomyslnie zapisana.</response> <response code="400">Gdy przeslany model danych jest nieprawidlowy.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Message/MessageController.cs

8.44 Dokumentacja klasy Navbar

Glowny komponent paska nawigacji.

Diagram współpracy dla Navbar:

8.44.1 Opis szczegółowy

Glowny komponent paska nawigacji.

Parametry

{number}	compareCount Liczba produktow w porownywarce.

Zwraca

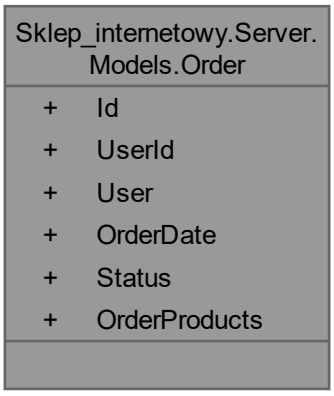
JSX.Element Element paska nawigacji.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/navbar/Navbar.jsx

8.45 Dokumentacja klasy Sklep_internetowy.Server.Models.Order

Klasa reprezentująca zamówienie klienta w systemie TechStore. Przechowuje informacje o dacie zakupu, statusie oraz powiązaniach z użytkownikiem i produktami.
Diagram współpracy dla Sklep_internetowy.Server.Models.Order:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator zamówienia w bazie danych.
- string **UserId** = null! `[get, set]`

  Identyfikator użytkownika (GUID), który dokonał zakupu.
- User **User** = null! `[get, set]`

  Obiekt nawigacyjny do szczegółowych danych użytkownika składającego zamówienie.
- DateTime **OrderDate** = DateTime.UtcNow `[get, set]`

  Data i godzina zarejestrowania zamówienia (domyślnie czas UTC).
- string **Status** = "pending" `[get, set]`

  Aktualny etap realizacji zamówienia (domyślnie "pending").
- ICollection< OrderProduct > **OrderProducts** = new List<OrderProduct>() `[get, set]`

  Kolekcja pozycji wchodzących w skład zamówienia (tabela łącząca z produktami).

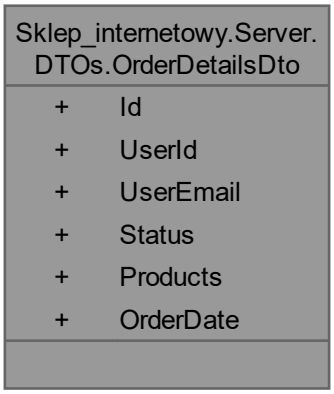
8.45.1 Opis szczegółowy

Klasa reprezentująca zamówienie klienta w systemie TechStore. Przechowuje informacje o dacie zakupu, statusie oraz powiązaniach z użytkownikiem i produktami.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/Order.cs

8.46 Dokumentacja klasy Sklep_internetowy.Server.DTOs.OrderDetailsDto

Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o zamówieniu klienta. Klasa agreguje dane identyfikacyjne użytkownika, aktualny status realizacji zamówienia oraz pełną listę produktów wraz z datą operacji, służąc do prezentacji danych w interfejsie użytkownika.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.OrderDetailsDto:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator zamówienia w bazie danych.

- string **UserId** = null! `[get, set]`

  Identyfikator użytkownika (GUID), który złożył zamówienie.
- string **UserEmail** = null! `[get, set]`

  Adres e-mail użytkownika przypisany do zamówienia, wykorzystywany do powiadomień.
- string **Status** = null! `[get, set]`

  Aktualny status realizacji zamówienia (np. "Pending", "Shipped", "Completed").
- List< OrderProductDetailsDto > **Products** = new List<OrderProductDetailsDto>() `[get, set]`

  Kolekcja szczegółowych informacji o produktach wchodzących w skład tego zamówienia.
- DateTime **OrderDate** `[get, set]`

  Data i godzina zarejestrowania zamówienia w systemie.

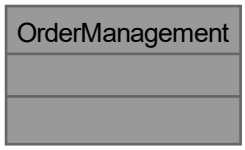
8.46.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o zamówieniu klienta. Klasa agreguje dane identyfikacyjne użytkownika, aktualny status realizacji zamówienia oraz pełną listę produktów wraz z datą operacji, służąc do prezentacji danych w interfejsie użytkownika.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/OrderDetailsDto.cs

8.47 Dokumentacja klasy OrderManagement

Diagram współpracy dla OrderManagement:



8.47.1 Opis szczegółowy

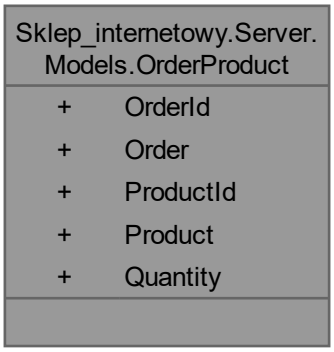
@description Glowny widok administracyjny zarzadzania zamowieniami.
Integruje operacje pobierania danych, walidacji stanow magazynowych oraz obslugi procesow CRUD.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/AdminDashboard/Order/OrderManagement.jsx

8.48 Dokumentacja klasy Sklep_internetowy.Server.Models.OrderProduct

Klasa pośrednicząca (tabela łącząca) reprezentująca relację wiele-do-wielu pomiędzy zamówieniami a produktami. Przechowuje informacje o tym, jakie produkty i w jakiej ilości wchodzą w skład danego zamówienia.

Diagram współpracy dla Sklep_internetowy.Server.Models.OrderProduct:



Właściwości

- int **OrderId** `[get, set]`

  Identyfikator powiązanego zamówienia.
- Order **Order** = null! `[get, set]`

  Obiekt nawigacyjny do szczegółowych danych zamówienia.
- int **ProductId** `[get, set]`

  Identyfikator powiązanego produktu.
- Product **Product** = null! `[get, set]`

  Obiekt nawigacyjny do szczegółowych danych produktu.
- int **Quantity** `[get, set]`

  Liczba sztuk danego produktu zakupiona w ramach tego zamówienia.

8.48.1 Opis szczegółowy

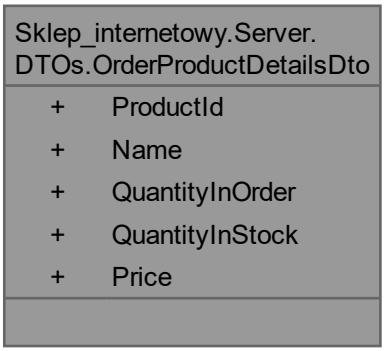
Klasa pośrednicząca (tabela łącząca) reprezentująca relację wiele-do-wielu pomiędzy zamówieniami a produktami. Przechowuje informacje o tym, jakie produkty i w jakiej ilości wchodzą w skład danego zamówienia.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/OrderProduct.cs

8.49 Dokumentacja klasy Sklep_internetowy.Server.DTOs.OrderProductDetailsDto

Obiekt transferu danych (DTO) zawierający szczegółowe informacje o konkretnym produkcie przypisanym do zamówienia. Przechowuje dane o wolumenie zakupu, stanach magazynowych oraz historycznej cenie sprzedaży.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.OrderProductDetailsDto:



Właściwości

- int **ProductId** `[get, set]`

  Unikalny identyfikator techniczny produktu.
- string **Name** = null! `[get, set]`

  Nazwa handlowa produktu wyświetlana w podsumowaniu zamówienia.
- int **QuantityInOrder** `[get, set]`

  Liczba sztuk danego produktu zakupiona w ramach tego zamówienia.
- int **QuantityInStock** `[get, set]`

  Aktualny stan magazynowy produktu (ułatwia weryfikację dostępności przy edycji).
- decimal **Price** `[get, set]`

  Cena jednostkowa produktu obowiązująca w momencie składania zamówienia.

8.49.1 Opis szczegółowy

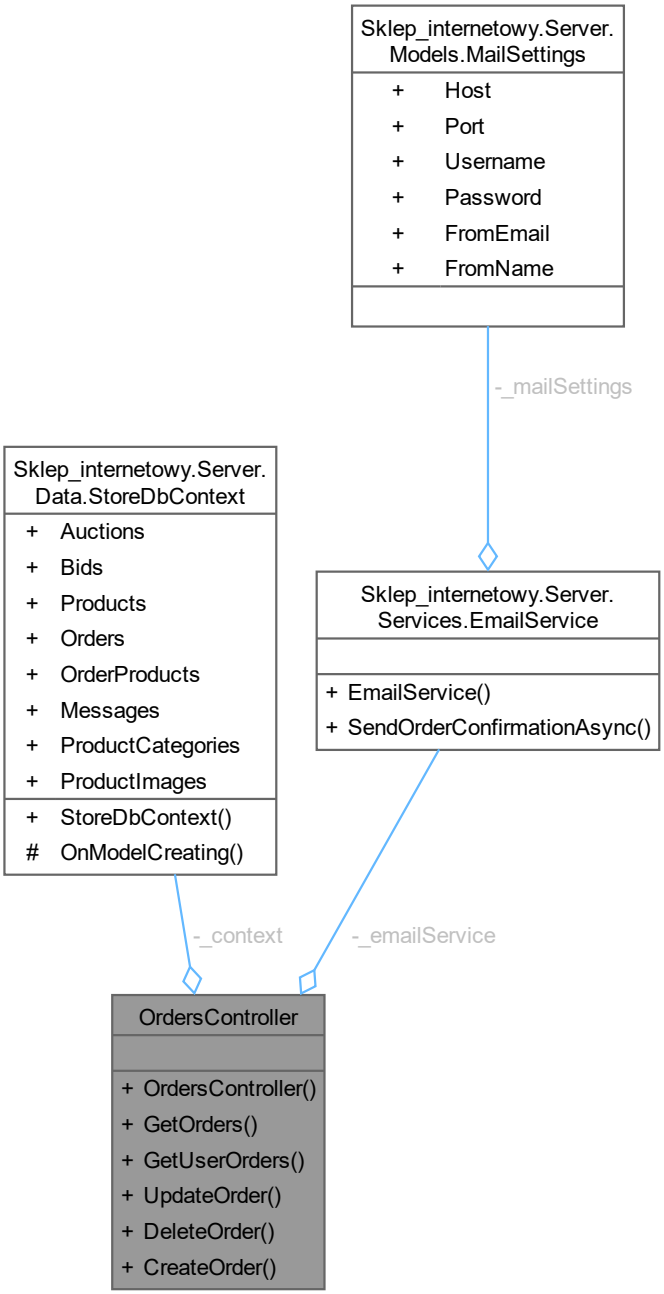
Obiekt transferu danych (DTO) zawierający szczegółowe informacje o konkretnym produkcie przypisanym do zamówienia. Przechowuje dane o wolumenie zakupu, stanach magazynowych oraz historycznej cenie sprzedaży.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/OrderProductDetailsDto.cs

8.50 Dokumentacja klasy OrdersController

Kontroler API odpowiedzialny za zarzadzanie zamowieniami w systemie. Obsluguje procesy przegladania zamowien, ich tworzenia, aktualizacji statusow oraz usuwania, uwzgledniajac przy tym automatyczna synchronizacje stanow magazynowych.

Diagram współpracy dla OrdersController:



Metody publiczne

- OrdersController (StoreDbContext context, EmailService emailService)

  Inicjalizuje nowa instancje klasy OrdersController.
- async Task< ActionResult< IEnumerable< OrderDetailsDto > > > GetOrders ()

  Pobiera liste wszystkich zamowien zarejestrowanych w systemie. Zawiera szczegolowe informacje o uzytkownikach oraz przypisanych produktach.
- async Task< ActionResult< IEnumerable< OrderDetailsDto > > > GetUserOrders (string userId)

Pobiera historie zamowien konkretnego uzytkownika na podstawie jego identyfikatora.

- async Task< IActionResult > UpdateOrder (int id, [FromBody] OrderUpdateDto dto)

  Aktualizuje istniejace zamowienie (status oraz liste produktow). Metoda zarzadza stanem magazynowym, przywracajac produkty ze starego zamowienia i odejmujac nowe ilosci. Operacja wykonywana jest w ramach transakcji.

- async Task< IActionResult > DeleteOrder (int id)

  Usuwa zamowienie z systemu i przywraca ilosci zarezerwowanych produktow do magazynu. Wykorzystuje transakcje w celu zapewnienia spojnosci danych.

- async Task< ActionResult< OrderDetailsDto > > CreateOrder ([FromBody] CreateOrderRequestDto dto)

  Tworzy nowe zamowienie w systemie. Weryfikuje dostepnosc produktow, aktualizuje stan magazynowy i wysyla e-mail potwierdzajacy.

8.50.1 Opis szczegółowy

Kontroler API odpowiedzialny za zarzadzanie zamowieniami w systemie. Obsluguje procesy przegladania zamowien, ich tworzenia, aktualizacji statusow oraz usuwania, uwzgledniajac przy tym automatyczna synchronizacje stanow magazynowych.

8.50.2 Dokumentacja konstruktora i destruktora

8.50.2.1 OrdersController()

```
OrdersController.OrdersController (
            StoreDbContext context,
            EmailService emailService)  [inline]
```

Inicjalizuje nowa instancje klasy OrdersController.

Parametry

context	Kontekst bazy danych do operacji na zamowieniach i produktach.
emailService	Serwis uzywany do wysylania powiadomien e-mail do klientow.

8.50.3 Dokumentacja funkcji składowych

8.50.3.1 CreateOrder()

```
async Task< ActionResult< OrderDetailsDto > > OrdersController.CreateOrder (
            [FromBody] CreateOrderRequestDto dto)  [inline]
```

Tworzy nowe zamowienie w systemie. Weryfikuje dostepnosc produktow, aktualizuje stan magazynowy i wysyla e-mail potwierdzajacy.

Parametry

dto	Dane niezbedne do utworzenia nowego zamowienia.

Zwraca

Szczegoly nowo utworzonego zamowienia (OrderDetailsDto).

8.50.3.2 DeleteOrder()

```
async Task< IActionResult > OrdersController.DeleteOrder (
            int id)  [inline]
```

Usuwa zamowienie z systemu i przywraca ilosci zarezerwowanych produktow do magazynu. Wykorzystuje transakcje w celu zapewnienia spojnosci danych.

Parametry

id	Identyfikator zamowienia przeznaczonego do usuniecia.

Zwraca

Status 204 No Content po pomyslnym usunieciu.

8.50.3.3 GetOrders()

```
async Task< ActionResult< IEnumerable< OrderDetailsDto > > > OrdersController.GetOrders ()
[inline]
```

Pobiera liste wszystkich zamowien zarejestrowanych w systemie. Zawiera szczegolowe informacje o uzytkownikach oraz przypisanych produktach.

Zwraca

Kolekcja obiektow OrderDetailsDto reprezentujacych zamowienia.

8.50.3.4 GetUserOrders()

```
async Task< ActionResult< IEnumerable< OrderDetailsDto > > > OrdersController.GetUserOrders
(
            string userId)  [inline]
```

Pobiera historie zamowien konkretnego uzytkownika na podstawie jego identyfikatora.

Parametry

user↩ Id	Unikalny identyfikator uzytkownika.

Zwraca

Lista zamowien przypisanych do danego uzytkownika, posortowana od najnowszych.

8.50.3.5 UpdateOrder()

```
async Task< IActionResult > OrdersController.UpdateOrder (
            int id,
            [FromBody] OrderUpdateDto dto)  [inline]
```

Aktualizuje istniejace zamowienie (status oraz liste produktow). Metoda zarzadza stanem magazynowym, przywracajac produkty ze starego zamowienia i odejmujac nowe ilosci. Operacja wykonywana jest w ramach transakcji.

Parametry

id	Identyfikator zamowienia do edycji.
dto	Obiekt DTO zawierajacy zaktualizowane dane zamowienia.

Zwraca

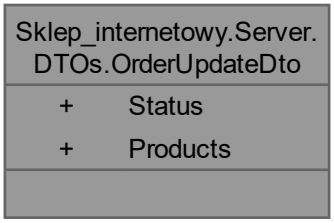
Status 204 No Content w przypadku sukcesu lub opis bledu walidacji.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Shop/OrdersController.cs

8.51 Dokumentacja klasy Sklep_internetowy.Server.DTOs.OrderUpdateDto

Obiekt transferu danych (DTO) wykorzystywany do kompleksowej modyfikacji istniejącego zamówienia. Klasa umożliwia jednoczesną zmianę statusu logistycznego zamówienia oraz aktualizację zestawienia produktów wchodzących w jego skład.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.OrderUpdateDto:



Właściwości

- string **Status** = null! `[get, set]`

  Nowy status realizacji zamówienia (np. "Processing", "Shipped", "Cancelled").
- List< OrderUpdateItemDto > **Products** = new List<OrderUpdateItemDto>() `[get, set]`

  Kolekcja pozycji (produktów i ich ilości), które mają zostać zaktualizowane w bazie danych.

8.51.1 Opis szczegółowy

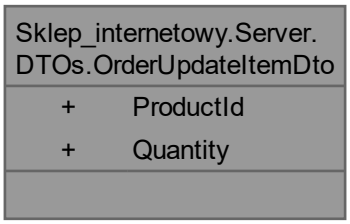
Obiekt transferu danych (DTO) wykorzystywany do kompleksowej modyfikacji istniejącego zamówienia. Klasa umożliwia jednoczesną zmianę statusu logistycznego zamówienia oraz aktualizację zestawienia produktów wchodzących w jego skład.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/OrderUpdateDto.cs

8.52 Dokumentacja klasy Sklep_internetowy.Server.DTOs.OrderUpdateItemDto

Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję towarową w żądaniu aktualizacji zamówienia. Przechowuje informacje niezbędne do skorygowania ilości konkretnego produktu w ramach istniejącej transakcji.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.OrderUpdateItemDto:



Właściwości

- int **ProductId** `[get, set]`

  Unikalny identyfikator techniczny produktu (ID).
- int **Quantity** `[get, set]`

  Docelowa liczba sztuk produktu, która ma zostać przypisana do zamówienia po aktualizacji.

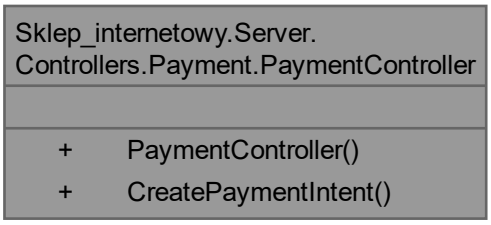
8.52.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący pojedynczą pozycję towarową w żądaniu aktualizacji zamówienia. Przechowuje informacje niezbędne do skorygowania ilości konkretnego produktu w ramach istniejącej transakcji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/OrderUpdateDto.cs

8.53 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Payment.PaymentController

Kontroler API odpowiedzialny za integracje z systemem platnosci Stripe. Umozliwia generowanie zamiarow platnosci (PaymentIntents) dla transakcji elektronicznych.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Payment.PaymentController:



Metody publiczne

- PaymentController (IConfiguration configuration)

Inicjalizuje nowa instancje klasy PaymentController. Konfiguruje globalny klucz API Stripe na podstawie pliku konfiguracyjnego aplikacji.

- ActionResult CreatePaymentIntent ([FromBody] PaymentRequest request)

  Tworzy nowy obiekt PaymentIntent w systemie Stripe. Zamiar ten jest wykorzystywany przez frontend do bezpiecznego sfinalizowania transakcji.

8.53.1 Opis szczegółowy

Kontroler API odpowiedzialny za integracje z systemem platnosci Stripe. Umozliwia generowanie zamiarow platnosci (PaymentIntents) dla transakcji elektronicznych.

8.53.2 Dokumentacja konstruktora i destruktora

8.53.2.1 PaymentController()

```
Sklep_internetowy.Server.Controllers.Payment.PaymentController.PaymentController (
            IConfiguration configuration)  [inline]
```

Inicjalizuje nowa instancje klasy PaymentController. Konfiguruje globalny klucz API Stripe na podstawie pliku konfiguracyjnego aplikacji.

Parametry

configuration	Interfejs dostepu do konfiguracji systemowej (sekcja Stripe:SecretKey).

8.53.3 Dokumentacja funkcji składowych

8.53.3.1 CreatePaymentIntent()

```
ActionResult Sklep_internetowy.Server.Controllers.Payment.PaymentController.CreatePayment↩
Intent (
            [FromBody] PaymentRequest request)  [inline]
```

Tworzy nowy obiekt PaymentIntent w systemie Stripe. Zamiar ten jest wykorzystywany przez frontend do bezpiecznego sfinalizowania transakcji.

Parametry

request	Obiekt zawierajacy calkowita kwote do zaplaty (w jednostkach glownych, np. zlotych).

Zwraca

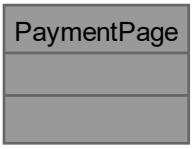
Obiekt JSON zawierajacy clientSecret podrzednego obiektu PaymentIntent.

<response code="200">Zwraca klucz sesji platnosci (clientSecret).</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Payment/PaymentController.cs

8.54 Dokumentacja klasy PaymentPage

Diagram współpracy dla PaymentPage:



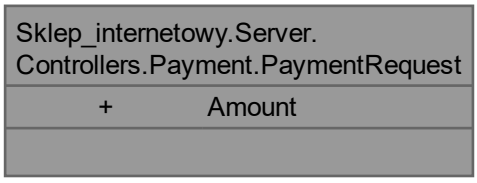
8.54.1 Opis szczegółowy

@description Glowny komponent strony platnosci. Pobiera clientSecret z backendu, wyswietla podsumowanie koszyka i zarzadza procesem zapisu zamowienia w bazie danych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Payment/PaymentPage.jsx

8.55 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Payment.PaymentRequest

Klasa pomocnicza (DTO) reprezentujaca zadanie utworzenia platnosci.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Payment.PaymentRequest:



Właściwości

- long **Amount** `[get, set]`

  Calkowita kwota zamowienia do zaplaty.

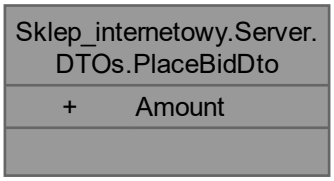
8.55.1 Opis szczegółowy

Klasa pomocnicza (DTO) reprezentujaca zadanie utworzenia platnosci.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Payment/PaymentController.cs

8.56 Dokumentacja klasy Sklep_internetowy.Server.DTOs.PlaceBidDto

Obiekt transferu danych (DTO) reprezentujący ofertę złożoną przez użytkownika w ramach aukcji. Przechowuje informację o kwocie postąpienia, która jest przesyłana do serwera w celu walidacji i aktualizacji bieżącej ceny licytowanego przedmiotu.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.PlaceBidDto:



Właściwości

- decimal **Amount** `[get, set]`

  Wartość pieniężna zadeklarowana przez licytanta jako nowa oferta cenowa.

8.56.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący ofertę złożoną przez użytkownika w ramach aukcji. Przechowuje informację o kwocie postąpienia, która jest przesyłana do serwera w celu walidacji i aktualizacji bieżącej ceny licytowanego przedmiotu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/PlaceBidDto.cs

8.57 Dokumentacja klasy Sklep_internetowy.Server.Models.Product

Klasa reprezentująca produkt w systemie sklepu internetowego. Zawiera szczegółowe informacje o towarze, stanach magazynowych oraz logikę zarządzania cenami promocyjnymi i powiązaniami z kategoriami.

Diagram współpracy dla Sklep_internetowy.Server.Models.Product:

Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator produktu w bazie danych.
- string **Name** = null! `[get, set]`

  Nazwa handlowa produktu.
- decimal **Price** `[get, set]`

  Podstawowa cena jednostkowa przed naliczeniem rabatów.
- int **Quantity** `[get, set]`

  Liczba dostępnych sztuk produktu w magazynie.
- string? **Description** `[get, set]`

  Opcjonalny opis techniczny lub marketingowy produktu.
- bool **IsOnAuction** `[get, set]`

  Flaga określająca, czy produkt jest aktualnie wystawiony w module aukcyjnym.
- string? **OwnerId** `[get, set]`

  Identyfikator właściciela lub sprzedawcy produktu (jeśli dotyczy).
- User? **Owner** `[get, set]`

  Obiekt nawigacyjny do danych właściciela produktu.
- string **Brand** = null! `[get, set]`

  Marka lub producent danego towaru.
- decimal? **DiscountPercentage** `[get, set]`

  Wartość procentowa rabatu (np. 20.00 dla 20% zniżki).
- DateTime? **DiscountStartDate** `[get, set]`

  Data i godzina rozpoczęcia okresu obowiązywania promocji.
- DateTime? **DiscountEndDate** `[get, set]`

  Data i godzina zakończenia okresu obowiązywania promocji.

- ICollection< ProductImage > **Images** = new List<ProductImage>() `[get, set]`

  Kolekcja zdjęć przypisanych do tego produktu.
- decimal **FinalPrice** `[get]`

  Wylicza cenę końcową produktu. Jeśli promocja jest aktywna, zwraca cenę pomniejszoną o rabat, w przeciwnym razie cenę bazową.
- bool **HasActiveDiscount** `[get]`

  Weryfikuje, czy produkt posiada w tej chwili aktywną i ważną zniżkę. Sprawdza zarówno wartość procentową, jak i ramy czasowe (UTC).
- int **ProductCategoryId** `[get, set]`

  Identyfikator kategorii nadrzędnej, do której należy produkt.
- ProductCategory **ProductCategory** = null! `[get, set]`

  Obiekt kategorii umożliwiający dostęp do jej nazwy i parametrów.
- ICollection< OrderProduct > **OrderProducts** = new List<OrderProduct>() `[get, set]`

  Kolekcja pozycji zamówień zawierających ten produkt (relacja wiele-do-wielu).

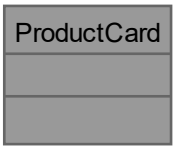
8.57.1 Opis szczegółowy

Klasa reprezentująca produkt w systemie sklepu internetowego. Zawiera szczegółowe informacje o towarze, stanach magazynowych oraz logikę zarządzania cenami promocyjnymi i powiązaniami z kategoriami.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/Product.cs

8.58 Dokumentacja klasy ProductCard

Komponent pojedynczej karty produktu.
Diagram współpracy dla ProductCard:



8.58.1 Opis szczegółowy

Komponent pojedynczej karty produktu.
@description Wyswietla skrocone informacje o produkcie w formie karty. Zawiera funkcje zarzadzania produktem, takie jak edycja i usuwanie, oraz wizualna informacje o promocjach.

Parametry

{Object}	props - Wlasciwosci komponentu.
{Object}	props.product - Obiekt danych produktu (ID, nazwa, cena, stan, rabaty).
{Function}	props.onEdit - Funkcja wywolywana przy kliknieciu przycisku edycji, przyjmuje obiekt produktu.
{Function}	props.onDelete - Funkcja wywolywana po potwierdzeniu usuniecia, przyjmuje ID produktu.
{Object}	product Obiekt produktu do wyswietlenia.

{Function}	addToCart Funkcja dodajaca produkt do koszyka.
{Function}	onClick Funkcja wywolywana po kliknieciu w karte produktu.

Zwraca

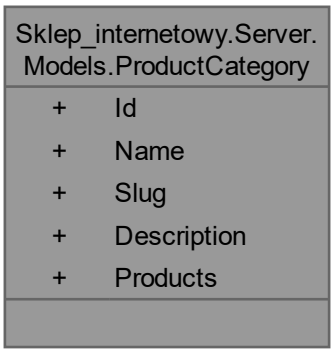
JSX.Element Element karty produktu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/admin/product/ProductCard.jsx

8.59 Dokumentacja klasy Sklep_internetowy.Server.Models.ProductCategory

Klasa reprezentująca kategorię produktów w systemie TechStore. Służy do logicznego grupowania towarów, co ułatwia zarządzanie asortymentem oraz nawigację użytkownika w części frontendowej sklepu.
Diagram współpracy dla Sklep_internetowy.Server.Models.ProductCategory:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator kategorii w bazie danych.
- string **Name** = null! `[get, set]`

  Nazwa wyświetlana kategorii (np. "Laptopy", "Akcesoria").
- string **Slug** = null! `[get, set]`

  Przyjazny dla wyszukiwarek tekstowy identyfikator URL (tzw. slug).
- string? **Description** `[get, set]`

  Opcjonalny opis charakterystyki lub przeznaczenia danej kategorii.
- ICollection< Product > **Products** = new List<Product>() `[get, set]`

  Lista produktów przypisanych do tej kategorii (relacja jeden-do-wielu).

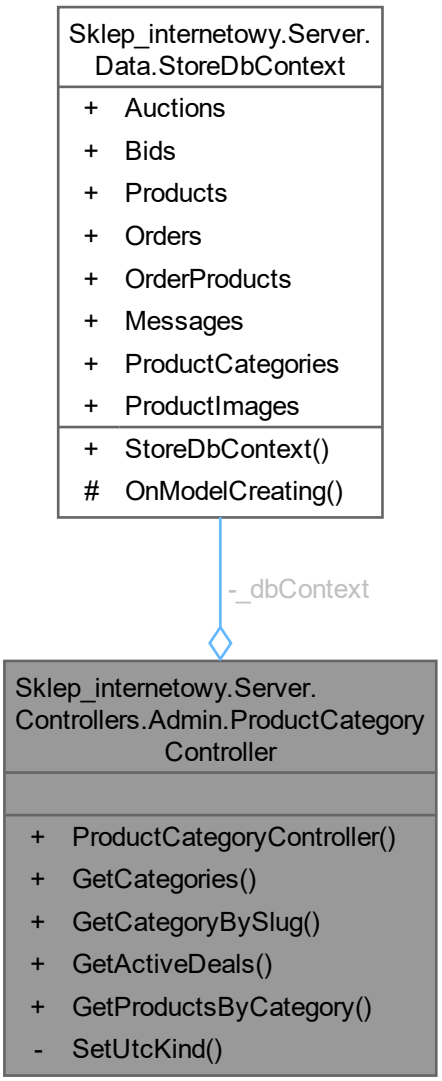
8.59.1 Opis szczegółowy

Klasa reprezentująca kategorię produktów w systemie TechStore. Służy do logicznego grupowania towarów, co ułatwia zarządzanie asortymentem oraz nawigację użytkownika w części frontendowej sklepu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/ProductCategory.cs

8.60 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.↩ ProductCategoryController

Kontroler API odpowiedzialny za zarzadzanie kategoriami produktow oraz filtrowanie ofert specjalnych. Umozliwia pobieranie list kategorii, szczegolowych danych o kategorii oraz powiazanych z nimi produktow.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.ProductCategoryController:



Metody publiczne

- ProductCategoryController (StoreDbContext dbContext)

  Inicjalizuje nowa instancje klasy ProductCategoryController.
- async Task< ActionResult< IEnumerable< ProductCategoryDto > > > GetCategories ()

  Pobiera liste wszystkich dostepnych kategorii produktow.
- async Task< ActionResult< ProductCategoryDto > > GetCategoryBySlug (string slug)

  Pobiera szczegolowe dane o konkretnej kategorii na podstawie jej tekstowego identyfikatora (slug).

- async Task< ActionResult< IEnumerable< ProductDto > > > GetActiveDeals ()

  Pobiera liste wszystkich produktow objetych aktywna promocja (Deals).
- async Task< ActionResult< IEnumerable< ProductDto > > > GetProductsByCategory (string slug)

  Pobiera wszystkie produkty przypisane do kategorii o wskazanym identyfikatorze slug.

Metody prywatne

- DateTime? SetUtcKind (DateTime? date)

  Pomocnicza funkcja ustawiajaca rodzaj daty na UTC. Zapewnia kompatybilnosc z wymaganiami baz danych dotyczacymi stref czasowych.

8.60.1 Opis szczegółowy

Kontroler API odpowiedzialny za zarzadzanie kategoriami produktow oraz filtrowanie ofert specjalnych. Umozliwia pobieranie list kategorii, szczegolowych danych o kategorii oraz powiazanych z nimi produktow.

8.60.2 Dokumentacja konstruktora i destruktora

8.60.2.1 ProductCategoryController()

```
Sklep_internetowy.Server.Controllers.Admin.ProductCategoryController.ProductCategoryController
(
            StoreDbContext dbContext)  [inline]
```

Inicjalizuje nowa instancje klasy ProductCategoryController.

Parametry

dbContext	Kontekst bazy danych do obslugi operacji na kategoriach i produktach.

8.60.3 Dokumentacja funkcji składowych

8.60.3.1 GetActiveDeals()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductCategoryController.GetActiveDeals ()  [inline]
```

Pobiera liste wszystkich produktow objetych aktywna promocja (Deals).

Szczegóły

Metoda weryfikuje posiadanie znizki oraz sprawdza, czy aktualna data miesci sie w zdefiniowanych ramach czasowych promocji.

Zwraca

Lista produktow spelniajacych kryteria promocji wraz z ich zdjeciami.

<response code="200">Zwraca liste aktywnych promocji.</response> <response code="500">Gdy wystapi blad serwera podczas przetwarzania danych.</response>

8.60.3.2 GetCategories()

```
async Task< ActionResult< IEnumerable< ProductCategoryDto > > > Sklep_internetowy.Server.↩
Controllers.Admin.ProductCategoryController.GetCategories ()  [inline]
```

Pobiera liste wszystkich dostepnych kategorii produktow.

Zwraca

Kolekcja obiektow ProductCategoryDto zawierajacych nazwe, slug i opis kategorii.

<response code="200">Zwraca liste kategorii.</response>

8.60.3.3 GetCategoryBySlug()

```
async Task< ActionResult< ProductCategoryDto > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductCategoryController.GetCategoryBySlug (
            string slug)  [inline]
```

Pobiera szczegolowe dane o konkretnej kategorii na podstawie jej tekstowego identyfikatora (slug).

Parametry

slug	Unikalny, przyjazny adres URL kategorii.

Zwraca

Obiekt ProductCategoryDto lub blad 404, jesli kategoria nie istnieje.

<response code="200">Zwraca dane wybranej kategorii.</response> <response code="404">Gdy kategoria o podanym identyfikatorze nie zostala znaleziona.</response>

8.60.3.4 GetProductsByCategory()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductCategoryController.GetProductsByCategory (
            string slug)  [inline]
```

Pobiera wszystkie produkty przypisane do kategorii o wskazanym identyfikatorze slug.

Parametry

slug	Tekstowy identyfikator kategorii (np. "laptops").

Zwraca

Kolekcja produktow nalezacych do danej kategorii.

<response code="200">Zwraca liste produktow dla kategorii.</response> <response code="500">Gdy wystapi blad bazy danych lub przetwarzania.</response>

8.60.3.5 SetUtcKind()

```
DateTime? Sklep_internetowy.Server.Controllers.Admin.ProductCategoryController.SetUtcKind (
            DateTime? date)  [inline], [private]
```

Pomocnicza funkcja ustawiajaca rodzaj daty na UTC. Zapewnia kompatybilnosc z wymaganiami baz danych dotyczacymi stref czasowych.

Parametry

date	Data do przetworzenia.

Zwraca

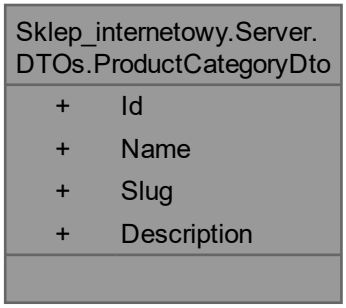
Data z wymuszonym rodzajem DateTimeKind.Utc lub null.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/ProductCategoryController.cs

8.61 Dokumentacja klasy Sklep_internetowy.Server.DTOs.ProductCategoryDto

Obiekt transferu danych (DTO) reprezentujący kategorię produktów w systemie. Służy do przesyłania podstawowych informacji o grupach towarowych, wykorzystywanych w menu nawigacyjnym, filtrach oraz przy przypisywaniu produktów do katalogu.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.ProductCategoryDto:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator kategorii w bazie danych.
- string **Name** = null! `[get, set]`

  Nazwa wyświetlana kategorii (np. "Elektronika", "Laptopy").
- string **Slug** = null! `[get, set]`

  Przyjazny dla wyszukiwarek tekstowy identyfikator URL (slug).
- string? **Description** `[get, set]`

  Opcjonalny opis charakterystyki lub przeznaczenia danej kategorii.

8.61.1 Opis szczegółowy

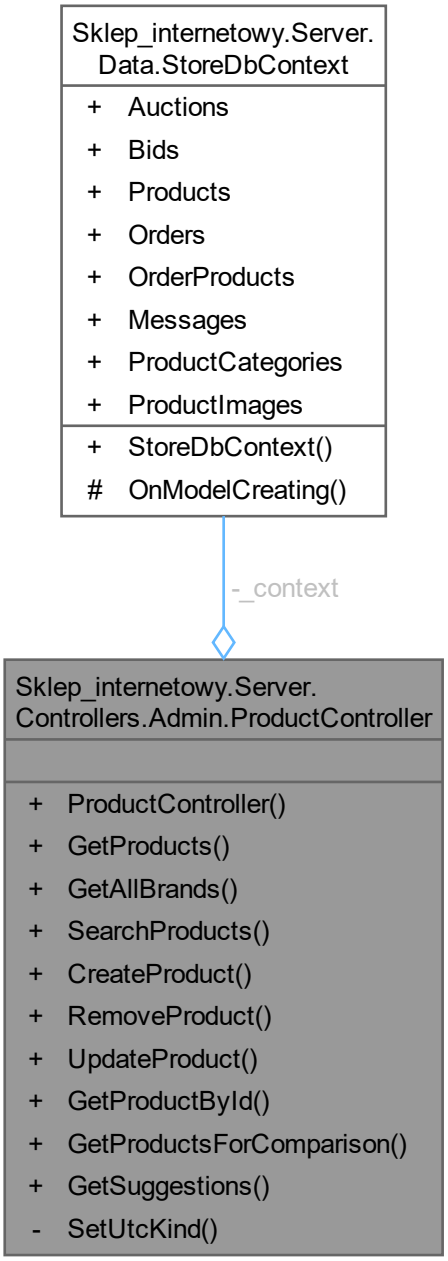
Obiekt transferu danych (DTO) reprezentujący kategorię produktów w systemie. Służy do przesyłania podstawowych informacji o grupach towarowych, wykorzystywanych w menu nawigacyjnym, filtrach oraz przy przypisywaniu produktów do katalogu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/ProductCategoryDto.cs

8.62 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.ProductController

Kontroler administracyjny odpowiedzialny za pelne zarzadzanie asortymentem produktow. Obsluguje operacje CRUD, przesyłanie plikow graficznych, wyszukiwanie oraz system sugestii.

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.ProductController:



Metody publiczne

- ProductController (StoreDbContext context, IWebHostEnvironment environment)

  Inicjalizuje nowa instancje klasy ProductController.
- async Task< ActionResult< IEnumerable< ProductDto > > > GetProducts ()

  Pobiera pelna liste wszystkich produktow wraz z ich kategoriami i zdjeciami.
- async Task< ActionResult< IEnumerable< string > > > GetAllBrands ()

Pobiera liste wszystkich unikalnych marek produktow dostepnych w bazie danych. Metoda dostepna publicznie (AllowAnonymous) dla potrzeb filtrowania na froncie.

- async Task< ActionResult< IEnumerable< ProductDto > > > SearchProducts ([FromQuery] string q)

  Wyszukuje produkty na podstawie frazy tekstowej dopasowanej do nazwy, marki lub opisu.
- async Task< ActionResult > CreateProduct ([FromForm] CreateProductWithFilesDto formDto)

  Tworzy nowy produkt w systemie wraz z obsluga przesyłania wielu plikow graficznych.
- async Task< ActionResult > RemoveProduct (RemoveProductDto removeProductDto)

  Usuwa wybrany produkt z bazy danych na podstawie identyfikatora.
- async Task< ActionResult > UpdateProduct (UpdateProductDto updateProductDto)

  Aktualizuje dane techniczne i cenowe istniejacego produktu.
- async Task< ActionResult< ProductDto > > GetProductById (int id)

  Pobiera szczegolowe dane pojedynczego produktu.
- async Task< ActionResult< IEnumerable< ProductDto > > > GetProductsForComparison ([FromBody] List< int > productIds)

  Pobiera liste dwoch unikalnych produktow do porownania na podstawie przekazanych identyfikatorow.
- async Task< ActionResult > GetSuggestions ([FromQuery] string q)

  Generuje dynamiczne sugestie wyszukiwania (kategorie i produkty) w czasie rzeczywistym.

Metody prywatne

- DateTime? SetUtcKind (DateTime? date)

  Metoda pomocnicza ustawiajaca rodzaj daty na UTC, wymagana dla kompatybilnosci z baza danych.

8.62.1 Opis szczegółowy

Kontroler administracyjny odpowiedzialny za pelne zarzadzanie asortymentem produktow. Obsluguje operacje CRUD, przesyłanie plikow graficznych, wyszukiwanie oraz system sugestii.

8.62.2 Dokumentacja konstruktora i destruktora

8.62.2.1 ProductController()

```
Sklep_internetowy.Server.Controllers.Admin.ProductController.ProductController (
            StoreDbContext context,
            IWebHostEnvironment environment)  [inline]
```

Inicjalizuje nowa instancje klasy ProductController.

Parametry

context	Kontekst bazy danych StoreDbContext.
environment	Srodowisko serwerowe do obslugi systemu plikow (przesyłanie zdjec).

8.62.3 Dokumentacja funkcji składowych

8.62.3.1 CreateProduct()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.ProductController.↩
CreateProduct (
            [FromForm] CreateProductWithFilesDto formDto)  [inline]
```

Tworzy nowy produkt w systemie wraz z obsluga przesyłania wielu plikow graficznych.

Parametry

formDto	Dane produktu przekazane w formacie multipart/form-data.

Zwraca

Status 200 OK po pomyslnym utworzeniu i zapisaniu plikow.

8.62.3.2 GetAllBrands()

```
async Task< ActionResult< IEnumerable< string > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductController.GetAllBrands ()  [inline]
```

Pobiera liste wszystkich unikalnych marek produktow dostepnych w bazie danych. Metoda dostepna publicznie (AllowAnonymous) dla potrzeb filtrowania na froncie.

Zwraca

Lista ciagow znakow reprezentujacych nazwy marek.

8.62.3.3 GetProductById()

```
async Task< ActionResult< ProductDto > > Sklep_internetowy.Server.Controllers.Admin.Product↩
Controller.GetProductById (
            int id)  [inline]
```

Pobiera szczegolowe dane pojedynczego produktu.

Parametry

id	ID produktu.

8.62.3.4 GetProducts()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductController.GetProducts ()  [inline]
```

Pobiera pelna liste wszystkich produktow wraz z ich kategoriami i zdjeciami.

Zwraca

Kolekcja obiektow ProductDto zawierajaca kompletne dane o produktach.

<response code="200">Zwraca liste produktow.</response> <response code="500">Gdy wystapi blad serwera podczas pobierania danych.</response>

8.62.3.5 GetProductsForComparison()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductController.GetProductsForComparison (
            [FromBody] List< int > productIds)  [inline]
```

Pobiera liste dwoch unikalnych produktow do porownania na podstawie przekazanych identyfikatorow.

Parametry

productIds	Lista identyfikatorow ID (maksymalnie 2 unikalne zostana przetworzone).

8.62.3.6 GetSuggestions()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.ProductController.Get↩
Suggestions (
            [FromQuery] string q)  [inline]
```

Generuje dynamiczne sugestie wyszukiwania (kategorie i produkty) w czasie rzeczywistym.

Parametry

q	Fragment nazwy wpisany przez uzytkownika.

Zwraca

Obiekt zawierajacy liste nazw kategorii oraz liste dopasowanych obiektow ProductDto.

8.62.3.7 RemoveProduct()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.ProductController.↩
RemoveProduct (
            RemoveProductDto removeProductDto)  [inline]
```

Usuwa wybrany produkt z bazy danych na podstawie identyfikatora.

Parametry

removeProductDto	DTO zawierajace identyfikator ID produktu do usuniecia.

8.62.3.8 SearchProducts()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Admin.ProductController.SearchProducts (
            [FromQuery] string q)  [inline]
```

Wyszukuje produkty na podstawie frazy tekstowej dopasowanej do nazwy, marki lub opisu.

Parametry

q	Zapytanie wyszukiwania wpisane przez uzytkownika.

Zwraca

Kolekcja produktow spelniajacych kryteria wyszukiwania.

8.62.3.9 SetUtcKind()

```
DateTime? Sklep_internetowy.Server.Controllers.Admin.ProductController.SetUtcKind (
            DateTime? date)  [inline], [private]
```

Metoda pomocnicza ustawiajaca rodzaj daty na UTC, wymagana dla kompatybilnosci z baza danych.

Parametry

date	Data do przetworzenia.

Zwraca

Data z rodzajem DateTimeKind.Utc lub null.

8.62.3.10 UpdateProduct()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.ProductController.↩
UpdateProduct (
            UpdateProductDto updateProductDto)  [inline]
```

Aktualizuje dane techniczne i cenowe istniejacego produktu.

Parametry

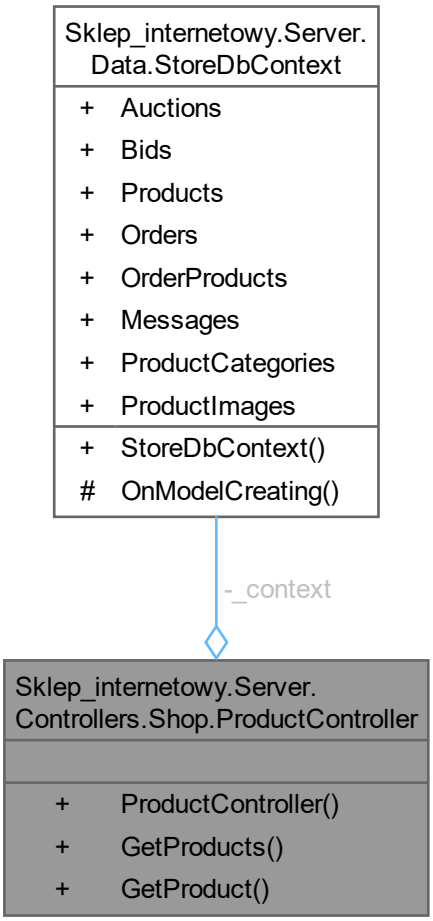
updateProductDto	Obiekt zawierajacy zaktualizowane pola produktu.

<response code="204">Gdy aktualizacja przebiegla pomyslnie.</response> <response code="404">Gdy produkt nie figuruje w bazie danych.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/ProductController.cs

8.63 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Shop.ProductController

Kontroler API odpowiedzialny za dostarczanie danych o produktach dla czesci sklepowej (ogolnodostepnej). Obsluguje operacje przegladania katalogu produktow oraz wyswietlania szczegolowych informacji o konkretnym towarze.
Diagram współpracy dla Sklep_internetowy.Server.Controllers.Shop.ProductController:



Metody publiczne

- ProductController (StoreDbContext context)

  Inicjalizuje nowa instancje klasy ProductController.
- async Task< ActionResult< IEnumerable< ProductDto > > > GetProducts ()

  Pobiera liste wszystkich produktow zarejestrowanych w systemie wraz z ich kategoriami i powiazanymi zdjeciami.
- async Task< ActionResult< ProductDto > > GetProduct (int id)

  Pobiera szczegolowe dane o pojedynczym produkcie na podstawie jego unikalnego identyfikatora ID.

8.63.1 Opis szczegółowy

Kontroler API odpowiedzialny za dostarczanie danych o produktach dla czesci sklepowej (ogolnodostepnej). Obsluguje operacje przegladania katalogu produktow oraz wyswietlania szczegolowych informacji o konkretnym towarze.

8.63.2 Dokumentacja konstruktora i destruktora

8.63.2.1 ProductController()

```
Sklep_internetowy.Server.Controllers.Shop.ProductController.ProductController (
            StoreDbContext context)  [inline]
```

Inicjalizuje nowa instancje klasy ProductController.

Parametry

context	Kontekst bazy danych StoreDbContext do obslugi zapytan.

8.63.3 Dokumentacja funkcji składowych

8.63.3.1 GetProduct()

```
async Task< ActionResult< ProductDto > > Sklep_internetowy.Server.Controllers.Shop.Product↩
Controller.GetProduct (
            int id)  [inline]
```

Pobiera szczegolowe dane o pojedynczym produkcie na podstawie jego unikalnego identyfikatora ID.

Parametry

id	Identyfikator techniczny produktu.

Zwraca

Obiekt ProductDto reprezentujacy szczegoly towaru lub NotFound w przypadku braku rekordu.

<response code="200">Zwraca dane wybranego produktu.</response> <response code="404">Gdy produkt o podanym ID nie istnieje w bazie danych.</response>

8.63.3.2 GetProducts()

```
async Task< ActionResult< IEnumerable< ProductDto > > > Sklep_internetowy.Server.Controllers.↩
Shop.ProductController.GetProducts ()  [inline]
```

Pobiera liste wszystkich produktow zarejestrowanych w systemie wraz z ich kategoriami i powiazanymi zdjeciami.

Zwraca

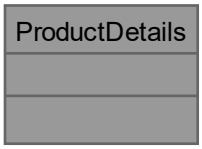
Kolekcja obiektow ProductDto zawierajaca kompletne dane o cenach, znizkach i mediach.

<response code="200">Zwraca liste produktow dostepnych w sklepie.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Shop/ProductController.cs

8.64 Dokumentacja klasy ProductDetails

Diagram współpracy dla ProductDetails:



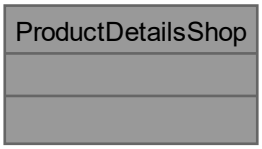
8.64.1 Opis szczegółowy

@description Zarzadza stanem pojedynczego produktu, obsluguje asynchroniczne pobieranie danych oraz koordynuje procesy aktualizacji i usuwania rekordow poprzez API.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/ProductDetails.jsx

8.65 Dokumentacja klasy ProductDetailsShop

Diagram współpracy dla ProductDetailsShop:



8.65.1 Opis szczegółowy

@description Renderuje widok szczegolowy produktu, umozliwiajac jego zakup lub dodanie do porownania.

Parametry

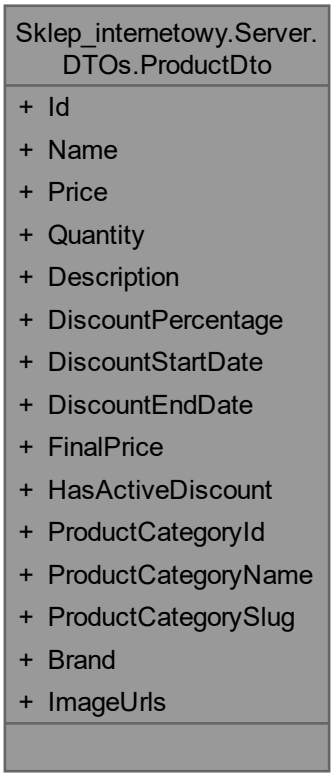
{Object}	props - Wlasciwosci komponentu.
{Object}	props.comparison - Obiekt stanu i funkcji porownywarki przekazywany z komponentu nadrzednego.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/Shop/ProductDetailsShop.jsx

8.66 Dokumentacja klasy Sklep_internetowy.Server.DTOs.ProductDto

Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o produkcie. Zawiera kompletny zestaw danych o towarze, w tym parametry cenowe, logikę promocyjną, przynależność do kategorii oraz powiązane zasoby multimedialne.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.ProductDto:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator produktu w systemie.
- string **Name** = null! `[get, set]`

  Nazwa handlowa produktu.
- decimal **Price** `[get, set]`

  Podstawowa cena jednostkowa brutto przed naliczeniem rabatów.
- int **Quantity** `[get, set]`

  Aktualna liczba sztuk produktu dostępna w magazynie.
- string? **Description** `[get, set]`

  Szczegółowy opis techniczny lub marketingowy produktu.
- decimal? **DiscountPercentage** `[get, set]`

  Wartość procentowa rabatu przypisana do produktu (jeśli dotyczy).
- DateTime? **DiscountStartDate** `[get, set]`

  Data i godzina rozpoczęcia okresu obowiązywania ceny promocyjnej.
- DateTime? **DiscountEndDate** `[get, set]`

Data i godzina zakończenia okresu obowiązywania ceny promocyjnej.

- decimal **FinalPrice** `[get, set]`

  Obliczona cena końcowa po uwzględnieniu aktywnych rabatów.
- bool **HasActiveDiscount** `[get, set]`

  Flaga logiczna określająca, czy produkt jest obecnie objęty aktywną promocją.
- int **ProductCategoryId** `[get, set]`

  Identyfikator techniczny kategorii, do której przypisano produkt.
- string **ProductCategoryName** = null! `[get, set]`

  Nazwa kategorii nadrzędnej dla celów prezentacyjnych.
- string **ProductCategorySlug** = null! `[get, set]`

  Tekstowy identyfikator URL kategorii nadrzędnej (slug).
- string **Brand** = null! `[get, set]`

  Marka lub producent danego towaru.
- List< string > **ImageUrls** = new List<string>() `[get, set]`

  Kolekcja adresów URL prowadzących do zdjęć produktu.

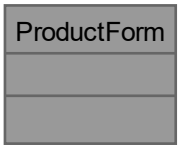
8.66.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o produkcie. Zawiera kompletny zestaw danych o towarze, w tym parametry cenowe, logikę promocyjną, przynależność do kategorii oraz powiązane zasoby multimedialne.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/ProductDto.cs

8.67 Dokumentacja klasy ProductForm

Diagram współpracy dla ProductForm:



8.67.1 Opis szczegółowy

@description Zarzadza stanem formularza produktu, walidacja pol oraz przygotowaniem danych do wysylki (w tym plikow).

Parametry

{Object}	props - Wlasciwosci komponentu.
{Function}	props.onSubmit - Funkcja wywolywana przy zapisie, przyjmuje (data, files).
{Function}	props.onCancel - Funkcja anulujaca edycje/tworzenie.
{Object}	props.initialData - Dane istniejacego produktu uzywane do wypelnienia pol w trybie edycji.

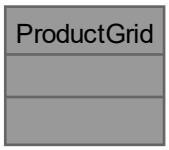
{Boolean}	props.isEditing - Flaga przelaczajaca naglowki i etykiety przyciskow miedzy trybem tworzenia a edycji.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/admin/product/ProductForm.jsx

8.68 Dokumentacja klasy ProductGrid

Glowny komponent siatki produktow.
Diagram współpracy dla ProductGrid:



8.68.1 Opis szczegółowy

Glowny komponent siatki produktow.
Odpowiada za pobieranie danych, paginacje, wybor produktu promocyjnego oraz obsluge klikniec w produkty.

Zwraca

JSX.Element Element siatki produktow.

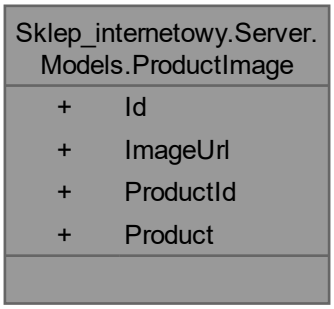
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/productGrid/ProductGrid.jsx

8.69 Dokumentacja klasy Sklep_internetowy.Server.Models.ProductImage

Klasa reprezentująca zdjęcie przypisane do konkretnego produktu w systemie TechStore. Przechowuje informacje o lokalizacji pliku graficznego oraz relację z produktem, co umożliwia renderowanie galerii zdjęć w części frontendowej sklepu.

Diagram współpracy dla Sklep_internetowy.Server.Models.ProductImage:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator rekordu zdjęcia w bazie danych.
- string **ImageUrl** = null! `[get, set]`

  Adres URL lub ścieżka do pliku graficznego (np. /uploads/image.jpg).
- int **ProductId** `[get, set]`

  Identyfikator produktu, do którego przypisane jest to zdjęcie.
- Product **Product** = null! `[get, set]`

  Obiekt nawigacyjny powiązanego produktu, umożliwiający dostęp do jego szczegółów.

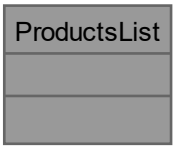
8.69.1 Opis szczegółowy

Klasa reprezentująca zdjęcie przypisane do konkretnego produktu w systemie TechStore. Przechowuje informacje o lokalizacji pliku graficznego oraz relację z produktem, co umożliwia renderowanie galerii zdjęć w części frontendowej sklepu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/ProductImage.cs

8.70 Dokumentacja klasy ProductsList

Diagram współpracy dla ProductsList:



8.70.1 Opis szczegółowy

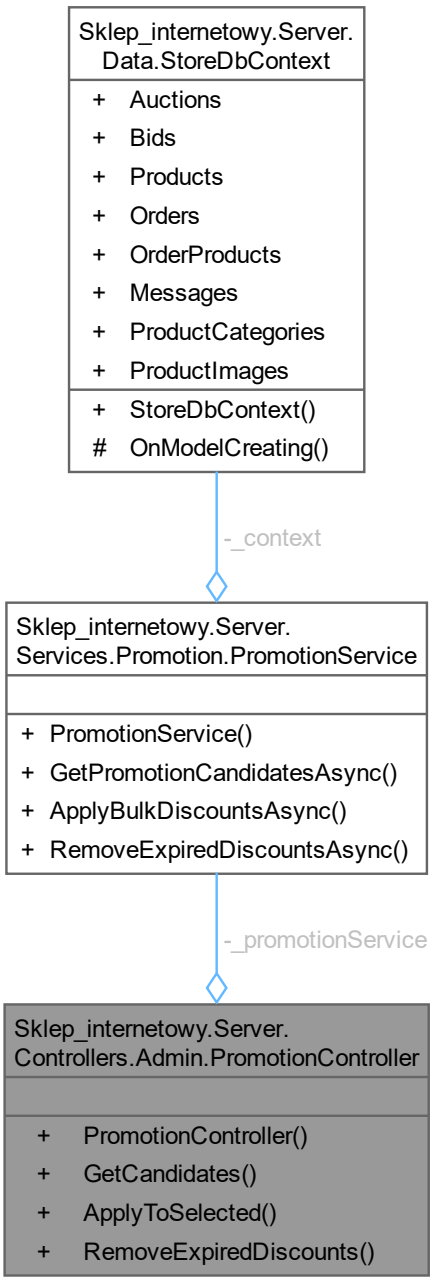
@description Glowny komponent widoku listy produktow. Zarzadza stanem kolekcji, ladowaniem danych oraz komunikatami o bledach.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/ProductList.jsx

8.71 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.PromotionController

Kontroler administracyjny odpowiedzialny za zarzadzanie kampaniami promocyjnymi. Umozliwia identyfikacje produktow kwalifikujacych sie do obnizek oraz masowe nakladanie rabatow.

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.PromotionController:



Metody publiczne

- PromotionController (PromotionService promotionService)

  Inicjalizuje nowa instancje klasy PromotionController.
- async Task< ActionResult > GetCandidates ([FromQuery] int? categoryId, [FromQuery] int minStock=10, [FromQuery] int daysInactive=60)

  Pobiera liste produktow spelniajacych kryteria do objecia nowa promocja.
- async Task< ActionResult > ApplyToSelected ([FromBody] ApplyBulkDiscountRequest request)

Naklada rabat procentowy na wybrana grupe produktow na okreslony czas.

- async Task< ActionResult > RemoveExpiredDiscounts ()

  Usuwa dane o znizkach dla wszystkich produktow, ktorych czas trwania promocji juz uplynal.

8.71.1 Opis szczegółowy

Kontroler administracyjny odpowiedzialny za zarzadzanie kampaniami promocyjnymi. Umozliwia identyfikacje produktow kwalifikujacych sie do obnizek oraz masowe nakladanie rabatow.

8.71.2 Dokumentacja konstruktora i destruktora

8.71.2.1 PromotionController()

```
Sklep_internetowy.Server.Controllers.Admin.PromotionController.PromotionController (
            PromotionService promotionService)  [inline]
```

Inicjalizuje nowa instancje klasy PromotionController.

Parametry

promotionService	Serwis obslugujacy logike biznesowa promocji.

8.71.3 Dokumentacja funkcji składowych

8.71.3.1 ApplyToSelected()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.PromotionController.↩
ApplyToSelected (
            [FromBody] ApplyBulkDiscountRequest request)  [inline]
```

Naklada rabat procentowy na wybrana grupe produktow na okreslony czas.

Parametry

request	Obiekt zawierajacy liste ID produktow, procent rabatu oraz czas trwania promocji w dniach.

Zwraca

Informacja o liczbie pomyslnie zaktualizowanych produktow.

<response code="200">Promocja zostala pomyslnie nalozona.</response> <response code="400">Gdy nie wybrano zadnych produktow w zadaniu.</response>

8.71.3.2 GetCandidates()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.PromotionController.↩
GetCandidates (
            [FromQuery] int? categoryId,
            [FromQuery] int minStock = 10,
            [FromQuery] int daysInactive = 60)  [inline]
```

Pobiera liste produktow spelniajacych kryteria do objecia nowa promocja.

Parametry

categoryId	Opcjonalny identyfikator kategorii do filtrowania.
minStock	Minimalny stan magazynowy wymagany do wyswietlenia produktu (domyslnie 10).
daysInactive	Liczba dni bez aktywnosci sprzedazowej (domyslnie 60).

Zwraca

Kolekcja obiektow DTO reprezentujacych kandydatow do promocji.

<response code="200">Zwraca liste kandydatow.</response> <response code="500">W przypadku bledu serwera podczas pobierania danych.</response>

8.71.3.3 RemoveExpiredDiscounts()

```
async Task< ActionResult > Sklep_internetowy.Server.Controllers.Admin.PromotionController.↩
RemoveExpiredDiscounts ()  [inline]
```

Usuwa dane o znizkach dla wszystkich produktow, ktorych czas trwania promocji juz uplynal.

Zwraca

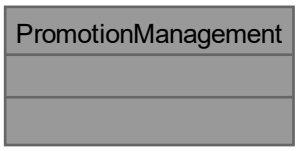
Komunikat o liczbie produktow, z ktorych usunieto wygasle rabaty.

<response code="200">Wygasle promocje zostaly pomyslnie usuniete z bazy danych.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/PromotionController.cs

8.72 Dokumentacja klasy PromotionManagement

Diagram współpracy dla PromotionManagement:



8.72.1 Opis szczegółowy

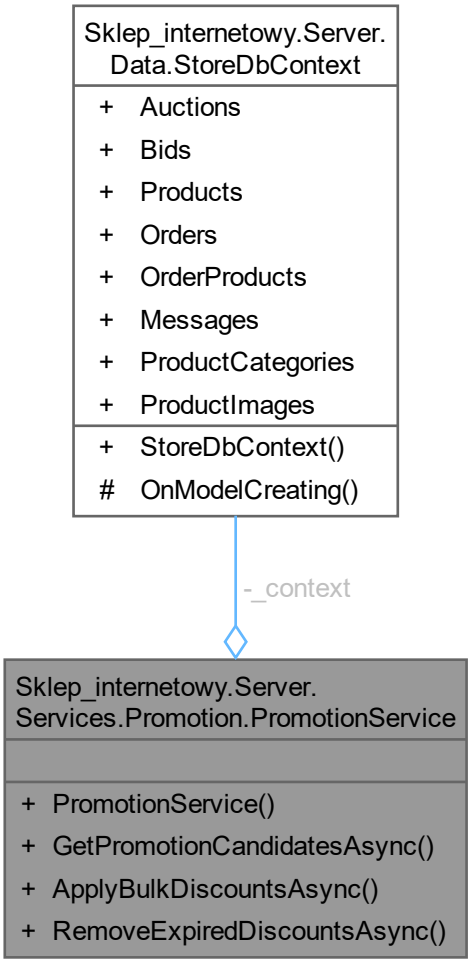
@description Glowny widok administracyjny do konfiguracji i aktywacji promocji. Zarzadza procesem wyszukiwania kandydatow, ich selekcji oraz komunikacji z API promocji.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/AdminDashboard/promotion/PromotionManagement.jsx

8.73 Dokumentacja klasy Sklep_internetowy.Server.Services.Promotion.PromotionService

Serwis biznesowy odpowiedzialny za zaawansowane zarządzanie akcjami promocyjnymi w systemie TechStore. Klasa dostarcza metody do identyfikacji produktów o niskiej rotacji, masowego nakładania rabatów oraz automatycznego czyszczenia bazy danych z wygasłych ofert cenowych.

Diagram współpracy dla Sklep_internetowy.Server.Services.Promotion.PromotionService:



Metody publiczne

- PromotionService (StoreDbContext context)

  Inicjalizuje nową instancję klasy PromotionService.
- async Task< IEnumerable< ProductDto > > GetPromotionCandidatesAsync (int? categoryId, int minStock, int daysSinceLastSale)

  Analizuje stan magazynowy i wyszukuje produkty kwalifikujące się do objęcia nową akcją promocyjną.
- async Task< int > ApplyBulkDiscountsAsync (List< int > productIds, int percentage, int durationDays)

  Wykonuje operację masowej aktualizacji cen (Bulk Update) dla wybranych produktów.
- async Task< int > RemoveExpiredDiscountsAsync ()

  Przeprowadza procedurę czyszczenia rekordów promocyjnych, których termin ważności upłynął.

8.73.1 Opis szczegółowy

Serwis biznesowy odpowiedzialny za zaawansowane zarządzanie akcjami promocyjnymi w systemie TechStore. Klasa dostarcza metody do identyfikacji produktów o niskiej rotacji, masowego nakładania rabatów oraz automatycznego czyszczenia bazy danych z wygasłych ofert cenowych.

8.73.2 Dokumentacja konstruktora i destruktora

8.73.2.1 PromotionService()

```
Sklep_internetowy.Server.Services.Promotion.PromotionService.PromotionService (
            StoreDbContext context)  [inline]
```

Inicjalizuje nową instancję klasy PromotionService.

Parametry

context	Kontekst bazy danych umożliwiający dostęp do repozytorium produktów i kategorii.

8.73.3 Dokumentacja funkcji składowych

8.73.3.1 ApplyBulkDiscountsAsync()

```
async Task< int > Sklep_internetowy.Server.Services.Promotion.PromotionService.ApplyBulk↩
DiscountsAsync (
            List< int > productIds,
            int percentage,
            int durationDays)  [inline]
```

Wykonuje operację masowej aktualizacji cen (Bulk Update) dla wybranych produktów.
Metoda automatycznie wyznacza datę rozpoczęcia (bieżący czas UTC) oraz datę zakończenia na podstawie przekazanego parametru czasu trwania. Wszystkie zmiany są zatwierdzane w ramach jednej transakcji.

Parametry

productIds	Lista unikalnych identyfikatorów produktów objętych przeceną.
percentage	Wartość procentowa rabatu (np. 15 dla 15% zniżki).
durationDays	Liczba dni, przez które promocja ma pozostać aktywna.

Zwraca

Liczba rekordów pomyślnie zaktualizowanych w bazie danych.

8.73.3.2 GetPromotionCandidatesAsync()

```
async Task< IEnumerable< ProductDto > > Sklep_internetowy.Server.Services.Promotion.Promotion↩
Service.GetPromotionCandidatesAsync (
            int? categoryId,
            int minStock,
            int daysSinceLastSale)  [inline]
```

Analizuje stan magazynowy i wyszukuje produkty kwalifikujące się do objęcia nową akcją promocyjną.
Metoda filtruje produkty, które posiadają stan magazynowy powyżej określonego progu i nie są obecnie objęte żadną aktywną zniżką (sprawdzane są ramy czasowe oraz status pola DiscountPercentage).

Parametry

categoryId	Opcjonalny identyfikator kategorii, do której ma zostać ograniczone wyszukiwanie.
minStock	Minimalna liczba sztuk produktu wymagana, aby system uznał go za kandydata do promocji.
daysSinceLastSale	Parametr określający czas braku aktywności sprzedażowej (liczba dni).

Zwraca

Kolekcja obiektów ProductDto reprezentujących wyselekcjonowane produkty.

8.73.3.3 RemoveExpiredDiscountsAsync()

```
async Task< int > Sklep_internetowy.Server.Services.Promotion.PromotionService.RemoveExpired↩
DiscountsAsync () [inline]
```

Przeprowadza procedurę czyszczenia rekordów promocyjnych, których termin ważności upłynął.
Metoda wyszukuje produkty z datą końcową mniejszą niż aktualny czas systemowy i resetuje wszystkie pola związane z rabatem do wartości domyślnych (null).

Zwraca

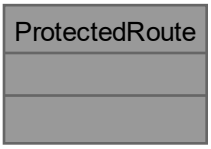
Liczba produktów, z których pomyślnie usunięto wygasłe promocje.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Services/Promotion/PromotionService.cs

8.74 Dokumentacja klasy ProtectedRoute

Komponent ochrony tras dla administratora.
Diagram współpracy dla ProtectedRoute:



8.74.1 Opis szczegółowy

Komponent ochrony tras dla administratora.

Parametry

{Object}	children Komponenty potomne do wyrenderowania.

Zwraca

JSX.Element Chroniony komponent lub przekierowanie.

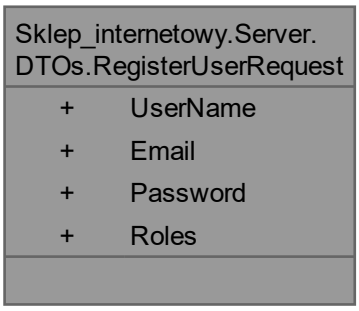
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/ProtectedRoute.jsx

8.75 Dokumentacja klasy Sklep_internetowy.Server.DTOs.RegisterUserRequest

Obiekt transferu danych (DTO) reprezentujący żądanie rejestracji nowego użytkownika w systemie TechStore. Klasa służy do przesyłania kompletu informacji niezbędnych do zainicjowania procesu tworzenia konta użytkownika, obejmując dane identyfikacyjne, uwierzytelniające oraz definicję ról dostępu.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.RegisterUserRequest:



Właściwości

- string **UserName** = null! `[get, set]`

  Unikalna nazwa użytkownika (login) wybrana podczas rejestracji.
- string **Email** = null! `[get, set]`

  Adres e-mail użytkownika służący do komunikacji systemowej, powiadomień oraz odzyskiwania hasła.
- string **Password** = null! `[get, set]`

  Hasło w formie jawnej, przesyłane do warstwy serwerowej w celu bezpiecznego zahaszowania przed zapisem w bazie danych.
- IList< string > **Roles** = new List<string>() `[get, set]`

  Kolekcja nazw ról systemowych (np. "Admin", "User"), które mają zostać przypisane do nowego konta w celu zdefiniowania zakresu uprawnień w systemie.

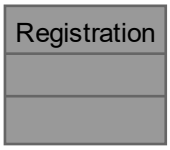
8.75.1 Opis szczegółowy

Obiekt transferu danych (DTO) reprezentujący żądanie rejestracji nowego użytkownika w systemie TechStore. Klasa służy do przesyłania kompletu informacji niezbędnych do zainicjowania procesu tworzenia konta użytkownika, obejmując dane identyfikacyjne, uwierzytelniające oraz definicję ról dostępu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/RegisterUserRequest.cs

8.76 Dokumentacja klasy Registration

Diagram współpracy dla Registration:



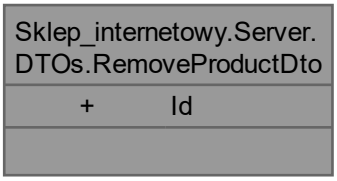
8.76.1 Opis szczegółowy

@description Renderuje interfejs rejestracji. Zarzadza lokalnym stanem formularza, komunikatami o bledach/↩ sukcesach oraz przekierowaniem po poprawnym utworzeniu konta.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Registration/Registration.jsx

8.77 Dokumentacja klasy Sklep_internetowy.Server.DTOs.RemoveProductDto

Obiekt transferu danych (DTO) wykorzystywany w procesie usuwania produktu z systemu. Zawiera minimalny zestaw danych niezbędny do jednoznacznej identyfikacji zasobu przeznaczonego do usunięcia z bazy danych.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.RemoveProductDto:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator techniczny produktu (ID), który ma zostać usunięty.

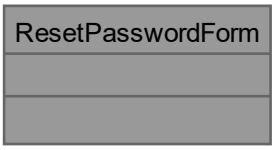
8.77.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany w procesie usuwania produktu z systemu. Zawiera minimalny zestaw danych niezbędny do jednoznacznej identyfikacji zasobu przeznaczonego do usunięcia z bazy danych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/RemoveProductDto.cs

8.78 Dokumentacja klasy ResetPasswordForm

Diagram współpracy dla ResetPasswordForm:



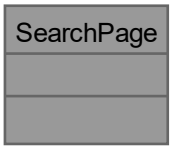
8.78.1 Opis szczegółowy

@description Renderuje interfejs do zmiany zapomnianego hasla. Zarzadza lokalnym stanem nowych poswiadczen i obsluguje logike wysylania danych do serwera.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/ResetPassword/ResetPasswordPage.jsx

8.79 Dokumentacja klasy SearchPage

Diagram współpracy dla SearchPage:



8.79.1 Opis szczegółowy

@description Glowny komponent strony wyszukiwania. Zarzadza stanem wynikow pobranych z API, lista producentow oraz kompleksowa logika filtrow po stronie klienta.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/Products/Shop/SearchPage.jsx

8.80 Dokumentacja klasy SimpleReceipt

Komponent renderujacy uproszczony paragon fiskalny.
Diagram współpracy dla SimpleReceipt:

8.80.1 Opis szczegółowy

Komponent renderujacy uproszczony paragon fiskalny.

Parametry

{Object}	props Wlasciwosci komponentu.
{Object}	props.order Obiekt zamowienia zawierajacy liste produktow.

Zwraca

{JSX.Element} Obiekt dokumentu PDF.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/components/admin/InvoiceGenerator/InvoiceDocument.jsx

8.81 Dokumentacja klasy Sklep_internetowy.Server.Data.StoreDbContext

Glowny kontekst bazy danych aplikacji TechStore. Klasa dziedziczy po IdentityDbContext, co integruje system ASP.NET Core Identity z modelem danych aplikacji, umozliwiajac zarzadzanie uzytkownikami i rolami.
Diagram współpracy dla Sklep_internetowy.Server.Data.StoreDbContext:

Metody publiczne

- StoreDbContext (DbContextOptions< StoreDbContext > options)

  Inicjalizuje nowa instancje klasy StoreDbContext.

Metody chronione

- override void OnModelCreating (ModelBuilder modelBuilder)

  Konfiguruje model encji i relacje miedzy nimi podczas inicjalizacji bazy danych. Definiuje klucze zlozone, zachowania usuwania kaskadowego oraz dane startowe (Seed Data).

Właściwości

- DbSet< Auction > **Auctions** `[get]`

  Zestaw danych reprezentujacy aukcje (licytacje) w systemie.
- DbSet< Bid > **Bids** `[get]`

  Zestaw danych reprezentujacy poszczegolne oferty zlozone w aukcjach.
- DbSet< Product > **Products** `[get]`

  Zestaw danych reprezentujacy produkty dostepne w katalogu sklepu.
- DbSet< Order > **Orders** `[get]`

  Rejestr zamowien zlozonych przez klientow.
- DbSet< OrderProduct > **OrderProducts** `[get]`

  Tabela laczaca produkty z zamowieniami (relacja wiele-do-wielu).
- DbSet< UserMessage > **Messages** `[get, set]`

  Wiadomosci kontaktowe przeslane przez uzytkownikow systemu.
- DbSet< ProductCategory > **ProductCategories** `[get, set]`

  Kategorie, do ktorych przypisane sa produkty.
- DbSet< ProductImage > **ProductImages** `[get, set]`

  Zdjecia i grafiki przypisane do konkretnych produktow.

8.81.1 Opis szczegółowy

Glowny kontekst bazy danych aplikacji TechStore. Klasa dziedziczy po IdentityDbContext, co integruje system ASP.NET Core Identity z modelem danych aplikacji, umozliwiajac zarzadzanie uzytkownikami i rolami.

8.81.2 Dokumentacja konstruktora i destruktora

8.81.2.1 StoreDbContext()

```
Sklep_internetowy.Server.Data.StoreDbContext.StoreDbContext (
            DbContextOptions< StoreDbContext > options)  [inline]
```

Inicjalizuje nowa instancje klasy StoreDbContext.

Parametry

options	Opcje konfiguracji kontekstu przekazywane przez system Dependency Injection.

8.81.3 Dokumentacja funkcji składowych

8.81.3.1 OnModelCreating()

```
override void Sklep_internetowy.Server.Data.StoreDbContext.OnModelCreating (
            ModelBuilder modelBuilder)  [inline], [protected]
```

Konfiguruje model encji i relacje miedzy nimi podczas inicjalizacji bazy danych. Definiuje klucze zlozone, zachowania usuwania kaskadowego oraz dane startowe (Seed Data).

Parametry

modelBuilder	Obiekt sluzacy do budowania modelu bazy danych.

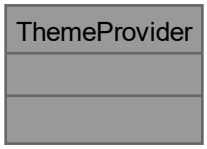
8.81.4 SeedData

Inicjalizacja bazy danych wstepnymi danymi testowymi.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Data/StoreDbContext.cs

8.82 Dokumentacja klasy ThemeProvider

Provider kontekstu motywu i rozmiaru czcionki dla calej aplikacji.
Diagram współpracy dla ThemeProvider:



8.82.1 Opis szczegółowy

Provider kontekstu motywu i rozmiaru czcionki dla calej aplikacji.

Parametry

{Object}	children Komponenty potomne.

Zwraca

JSX.Element Provider kontekstu motywu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/context/ThemeContext.jsx

8.83 Dokumentacja klasy Sklep_internetowy.Server.DTOs.UpdateProductDto

Obiekt transferu danych (DTO) służący do aktualizacji parametrów istniejącego produktu. Klasa zawiera kompletny zestaw właściwości niezbędnych do modyfikacji danych technicznych, logiki cenowej oraz przypisań kategoryzacyjnych towaru w systemie TechStore.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.UpdateProductDto:

Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator produktu (ID) podlegającego aktualizacji.
- string **Name** = null! `[get, set]`

  Zaktualizowana nazwa handlowa produktu.
- decimal **Price** `[get, set]`

  Nowa podstawowa cena jednostkowa towaru.
- int **Quantity** `[get, set]`

  Zaktualizowana liczba sztuk produktu dostępna na stanie magazynowym.
- string? **Description** `[get, set]`

  Nowy opis charakterystyki lub specyfikacji technicznej produktu.
- decimal? **DiscountPercentage** `[get, set]`

  Wartość procentowa rabatu w przedziale od 0 do 100.
- DateTime? **DiscountStartDate** `[get, set]`

  Data i godzina rozpoczęcia nowego okresu promocyjnego.
- DateTime? **DiscountEndDate** `[get, set]`

  Data i godzina zakończenia nowego okresu promocyjnego.
- int **ProductCategoryId** `[get, set]`

  Identyfikator nowej kategorii nadrzędnej, do której ma zostać przypisany produkt.
- string **Brand** = null! `[get, set]`

  Nazwa marki lub producenta produktu.

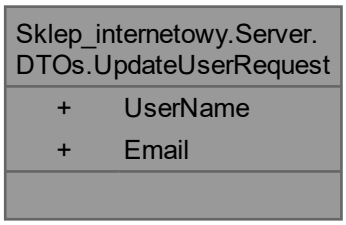
8.83.1 Opis szczegółowy

Obiekt transferu danych (DTO) służący do aktualizacji parametrów istniejącego produktu. Klasa zawiera kompletny zestaw właściwości niezbędnych do modyfikacji danych technicznych, logiki cenowej oraz przypisań kategoryzacyjnych towaru w systemie TechStore.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/UpdateProductDto.cs

8.84 Dokumentacja klasy Sklep_internetowy.Server.DTOs.UpdateUserRequest

Obiekt transferu danych (DTO) służący do aktualizacji podstawowych informacji o koncie użytkownika. Klasa ta jest wykorzystywana w procesie modyfikacji danych profilowych, umożliwiając synchronizację zmian nazwy użytkownika oraz adresu e-mail pomiędzy warstwą prezentacji a bazą danych.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.UpdateUserRequest:



Właściwości

- string **UserName** `[get, set]`

  Zaktualizowana nazwa użytkownika (login) służąca do identyfikacji konta w systemie.
- string **Email** `[get, set]`

  Nowy adres e-mail użytkownika, wykorzystywany do komunikacji systemowej i powiadomień.

8.84.1 Opis szczegółowy

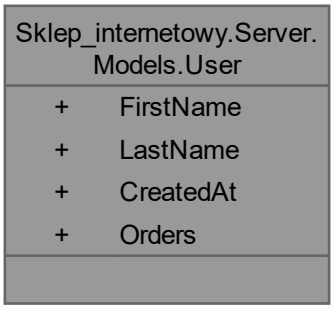
Obiekt transferu danych (DTO) służący do aktualizacji podstawowych informacji o koncie użytkownika. Klasa ta jest wykorzystywana w procesie modyfikacji danych profilowych, umożliwiając synchronizację zmian nazwy użytkownika oraz adresu e-mail pomiędzy warstwą prezentacji a bazą danych.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/UpdateUserRequest.cs

8.85 Dokumentacja klasy Sklep_internetowy.Server.Models.User

Rozszerzona klasa użytkownika systemu, dziedzicząca po IdentityUser. Przechowuje dodatkowe dane profilowe, takie jak imię i nazwisko, oraz zarządza relacją z historią zamówień klienta.

Diagram współpracy dla Sklep_internetowy.Server.Models.User:



Właściwości

- string? **FirstName** `[get, set]`

  Imię użytkownika.
- string? **LastName** `[get, set]`

  Nazwisko użytkownika.
- DateTime **CreatedAt** = DateTime.UtcNow `[get, set]`

  Data i godzina rejestracji konta w systemie (domyślnie czas UTC).
- ICollection< Order > **Orders** = new List<Order>() `[get, set]`

  Kolekcja wszystkich zamówień złożonych przez danego użytkownika (relacja jeden-do-wielu).

8.85.1 Opis szczegółowy

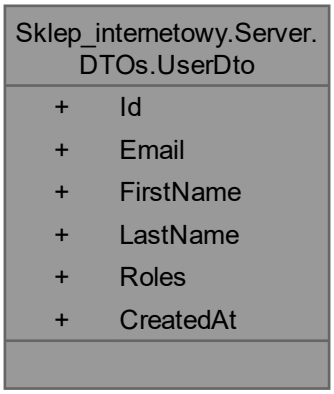
Rozszerzona klasa użytkownika systemu, dziedzicząca po IdentityUser. Przechowuje dodatkowe dane profilowe, takie jak imię i nazwisko, oraz zarządza relacją z historią zamówień klienta.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/User.cs

8.86 Dokumentacja klasy Sklep_internetowy.Server.DTOs.UserDto

Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o profilu użytkownika. Klasa agreguje dane identyfikacyjne, personalne oraz przypisane role systemowe, służąc do bezpiecznego przesyłania informacji o koncie z warstwy serwerowej do interfejsu klienta.

Diagram współpracy dla Sklep_internetowy.Server.DTOs.UserDto:



Właściwości

- string **Id** `[get, set]`

  Unikalny identyfikator użytkownika (GUID) w bazie danych systemu.
- string? **Email** `[get, set]`

  Adres e-mail przypisany do konta, wykorzystywany do autoryzacji i komunikacji.
- string? **FirstName** `[get, set]`

  Imię użytkownika zapisane w profilu personalnym.
- string? **LastName** `[get, set]`

  Nazwisko użytkownika zapisane w profilu personalnym.
- IList< string > **Roles** = new List<string>() `[get, set]`

  Kolekcja ról systemowych przypisanych do konta (np. "Admin", "User"), definiująca zakres uprawnień.
- DateTime **CreatedAt** `[get, set]`

  Data i godzina zarejestrowania konta w systemie (zapisana w formacie UTC).

8.86.1 Opis szczegółowy

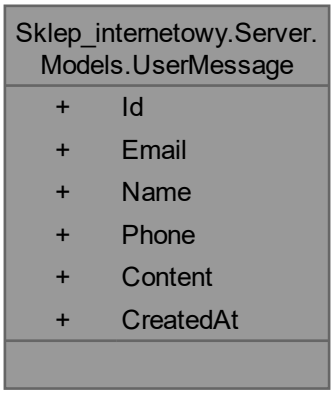
Obiekt transferu danych (DTO) reprezentujący szczegółowe informacje o profilu użytkownika. Klasa agreguje dane identyfikacyjne, personalne oraz przypisane role systemowe, służąc do bezpiecznego przesyłania informacji o koncie z warstwy serwerowej do interfejsu klienta.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/UserDto.cs

8.87 Dokumentacja klasy Sklep_internetowy.Server.Models.UserMessage

Klasa reprezentująca wiadomość przesłaną przez użytkownika poprzez formularz kontaktowy. Przechowuje dane teleadresowe nadawcy oraz treść zapytania, umożliwiając obsługę zgłoszeń klientów.

Diagram współpracy dla Sklep_internetowy.Server.Models.UserMessage:



Właściwości

- int **Id** `[get, set]`

  Unikalny identyfikator wiadomości w bazie danych.
- string **Email** = string.Empty `[get, set]`

  Adres e-mail nadawcy wiadomości (wymagany, walidacja formatu e-mail).
- string **Name** = string.Empty `[get, set]`

  Imię i nazwisko lub nazwa wyświetlana nadawcy (maksymalnie 100 znaków).
- string? **Phone** `[get, set]`

  Opcjonalny numer telefonu kontaktowego nadawcy.
- string **Content** = string.Empty `[get, set]`

  Treść wiadomości lub zapytania przesłanego przez użytkownika (maksymalnie 1000 znaków).
- DateTime **CreatedAt** = DateTime.UtcNow `[get, set]`

  Data i godzina zarejestrowania wiadomości w systemie (domyślnie czas UTC).

8.87.1 Opis szczegółowy

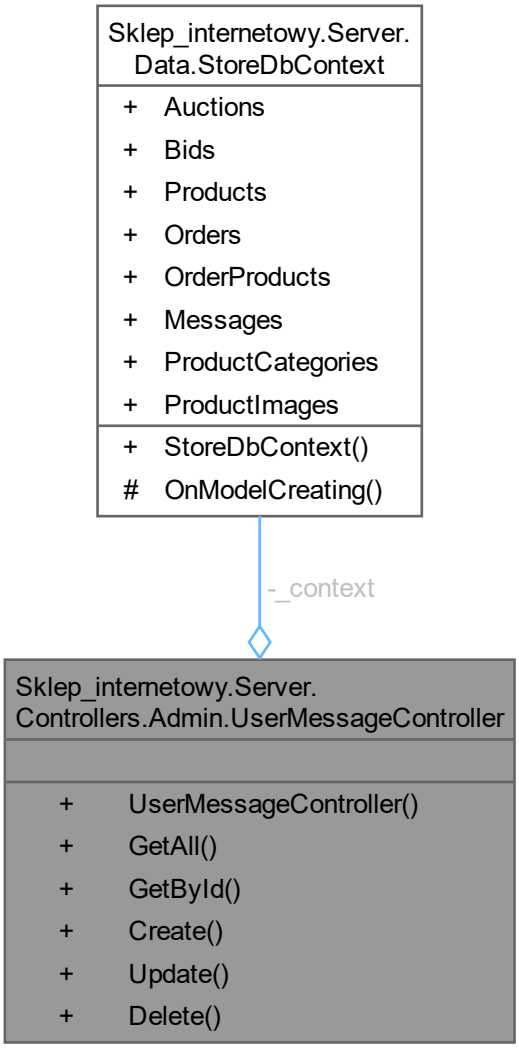
Klasa reprezentująca wiadomość przesłaną przez użytkownika poprzez formularz kontaktowy. Przechowuje dane teleadresowe nadawcy oraz treść zapytania, umożliwiając obsługę zgłoszeń klientów.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Models/UserMessage.cs

8.88 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.↩ UserMessageController

Kontroler API odpowiedzialny za zarzadzanie wiadomosciami kontaktowymi uzytkownikow. Umozliwia przegladanie, wyszukiwanie, tworzenie oraz edycje zgloszen w panelu administracyjnym.

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.UserMessageController:



Metody publiczne

- UserMessageController (StoreDbContext context)

  Inicjalizuje nowa instancje klasy UserMessageController.
- async Task< ActionResult< IEnumerable< UserMessage > > > GetAll ()

  Pobiera pelna liste wiadomosci zapisanych w systemie. Wiadomosci sa sortowane malejaco wedlug daty ich utworzenia.
- async Task< ActionResult< UserMessage > > GetById (int id)

  Pobiera szczegolowe dane konkretnej wiadomosci na podstawie jej identyfikatora.
- async Task< ActionResult< UserMessage > > Create (UserMessage message)

  Tworzy i zapisuje nowa wiadomosc w bazie danych. Automatycznie przypisuje aktualna date UTC do pola CreatedAt.
- async Task< IActionResult > Update (int id, UserMessage updated)

  Aktualizuje dane istniejacej wiadomosci kontaktowej.
- async Task< IActionResult > Delete (int id)

Trwale usuwa wiadomosc z systemu.

8.88.1 Opis szczegółowy

Kontroler API odpowiedzialny za zarzadzanie wiadomosciami kontaktowymi uzytkownikow. Umozliwia przegladanie, wyszukiwanie, tworzenie oraz edycje zgloszen w panelu administracyjnym.

8.88.2 Dokumentacja konstruktora i destruktora

8.88.2.1 UserMessageController()

```
Sklep_internetowy.Server.Controllers.Admin.UserMessageController.UserMessageController (
            StoreDbContext context)  [inline]
```

Inicjalizuje nowa instancje klasy UserMessageController.

Parametry

context	Kontekst bazy danych StoreDbContext.

8.88.3 Dokumentacja funkcji składowych

8.88.3.1 Create()

```
async Task< ActionResult< UserMessage > > Sklep_internetowy.Server.Controllers.Admin.User↩
MessageController.Create (
            UserMessage message)  [inline]
```

Tworzy i zapisuje nowa wiadomosc w bazie danych. Automatycznie przypisuje aktualna date UTC do pola Created↩ At.

Parametry

message	Obiekt wiadomosci przeslany w korpusie zadania.

Zwraca

Obiekt nowo utworzonej wiadomosci wraz z linkiem do jej pobrania.

<response code="201">Wiadomosc zostala pomyslnie utworzona.</response> <response code="400">Gdy model danych jest nieprawidlowy.</response>

8.88.3.2 Delete()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UserMessageController.↩
Delete (
            int id)  [inline]
```

Trwale usuwa wiadomosc z systemu.

Parametry

id	ID wiadomosci przeznaczonej do usuniecia.

Zwraca

Brak zawartosci (204) po pomyslnym usunieciu.

<response code="204">Wiadomosc zostala usunieta.</response> <response code="404">Gdy wiadomosc nie istnieje.</response>

8.88.3.3 GetAll()

```
async Task< ActionResult< IEnumerable< UserMessage > > > Sklep_internetowy.Server.Controllers.↩
Admin.UserMessageController.GetAll ()  [inline]
```

Pobiera pelna liste wiadomosci zapisanych w systemie. Wiadomosci sa sortowane malejaco wedlug daty ich utworzenia.

Zwraca

Kolekcja obiektow UserMessage.

<response code="200">Zwraca liste wiadomosci.</response>

8.88.3.4 GetById()

```
async Task< ActionResult< UserMessage > > Sklep_internetowy.Server.Controllers.Admin.User↩
MessageController.GetById (
            int id)  [inline]
```

Pobiera szczegolowe dane konkretnej wiadomosci na podstawie jej identyfikatora.

Parametry

id	Unikalny identyfikator wiadomosci (ID).

Zwraca

Obiekt UserMessage lub blad 404, jesli wiadomosc nie istnieje.

<response code="200">Zwraca znaleziona wiadomosc.</response> <response code="404">Gdy wiadomosc o podanym ID nie zostala znaleziona.</response>

8.88.3.5 Update()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UserMessageController.↩
Update (
            int id,
            UserMessage updated)  [inline]
```

Aktualizuje dane istniejacej wiadomosci kontaktowej.

Parametry

id	ID wiadomosci do aktualizacji.
updated	Zaktualizowany obiekt wiadomosci.

Zwraca

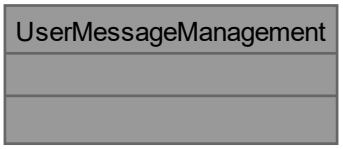
Brak zawartosci (204) w przypadku sukcesu lub blad walidacji.

<response code="204">Dane zostaly pomyslnie zaktualizowane.</response> <response code="400">Gdy ID w sciezce nie zgadza sie z ID w obiekcie.</response> <response code="404">Gdy wiadomosc o podanym ID nie istnieje.</response>
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/UserMessageController.cs

8.89 Dokumentacja klasy UserMessageManagement

Diagram współpracy dla UserMessageManagement:



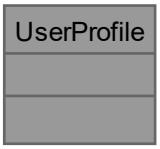
8.89.1 Opis szczegółowy

@description Renderuje interfejs administratora do obslugi wiadomosci. Zawiera logike pobierania danych z API oraz zarzadzania oknami modalnymi dla operacji CRUD.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/AdminDashboard/messages/UserMessageManagement.jsx

8.90 Dokumentacja klasy UserProfile

Diagram współpracy dla UserProfile:



8.90.1 Opis szczegółowy

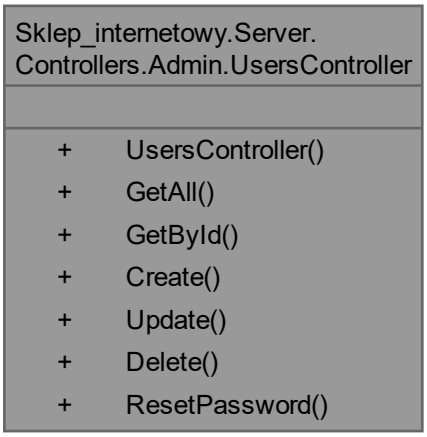
@description Glowny komponent zarzadzajacy danymi zalogowanego uzytkownika. Obsluguje pobieranie profilu, aktualizacje danych oraz ladowanie historii zakupow z API.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/UserProfile/UserProfile.jsx

8.91 Dokumentacja klasy Sklep_internetowy.Server.Controllers.Admin.UsersController

Kontroler administracyjny odpowiedzialny za zarzadzanie kontami uzytkownikow i ich uprawnieniami (rolami). Umozliwia pelny cykl zycia uzytkownika, w tym tworzenie, edycje, usuwanie oraz resetowanie hasel.

Diagram współpracy dla Sklep_internetowy.Server.Controllers.Admin.UsersController:



Metody publiczne

- UsersController (UserManager< User > userManager, RoleManager< IdentityRole > roleManager)

  Inicjalizuje nowa instancje klasy UsersController.
- async Task< IActionResult > GetAll ()

  Pobiera liste wszystkich uzytkownikow zarejestrowanych w systemie wraz z ich rolami.
- async Task< IActionResult > GetById (string id)

  Pobiera szczegolowe dane konkretnego uzytkownika na podstawie identyfikatora ID.
- async Task< IActionResult > Create ([FromBody] RegisterUserRequest dto)

  Tworzy nowe konto uzytkownika i przypisuje mu zdefiniowane role systemowe.
- async Task< IActionResult > Update (string id, [FromBody] UserUpdateDto dto)

  Aktualizuje dane istniejacego uzytkownika, w tym adres email oraz zestaw rol. Proces rol polega na usunieciu dotychczasowych i nadaniu nowych z zadania.
- async Task< IActionResult > Delete (string id)

  Trwale usuwa konto uzytkownika z systemu. Zawiera blokade uniemozliwiajaca administratorowi usuniecie samego siebie.
- async Task< IActionResult > ResetPassword (string id)

  Resetuje haslo wybranego uzytkownika, generujac nowe, losowe haslo tymczasowe.

8.91.1 Opis szczegółowy

Kontroler administracyjny odpowiedzialny za zarzadzanie kontami uzytkownikow i ich uprawnieniami (rolami). Umozliwia pelny cykl zycia uzytkownika, w tym tworzenie, edycje, usuwanie oraz resetowanie hasel.

8.91.2 Dokumentacja konstruktora i destruktora

8.91.2.1 UsersController()

```
Sklep_internetowy.Server.Controllers.Admin.UsersController.UsersController (
            UserManager< User > userManager,
            RoleManager< IdentityRole > roleManager)  [inline]
```

Inicjalizuje nowa instancje klasy UsersController.

Parametry

userManager	Menedzer uzytkownikow systemu Identity.
roleManager	Menedzer rol systemu Identity.

8.91.3 Dokumentacja funkcji składowych

8.91.3.1 Create()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.Create
(
            [FromBody] RegisterUserRequest dto)  [inline]
```

Tworzy nowe konto uzytkownika i przypisuje mu zdefiniowane role systemowe.

Parametry

dto	Dane rejestracyjne uzytkownika (email, haslo, role).

Zwraca

Dane nowo utworzonego uzytkownika.

<response code="200">Uzytkownik zostal utworzony.</response> <response code="400">Gdy walidacja danych lub proces tworzenia konta zakonczyl sie bledem.</response>

8.91.3.2 Delete()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.Delete
(
            string id)  [inline]
```

Trwale usuwa konto uzytkownika z systemu. Zawiera blokade uniemozliwiajaca administratorowi usuniecie samego siebie.

Parametry

id	ID uzytkownika przeznaczonego do usuniecia.

Zwraca

Komunikat o statusie operacji.

<response code="200">Uzytkownik usuniety.</response> <response code="400">Gdy probowano usunac wlasne konto.</response> Pobranie identyfikatora aktualnie zalogowanego uzytkownika wykonujacego akcje.

8.91.3.3 GetAll()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.GetAll
()  [inline]
```

Pobiera liste wszystkich uzytkownikow zarejestrowanych w systemie wraz z ich rolami.

Zwraca

Kolekcja obiektow UserDto zawierajaca podstawowe dane profilowe i przypisane role.

<response code="200">Zwraca liste uzytkownikow.</response>

8.91.3.4 GetById()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.GetById
(
            string id)  [inline]
```

Pobiera szczegolowe dane konkretnego uzytkownika na podstawie identyfikatora ID.

Parametry

id	Unikalny identyfikator uzytkownika (GUID).

Zwraca

Obiekt UserDto lub blad 404, jesli uzytkownik nie istnieje.

8.91.3.5 ResetPassword()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.Reset↩
Password (
            string id)  [inline]
```

Resetuje haslo wybranego uzytkownika, generujac nowe, losowe haslo tymczasowe.

Parametry

id	ID uzytkownika, ktorego haslo ma zostac zresetowane.

Zwraca

Obiekt zawierajacy nowe haslo tymczasowe.

8.91.3.6 Update()

```
async Task< IActionResult > Sklep_internetowy.Server.Controllers.Admin.UsersController.Update
(
            string id,
            [FromBody] UserUpdateDto dto)  [inline]
```

Aktualizuje dane istniejacego uzytkownika, w tym adres email oraz zestaw rol. Proces rol polega na usunieciu dotychczasowych i nadaniu nowych z zadania.

Parametry

id	ID uzytkownika do edycji.
dto	Obiekt zawierajacy zaktualizowany email oraz liste rol.

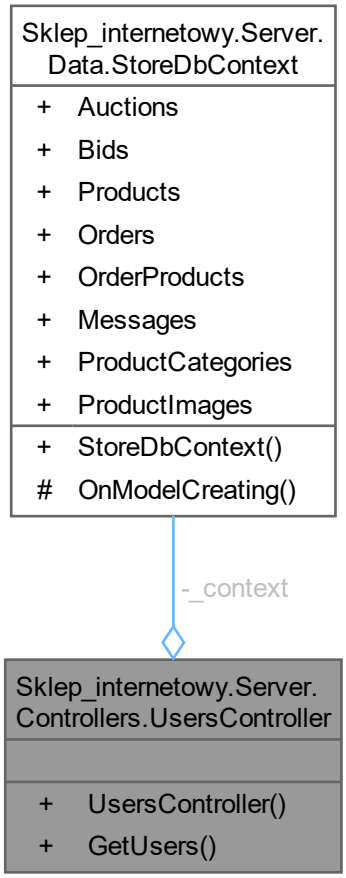
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Admin/UsersController.cs

8.92 Dokumentacja klasy Sklep_internetowy.Server.Controllers.UsersController

Kontroler API odpowiedzialny za udostepnianie informacji o uzytkownikach systemu. Umozliwia bezpieczne pobieranie danych profilowych przeksztalconych do formatu obiektow transferu danych (DTO).

Diagram współpracy dla Sklep_internetowy.Server.Controllers.UsersController:



Metody publiczne

- UsersController (StoreDbContext context)

  Inicjalizuje nowa instancje klasy UsersController.
- async Task< ActionResult< IEnumerable< UserDto > > > GetUsers ()

  Pobiera pelna liste uzytkownikow zarejestrowanych w systemie. Metoda mapuje encje bazodanowe na bezpieczne obiekty UserDto, aby uniknac przesylania wrazliwych danych (np. hasel) do klienta.

8.92.1 Opis szczegółowy

Kontroler API odpowiedzialny za udostepnianie informacji o uzytkownikach systemu. Umozliwia bezpieczne pobieranie danych profilowych przeksztalconych do formatu obiektow transferu danych (DTO).

8.92.2 Dokumentacja konstruktora i destruktora

8.92.2.1 UsersController()

```
Sklep_internetowy.Server.Controllers.UsersController.UsersController (
            StoreDbContext context)  [inline]
```

Inicjalizuje nowa instancje klasy UsersController.

Parametry

context	Kontekst bazy danych StoreDbContext wykorzystywany do dostepu do danych uzytkownikow.

8.92.3 Dokumentacja funkcji składowych

8.92.3.1 GetUsers()

```
async Task< ActionResult< IEnumerable< UserDto > > > Sklep_internetowy.Server.Controllers.↩
UsersController.GetUsers () [inline]
```

Pobiera pelna liste uzytkownikow zarejestrowanych w systemie. Metoda mapuje encje bazodanowe na bezpieczne obiekty UserDto, aby uniknac przesylania wrazliwych danych (np. hasel) do klienta.

Zwraca

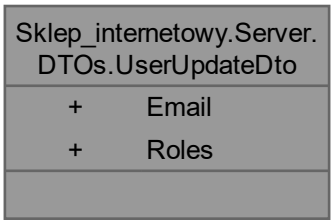
Kolekcja obiektow UserDto zawierajaca podstawowe informacje o uzytkownikach.

<response code="200">Zwraca liste uzytkownikow.</response>
Zapytanie wykorzystuje metode AsNoTracking() w celu zwiekszenia wydajnosci operacji tylko do odczytu.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/Controllers/Shop/UsersController.cs

8.93 Dokumentacja klasy Sklep_internetowy.Server.DTOs.UserUpdateDto

Obiekt transferu danych (DTO) wykorzystywany przez administratora do aktualizacji kluczowych parametrów konta użytkownika. Klasa umożliwia synchronizację zmian w adresie e-mail oraz modyfikację zestawu ról przypisanych do tożsamości użytkownika.
Diagram współpracy dla Sklep_internetowy.Server.DTOs.UserUpdateDto:



Właściwości

- string **Email** = null! `[get, set]`

  Zaktualizowany adres e-mail użytkownika, służący do autoryzacji i komunikacji systemowej.
- IList< string > **Roles** = new List<string>() `[get, set]`

  Kolekcja ról systemowych (np. "Admin", "User"), które mają zostać nadane użytkownikowi w celu definicji poziomu dostępu.

8.93.1 Opis szczegółowy

Obiekt transferu danych (DTO) wykorzystywany przez administratora do aktualizacji kluczowych parametrów konta użytkownika. Klasa umożliwia synchronizację zmian w adresie e-mail oraz modyfikację zestawu ról przypisanych do tożsamości użytkownika.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Sklep_internetowy.Server/DTOs/UserUpdateDto.cs

8.94 Dokumentacja klasy WishlistPage

Diagram współpracy dla WishlistPage:

8.94.1 Opis szczegółowy

@description Renderuje widok kolekcji ulubionych produktow. Zarzadza wyswietlaniem stanu pustego oraz dynamicznej siatki kart produktow pobranych z WishlistContext.
Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/pages/WishList/WishlistPage.jsx

8.95 Dokumentacja klasy WishlistProvider

Komponent dostawcy, ktory opakowuje aplikacje i zarzadza stanem listy zyczen.
Diagram współpracy dla WishlistProvider:

8.95.1 Opis szczegółowy

Komponent dostawcy, ktory opakowuje aplikacje i zarzadza stanem listy zyczen.

Parametry

{Object}	props - Wlasciwosci komponentu, w tym dzieci (children).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- sklep_internetowy.client/src/context/WishListContext.jsx

Rozdział 9

Dokumentacja plików

9.1 Dokumentacja pliku sklep_internetowy.client/src/api/auctionApi.js

Serwis kliencki do obslugi zapytan API zwiazanych z systemem aukcyjnym.

Zmienne

- const getHeaders
- export const getActiveAuctions
- export const getAuction
- export const placeBid
- export const createAuction

9.1.1 Opis szczegółowy

Serwis kliencki do obslugi zapytan API zwiazanych z systemem aukcyjnym.
Zawiera zestaw funkcji asynchronicznych do komunikacji z kontrolerem BidController, obslugujac pobieranie danych aukcji, licytowanie oraz tworzenie nowych licytacji.

9.1.2 Dokumentacja zmiennych

9.1.2.1 createAuction

```
export const createAuction
```

Wartość początkowa:

```
                        = async (data) => {
    const response = await fetch("/api/bid/create", {
        method: "POST",
        headers: getHeaders({ "Content-Type": "application/json" }),
        body: JSON.stringify({
            productId: data.productId,
            startingPrice: data.startingPrice
        })
    });
    if (!response.ok) throw new Error("Failed to create auction");
    try {
        return await response.json();
    } catch {
        return null;
    }
}
```

Nota

Funkcja asynchroniczna @description Tworzy nowa aukcje dla wskazanego produktu.

Parametry

{Object}	data - Obiekt z danymi aukcji.

{number}	data.productId - ID produktu do wystawienia.
{number}	data.startingPrice - Cena wywolawcza.

Zwraca

{Promise<Object|null>} Dane utworzonej aukcji lub null w przypadku bledu parsowania.

Wyjątki

{Error}	Gdy wystapi blad podczas tworzenia licytacji.

9.1.2.2 getActiveAuctions

```
export const getActiveAuctions
```

Wartość początkowa:

```
                              = async () => {
    const response = await fetch("/api/bid/active", {
        method: "GET",
        headers: getHeaders()
    });
    if (!response.ok) throw new Error("Failed to fetch active auctions");
    try {
        return await response.json();
    } catch {
        return [];
    }
}
```

Nota

Funkcja asynchroniczna @description Pobiera liste wszystkich obecnie aktywnych (trwajacych) aukcji.

Zwraca

{Promise<Array>} Obietnica zwracajaca tablice obiektow AuctionDto.

Wyjątki

{Error}	Gdy serwer zwroci blad podczas pobierania.

9.1.2.3 getAuction

```
export const getAuction
```

Wartość początkowa:

```
                          = async (id) => {
    const response = await fetch(`/api/bid/${id}`, {
        method: "GET",
        headers: getHeaders()
    });
    if (!response.ok) throw new Error(`Failed to fetch auction with id ${id}`);
    try {
        const data = await response.json();
        return { data };
    } catch {
        return { data: null };
    }
}
```

Nota

Funkcja asynchroniczna @description Pobiera szczegolowe dane konkretnej aukcji na podstawie jej identyfikatora.

Parametry

{number|string}	id - Unikalny identyfikator aukcji.

Zwraca

{Promise<Object>} Obiekt zawierajacy dane aukcji w polu 'data'.

Wyjątki

{Error}	Gdy aukcja nie zostanie znaleziona lub wystapi blad sieci.

9.1.2.4 getHeaders

```
const getHeaders
```

Wartość początkowa:

```
                = (customHeaders = {}) => {
    const token = localStorage.getItem("token");
    const headers = { ...customHeaders };
    if (token) {
        headers["Authorization"] = `Bearer ${token}`;
    }
    return headers;
}
```

@description Funkcja pomocnicza generujaca naglowki zapytan HTTP, w tym token autoryzacyjny Bearer.

Parametry

{Object}	customHeaders - Dodatkowe specyficzne naglowki (np. Content-Type).

Zwraca

{Object} Zestaw naglowkow z dolaczonym tokenem JWT z localStorage.

9.1.2.5 placeBid

```
export const placeBid
```

Wartość początkowa:

```
                  = async (id, amount) => {
    const response = await fetch(`/api/bid/${id}/bid`, {
        method: "POST",
        headers: getHeaders({ "Content-Type": "application/json" }),
        body: JSON.stringify({ amount })
    });
    if (!response.ok) throw new Error(`Failed to place bid on auction ${id}`);
    return response.json();
}
```

Nota

Funkcja asynchroniczna @description Przesyla nowa oferte licytacji dla wybranej aukcji.

Parametry

{number|string}	id - ID aukcji, w ktorej skladana jest oferta.
{number}	amount - Kwota nowej oferty (podbicie).

Zwraca

{Promise<Object>} Wynik operacji zwrocony przez serwer.

Wyjątki

{Error}	Gdy oferta jest zbyt niska, aukcja wygasla lub wystapil blad autoryzacji.

9.2 Dokumentacja pliku sklep_internetowy.client/src/App.jsx

Glowny komponent konfiguracyjny aplikacji TechStore.

Zmienne

- import LoginPage from pages LoginPage LoginPage
- import Registration from pages Registration Registration
- import Footer from components footer Footer
- import UserProfile from pages UserProfile UserProfile
- import UserMessageManagement from pages AdminDashboard messages UserMessageManagement
- import ProductDetailsShop from pages Products Shop ProductDetailsShop
- import ComparePage from pages Products ComparePage
- import AuctionList from pages Auction AuctionList
- import AuctionDetails from pages Auction AuctionDetails
- import CreateAuction from pages Auction CreateAuction
- import WishlistPage from pages WishList WishlistPage
- import PaymentPage from pages Payment PaymentPage
- export default App

9.2.1 Opis szczegółowy

Glowny komponent konfiguracyjny aplikacji TechStore.
Modul ten definiuje strukture routingu, zarzadza globalnymi dostawcami kontekstu (Theme) oraz integruje kluczowe elementy interfejsu takie jak pasek nawigacji i stopka.

9.2.2 Dokumentacja zmiennych

9.2.2.1 App

```
function App
```

Inicjalizacja hooka zarzadzajacego stanem porownywarki produktow.

9.2.2.2 AuctionDetails

```
import AuctionDetails from pages Auction AuctionDetails
```

Pobranie identyfikatora aukcji z parametrow sciezki URL.
Status autoryzacji uzytkownika pobrany z dedykowanego hooka.
Stan przechowujacy szczegolowe informacje o aukcji (produkt, cena, czas).

Wartosc nowej oferty licytacyjnej wpisana w formularzu.
Flaga kontrolujaca blokowanie interfejsu podczas wysylania oferty.
Referencja do obiektu polaczenia SignalR zachowujaca stan miedzy renderami.

Nota

React useEffect: Inicjalizacja komponentu i konfiguracja SignalR.

Pobiera dane poczatkowe aukcji i ustanawia polaczenie z Hubem aukcyjnym. Rejestruje handlery dla zdarzen "↩ BidPlaced" oraz "AuctionFinished".
@callback BidPlaced @description Reaguje na powiadomienie o nowej najwyzszej ofercie od innego uzytkownika.
@callback AuctionFinished @description Wyswietla powiadomienie o zakonczeniu licytacji i odswieza dane.
@description Pobiera aktualny stan aukcji z serwera za pomoca API REST.

9.2.2.3 AuctionList

```
import AuctionList from pages Auction AuctionList
```

Stan przechowujacy tablice aktywnych aukcji pobranych z serwera.
Flaga logiczna okreslajaca, czy trwa proces pobierania danych.
Przechowuje informacie o bledzie w przypadku niepowodzenia komunikacji z API.

Nota

React useEffect: Inicjalizacja komponentu: asynchroniczne pobieranie listy aukcji.

Wywoluje funkcje getActiveAuctions i aktualizuje stany komponentu.
@description Funkcja nawigacyjna przekierowujaca administratora do formularza tworzenia aukcji.

9.2.2.4 ComparePage

```
import ComparePage from pages Products ComparePage
```

Stan przechowujacy szczegolowe dane produktow pobrane z serwera.
Flaga okreslajaca stan ladowania danych asynchronicznych.
Tresc komunikatu o bledzie.
Adres domyslnego obrazu w przypadku braku grafik produktu.

Nota

React useEffect: Pobieranie danych do porownania. @description Uruchamia sie przy kazdej zmianie listy compareItems. Wysyla zapytanie POST z tablica ID do punktu koncowego API.

@description Pomocnicza funkcja do renderowania bloku ceny.
Obsluguje wyswietlanie ceny promocyjnej (z przekresleniem) oraz standardowej.

Parametry

{Object}	product - Obiekt danych produktu.

Zwraca

{JSX.Element} Sformatowany widok ceny.

9.2.2.5 CreateAuction

```
import CreateAuction from pages Auction CreateAuction
```

Stan przechowujacy tablice produktow pobranych z serwera.
Wybrany identyfikator produktu, ktory ma zostac wystawiony.
Cena wywolawcza dla nowej aukcji.
Flaga blokujaca przycisk podczas trwania operacji asynchronicznej.

Nota

React useEffect: Inicjalizacja komponentu. @description Pobiera liste produktow z endpointu domowego w celu wypelnienia listy wyboru.

Nota

Funkcja asynchroniczna @description Obsluguje zdarzenie wyslania formularza, waliduje dane i wywoluje API aukcji.

Parametry

{Event}	e - Obiekt zdarzenia submit.

9.2.2.6 Footer

`function Footer`
Funkcja tlumaczen z biblioteki i18next.
Stan formularza kontaktowego.
Przechowuje dane wprowadzone przez uzytkownika do formularza.
Obsluga zmiany wartosci w polach formularza.

Parametry

{Event}	e Zdarzenie zmiany pola formularza.

Obsluga wyslania formularza kontaktowego.
Wysyla dane formularza do API przy uzyciu metody POST.

Parametry

{Event}	e Zdarzenie wyslania formularza.

9.2.2.7 LoginPage

`import LoginPage from pages LoginPage LoginPage`
Stan przechowujacy dane logowania (login i haslo).
Komunikat o sukcesie operacji.
Przechowuje komunikaty o bledach autoryzacji lub sieci.
@description Aktualizuje lokalny stan formularza przy kazdej zmianie w polach tekstowych.

Parametry

{Event}	e - Obiekt zdarzenia zmiany pola wejsciowego.

9.2.2.8 PaymentPage

`import PaymentPage from pages Payment PaymentPage`
Dane pobrane z kontekstu koszyka zakupowego.
Klucz sesji platnosci otrzymany z serwera.

Nota

React useEffect: Przekierowanie do koszyka, jesli jest on pusty.

Nota

- React useEffect: Pobieranie PaymentIntent. @description Tworzy zamiar platnosci na serwerze na podstawie wartosci koszyka.

Nota

Funkcja asynchroniczna @description Finalizuje proces zakupowy: zapisuje zamowienie w bazie danych, generuje plik PDF z faktura i czysci koszyk.

Parametry

{string}	paymentId - Identyfikator pomyslnej platnosci Stripe.

9.2.2.9 ProductDetailsShop

`import ProductDetailsShop from pages Products Shop ProductDetailsShop`
Pobranie identyfikatora produktu z parametrow sciezki URL.
Stan przechowujacy dane pobranego produktu.
Flaga kontrolujaca wyswietlanie stanu ladowania danych.
Hook zapewniajacy dostep do funkcji dodawania do koszyka.
Hook umozliwiajacy nawigacje wsteczna lub przekierowania.
Indeks aktualnie wyswietlanego zdjecia w galerii produktu.
Adres domyslnego obrazu uzywanego w przypadku braku grafik produktu.

Nota

React useEffect: Pobieranie szczegolow produktu. @description Inicjuje zapytanie API po zamontowaniu komponentu lub zmianie ID produktu.

Sprawdzenie czy produkt znajduje sie juz w porownywarce.
Sprawdzenie czy osiagnieto limit produktow w porownywarce (blokada przycisku).
Tablica zdjec produktu lub zestaw z obrazem domyslnym.
@description Przelacza wyswietlane zdjecie na nastepne w kolejnosci.

9.2.2.10 Registration

`import Registration from pages Registration Registration`
Stan przechowujacy dane wejsciowe formularza (login, email, hasla).
Tresc komunikatu o sukcesie operacji.
Tresc komunikatu o bledzie (walidacja lub odpowiedz serwera).
@description Handler aktualizujacy stan formularza na podstawie zmian w polach input.

Parametry

{Event}	e - Obiekt zdarzenia zmiany.

9.2.2.11 UserMessageManagement

`import UserMessageManagement from pages AdminDashboard messages UserMessageManagement`
Stan przechowujacy liste wiadomosci pobranych z bazy danych.
Flaga okreslajaca stan ladowania danych.
Przechowuje komunikaty o bledach operacji asynchronicznych.
Status widocznosci modala edycji.
Status widocznosci modala usuwania.
Status widocznosci modala tworzenia nowej wiadomosci.
Obiekt aktualnie edytowanej lub usuwanej wiadomosci.
Dane poczatkowe dla formularza nowej wiadomosci.

Nota

Funkcja asynchroniczna @description Pobiera wszystkie wiadomosci z punktu koncowego API.

9.2.2.12 UserProfile

`import UserProfile from pages UserProfile UserProfile`
Stan przechowujacy podstawowe dane profilowe uzytkownika.
Obiekt stanu do obslugi komunikatow o statusie aktualizacji profilu.
Lista zamowien przypisanych do zalogowanego uzytkownika.
Flaga kontrolujaca wyswietlanie spinnera ladowania dla sekcji zamowien.

Nota

React useEffect: Inicjalizacja danych profilu i zamowien. @description Sprawdza obecnosc tokenu JWT. Jesli uzytkownik jest zalogowany, pobiera dane profilu z /api/Account/me oraz historie zamowien.

Nota

Funkcja asynchroniczna @description Pobiera aktualne dane uzytkownika z API Account.

9.2.2.13 WishlistPage

`function WishlistPage`
Pobranie danych i funkcji sterujacych z kontekstu listy zyczen.
Funkcja dodawania produktu do koszyka pobrana z CartContext.
Adres URL domyslnego obrazu uzywanego w przypadku braku zdjec produktu.

-

9.2.3 EmptyState

Obsluga wyswietlania komunikatu w przypadku braku produktow na liscie.
Logika wyboru glownego zdjecia produktu.

9.3 Dokumentacja pliku sklep_internetowy.client/src/auth/useAuth.js

Hook Reactowy do zarzadzania stanem uwierzytelniania uzytkownika.

Zmienne

- export const useAuth

9.3.1 Opis szczegółowy

Hook Reactowy do zarzadzania stanem uwierzytelniania uzytkownika.

9.3.2 Dokumentacja zmiennych

9.3.2.1 useAuth

`export const useAuth`
Wartość początkowa:

```
                    = () => {
    const token = localStorage.getItem("token");
    const role = localStorage.getItem("role");

    return {
        isAuthenticated: !!token,
        isAdmin: role === "Admin"
    };
}
```

@description Funkcja pomocnicza (helper) do sprawdzania stanu autoryzacji uzytkownika.
Pobiera token oraz role z magazynu lokalnego (localStorage) w celu weryfikacji uprawnien.

Zwraca

{Object} Obiekt zawierajacy flagi logiczne: isAuthenticated (czy zalogowany) oraz isAdmin (czy posiada uprawnienia administratora).

9.4 Dokumentacja pliku sklep_internetowy.client/src/components/↩ admin/product/ProductCard.jsx

Komponent karty produktu dedykowany dla panelu administracyjnego.

Zmienne

- export default ProductCard

9.4.1 Opis szczegółowy

Komponent karty produktu dedykowany dla panelu administracyjnego.

9.4.2 Dokumentacja zmiennych

9.4.2.1 ProductCard

```
export default ProductCard
```

- Flaga okreslajaca, czy produkt posiada obecnie aktywna znizke.

@description Wyswietla systemowe okno potwierdzenia. Jesli uzytkownik zatwierdzi, wywoluje funkcje onDelete przekazana w parametrach.

9.5 Dokumentacja pliku sklep_internetowy.client/src/components/↩ shop/product/ProductCard.jsx

Komponent karty produktu wyswietlanej w siatce produktow.

Funkcje

- function ProductCard ({ product, addToCart, onClick })

9.5.1 Opis szczegółowy

Komponent karty produktu wyswietlanej w siatce produktow.
Odpowiada za prezentacje pojedynczego produktu, wyswietlanie ceny, obsluge znizek, dodawanie do koszyka oraz zarzadzanie lista zyczen.

9.5.2 Dokumentacja funkcji

9.5.2.1 ProductCard()

```
function ProductCard (
              { product, addToCart, onClick })
```

Funkcja tlumaczen interfejsu uzytkownika.
Hook do nawigacji pomiedzy stronami.
Sprawdza czy produkt ma aktywna znizke.

Parametry

{Object}	p Obiekt produktu.

Zwraca

{boolean} True jesli znizka jest aktywna.

Domyslny obrazek produktu uzywany gdy brak zdjec.
Funkcje oraz dane z kontekstu listy zyczen.
Flaga informujaca czy produkt znajduje sie w liscie zyczen.
Glowny obrazek produktu.

9.6 Dokumentacja pliku sklep_internetowy.client/src/components/admin/product/ProductForm.jsx

Komponent formularza przeznaczony do tworzenia i edycji produktow w panelu administracyjnym.

9.6.1 Opis szczegółowy

Komponent formularza przeznaczony do tworzenia i edycji produktow w panelu administracyjnym.
Obsluguje wprowadzanie danych tekstowych, numerycznych, wybor kategorii z API oraz przesylanie plikow graficznych.

9.7 Dokumentacja pliku sklep_internetowy.client/src/components/carousel/Carousel.jsx

Komponent karuzeli obrazow wykorzystujacy biblioteke Bootstrap.

9.7.1 Opis szczegółowy

Komponent karuzeli obrazow wykorzystujacy biblioteke Bootstrap.
Komponent renderuje pelnowymiarowy baner z funkcja automatycznego przewijania, zoptymalizowany pod katem stalej wysokosci i responsywnego wypelnienia obrazem.

9.8 Dokumentacja pliku sklep_internetowy.client/src/components/featuredProductCard/FeaturedProductCard.jsx

Komponent karty produktu wyroznionego (Featured Product).

Zmienne

- export default FeaturedProductCard

9.8.1 Opis szczegółowy

Komponent karty produktu wyroznionego (Featured Product).
Prezentuje kluczowe informacje o produkcie w powiekszonym formacie, z obsluga dynamicznych cen, etykiet promocyjnych oraz nawigacji.

9.8.2 Dokumentacja zmiennych

9.8.2.1 FeaturedProductCard

```
export default FeaturedProductCard
```

@description Funkcja pomocnicza sprawdzajaca, czy produkt posiada obecnie aktywny rabat.

Parametry

{Object}	p - Obiekt produktu.

Zwraca

{Boolean} Prawda, jesli promocja jest aktywna lub procent zni?ki jest wi?kszy od zera.

9.9 Dokumentacja pliku sklep_internetowy.client/src/components/footer/Footer.jsx

Komponent stopki strony zawierajacy formularz kontaktowy oraz mape Google.

Zmienne

- export default Footer

9.9.1 Opis szczegółowy

Komponent stopki strony zawierajacy formularz kontaktowy oraz mape Google.
Komponent umozliwia uzytkownikowi wyslanie wiadomosci przez formularz kontaktowy oraz wyswietla informacje o lokalizacji firmy za pomoca osadzonej mapy Google.

9.9.2 Dokumentacja zmiennych

9.9.2.1 Footer

```
export default Footer
```

Funkcja tlumaczen z biblioteki i18next.
Stan formularza kontaktowego.
Przechowuje dane wprowadzone przez uzytkownika do formularza.
Obsluga zmiany wartosci w polach formularza.

Parametry

{Event}	e Zdarzenie zmiany pola formularza.

Obsluga wyslania formularza kontaktowego.
Wysyla dane formularza do API przy uzyciu metody POST.

Parametry

{Event}	e Zdarzenie wyslania formularza.

9.10 Dokumentacja pliku sklep_internetowy.client/src/components/navbar/Navbar.jsx

Komponent paska nawigacji aplikacji.

Funkcje

- function Navbar ({ compareCount })

9.10.1 Opis szczegółowy

Komponent paska nawigacji aplikacji.

Odpowiada za wyswietlanie glownego menu, wyszukiwarki z podpowiedziami, przelaczanie jezyka, motywu, rozmiaru czcionki oraz za obsluge stanu logowania uzytkownika.

9.10.2 Dokumentacja funkcji

9.10.2.1 Navbar()

```
function Navbar (
              { compareCount })
```

Hook tlumaczen oraz obiektu konfiguracji jezyka.
Aktualna wartosc zapytania wyszukiwania.
Podpowiedzi wyszukiwania dla kategorii i produktow.
Flaga okreslajaca widocznosc listy podpowiedzi.
Domyslny obrazek produktu uzywany gdy brak zdjecia.
Flaga informujaca czy uzytkownik jest zalogowany.
Flaga informujaca czy uzytkownik posiada uprawnienia administratora.
Kontekst motywu oraz rozmiaru czcionki.
Hook do nawigacji pomiedzy stronami.
Aktualna lokalizacja routingu.
Referencja do kontenera listy podpowiedzi wyszukiwania.
Lista produktow w liscie zyczen.
Lacznba liczba produktow w koszyku.
Zmienia aktualny jezyk aplikacji.

Parametry

{string}	lng Kod jezyka.

Wylogowuje aktualnego uzytkownika.
Usuwa dane sesji z localStorage i przekierowuje na strone logowania.
Efekt odpowiedzialny za zamykanie listy podpowiedzi po kliknieciu poza nia.
Efekt pobierajacy podpowiedzi wyszukiwania z API.
Wysylany jest request po krotkim opoznieniu (debounce).
Efekt reagujacy na zmiane trasy.
Resetuje pole wyszukiwania oraz sprawdza stan logowania uzytkownika.
Obsluguje wyslanie formularza wyszukiwania.

Parametry

{Event}	e Zdarzenie formularza.

Aktualna liczba produktow w porownywarce.
Lista glownego menu kategorii.

9.11 Dokumentacja pliku sklep_internetowy.client/src/components/↩ productGrid/ProductGrid.jsx

Komponent wyswietlajacy siatke produktow na stronie glownej.

Zmienne

- export default ProductGrid

9.11.1 Opis szczegółowy

Komponent wyswietlajacy siatke produktow na stronie glownej.

Pobiera liste produktow z API, losuje jeden produkt promocyjny oraz wyswietla pozostale produkty z podzialem na strony.

9.11.2 Dokumentacja zmiennych

9.11.2.1 ProductGrid

`function ProductGrid`
Lista wszystkich produktow pobranych z API.
Aktualnie wybrany produkt promocyjny.
Numer aktualnej strony paginacji.
Liczba produktow wyswietlanych na jednej stronie.
Funkcja dodajaca produkt do koszyka.
Hook do nawigacji pomiedzy stronami.
Efekt pobierajacy produkty z API przy pierwszym renderze komponentu.
Po pobraniu danych losowany jest jeden produkt z aktywna znizka.
Lista produktow bez produktu promocyjnego.
Zapobiega wyswietlaniu produktu promocyjnego dwa razy.
Indeks ostatniego produktu na aktualnej stronie.
Indeks pierwszego produktu na aktualnej stronie.
Lista produktow wyswietlanych na aktualnej stronie.
Calkowita liczba stron paginacji.
Zmienia aktualna strone paginacji.

Parametry

{number}	pageNumber Numer strony do wyswietlenia.

Obsluguje klikniecie w produkt.

Parametry

{number|string}	id Identyfikator produktu.

9.12 Dokumentacja pliku sklep_internetowy.client/src/components/ProtectedRoute.jsx

Komponent chroniacy trasy dostepne tylko dla administratora.

9.12.1 Opis szczegółowy

Komponent chroniacy trasy dostepne tylko dla administratora.
Sprawdza czy aktualny uzytkownik posiada uprawnienia administratora. Jesli nie, przekierowuje go na strone glowna.

9.13 Dokumentacja pliku sklep_internetowy.client/src/context/CartContext.jsx

Kontekst globalny koszyka zakupowego.

Zmienne

- const **CartContext** = createContext()

  Kontekst React przechowujacy dane koszyka.
- export const useCart = () => useContext(CartContext)

  Hook do latwego dostepu do kontekstu koszyka.

9.13.1 Opis szczegółowy

Kontekst globalny koszyka zakupowego.
Odpowiada za przechowywanie produktow w koszyku, zapisywanie ich do localStorage oraz udostepnianie funkcji zarzadzania koszykiem.

9.13.2 Dokumentacja zmiennych

9.13.2.1 useCart

```
export const useCart = () => useContext(CartContext)
```
Hook do latwego dostepu do kontekstu koszyka.

Zwraca

{Object} Aktualny kontekst koszyka.

9.14 Dokumentacja pliku sklep_internetowy.client/src/context/ThemeContext.jsx

Kontekst globalny do zarzadzania motywem oraz rozmiarem czcionki aplikacji.

Zmienne

- const **ThemeContext** = createContext()

  Kontekst React przechowujacy ustawienia wygladu aplikacji.
- export const useTheme = () => useContext(ThemeContext)

  Hook do pobierania kontekstu motywu aplikacji.

9.14.1 Opis szczegółowy

Kontekst globalny do zarzadzania motywem oraz rozmiarem czcionki aplikacji.
Pozwala przelaczac miedzy trybem jasnym i ciemnym oraz zmieniac rozmiar czcionki. Ustawienia sa zapisywane w localStorage.

9.14.2 Dokumentacja zmiennych

9.14.2.1 useTheme

```
export const useTheme = () => useContext(ThemeContext)
```
Hook do pobierania kontekstu motywu aplikacji.

Zwraca

{Object} Aktualny kontekst motywu.

9.15 Dokumentacja pliku sklep_internetowy.client/src/context/WishListContext.jsx

Kontekst React do zarzadzania lista zyczen (ulubionymi produktami).

Zmienne

- const **WishlistContext** = createContext()

  Obiekt kontekstu dla listy zyczen.
- export const useWishlist = () => useContext(WishlistContext)

  Hook zapewniajacy dostep do danych i metod listy zyczen.

9.15.1 Opis szczegółowy

Kontekst React do zarzadzania lista zyczen (ulubionymi produktami).
Umozliwia dodawanie i usuwanie produktow z listy oraz ich trwale przechowywanie w pamieci lokalnej przegladarki (localStorage).

9.15.2 Dokumentacja zmiennych

9.15.2.1 useWishlist

```
export const useWishlist = () => useContext(WishlistContext)
```
Hook zapewniajacy dostep do danych i metod listy zyczen.

Zwraca

{Object} Zawiera tablice wishlist, funkcje toggleWishlist oraz isInWishlist.

9.16 Dokumentacja pliku sklep_internetowy.client/src/hooks/useComparison.js

Hook React do zarzadzania stanem porownywarki produktow.

Zmienne

- const **MAX_PRODUCTS** = 2

  Maksymalna dopuszczalna liczba produktow w porownywarce.
- export const useComparison

  Glowny hook zarzadzajacy logika porownywania.

9.16.1 Opis szczegółowy

Hook React do zarzadzania stanem porownywarki produktow.
Modul umozliwia dodawanie, usuwanie oraz sprawdzanie obecnosci produktow w porownywarce. Implementuje logike kolejki (FIFO) przy osiagnieciu limitu produktow.

9.16.2 Dokumentacja zmiennych

9.16.2.1 useComparison

```
export const useComparison
```
Glowny hook zarzadzajacy logika porownywania.

Zwraca

{Object} Zestaw stanow i funkcji do obslugi listy porownawczej.

9.17 Dokumentacja pliku sklep_internetowy.client/src/i18n.js

Modul konfiguracji tlumaczen i lokalizacji aplikacji TechStore.

Funkcje

- i18n use (LanguageDetector) .use(initReactI18next) .init(

Zmienne

- const resources

  Obiekt zasobow zawierajacy mapowanie kluczy jezykowych na zaimportowane pliki tlumaczen.

9.17.1 Opis szczegółowy

Modul konfiguracji tlumaczen i lokalizacji aplikacji TechStore.
Plik odpowiada za inicjalizacje biblioteki i18next, automatyczne wykrywanie jezyka uzytkownika oraz mapowanie plikow JSON z tlumaczeniami na wspierane kody jezykowe (PL, EN).

9.17.2 Dokumentacja funkcji

9.17.2.1 use()

```
i18n use (
                LanguageDetector)
```

@description Inicjalizacja i konfiguracja i18next. @use LanguageDetector - Wykrywa preferowany jezyk na podstawie ustawien przegladarki lub ciasteczek. @use initReactI18next - Przekazuje instancje i18n do biblioteki react-i18next w celu obslugi hookow. Jezyk domyslny, uzywany gdy tlumaczenie dla wybranego jezyka nie jest dostepne. Wylaczenie eskapowania wartosci (React domyslnie chroni przed atakami XSS).

9.17.3 Dokumentacja zmiennych

9.17.3.1 resources

```
const resources
```

Wartość początkowa:

```
                = {
    en: {
        translation: translationEN
    },
    pl: {
        translation: translationPL
    }
}
```

Obiekt zasobow zawierajacy mapowanie kluczy jezykowych na zaimportowane pliki tlumaczen.
Struktura obejmuje jezyk angielski (en) oraz polski (pl).

9.18 Dokumentacja pliku sklep_internetowy.client/src/main.jsx

Glowny punkt wejscia (entry point) aplikacji TechStore.

9.18.1 Opis szczegółowy

Glowny punkt wejscia (entry point) aplikacji TechStore.
Plik odpowiada za zainicjowanie drzewa komponentow React, konfiguracje mechanizmu Suspense dla asynchronicznego ladowania tlumaczen oraz rejestracje dostawcow kontekstu (Context Providers) dla koszyka, listy zyczen i routingu.

9.19 Dokumentacja pliku sklep_internetowy.client/src/pages/Admin↩ Dashboard/AdminDashboard.jsx

Glowny panel sterowania (Dashboard) administratora systemu TechStore.

9.19.1 Opis szczegółowy

Glowny panel sterowania (Dashboard) administratora systemu TechStore.
Komponent sluzy jako centralny punkt nawigacyjny, agregujacy skroty do wszystkich kluczowych modulow administracyjnych: zamowien, uzytkownikow, produktow, wiadomosci oraz promocji.

9.20 Dokumentacja pliku sklep_internetowy.client/src/pages/Admin↩ Dashboard/messages/UserMessageManagement.jsx

Komponent panelu administracyjnego do zarzadzania wiadomosciami kontaktowymi.

9.20.1 Opis szczegółowy

Komponent panelu administracyjnego do zarzadzania wiadomosciami kontaktowymi.
Umozliwia przegladanie, tworzenie, edycje oraz usuwanie zgloszen przeslanych przez uzytkownikow.

9.21 Dokumentacja pliku sklep_internetowy.client/src/pages/Admin↩ Dashboard/Order/OrderManagement.jsx

Komponent panelu administracyjnego do kompleksowego zarzadzania zamowieniami.

Zmienne

- export default OrderManagement

9.21.1 Opis szczegółowy

Komponent panelu administracyjnego do kompleksowego zarzadzania zamowieniami.
Obsluguje pelny cykl zycia zamowienia (CRUD), w tym dynamiczna edycje listy produktow, automatyczna korekte stanow magazynowych oraz generowanie faktur PDF.

9.21.2 Dokumentacja zmiennych

9.21.2.1 OrderManagement

```
export default OrderManagement
```

Lista wszystkich zamowien pobrana z API.
Katalog produktow uzywany do walidacji i dodawania pozycji.
Lista uzytkownikow do przypisania przy tworzeniu recznym zamowienia.
Flaga okreslajaca stan ladowania danych.
Przechowuje komunikaty o bledach operacji asynchronicznych.
Stany widocznosci okien modalnych.
Dane zamowienia bedacego w procesie edycji (gleboka kopia).
Stan inicjalny dla nowego zamowienia.

Nota

React useEffect: Pobiera dane startowe (zamowienia, produkty, uzytkownicy) przy montowaniu komponentu.

Nota

Funkcja asynchroniczna @description Odswieza liste zamowien z serwera.

9.22 Dokumentacja pliku sklep_internetowy.client/src/pages/Admin↩ Dashboard/promotion/PromotionManagement.jsx

Komponent panelu administratora do zarzadzania kampaniami promocyjnymi.

9.22.1 Opis szczegółowy

Komponent panelu administratora do zarzadzania kampaniami promocyjnymi.
Umozliwia filtrowanie produktow spelniajacych kryteria promocji (np. stan magazynowy, brak aktywnosci) oraz masowe nakladanie znizek procentowych na wybrane towary.

9.23 Dokumentacja pliku sklep_internetowy.client/src/pages/Auction/AuctionDetails.jsx

Komponent widoku szczegolowego aukcji z obsluga licytacji w czasie rzeczywistym.

9.23.1 Opis szczegółowy

Komponent widoku szczegolowego aukcji z obsluga licytacji w czasie rzeczywistym.
Modul integruje sie z SignalR w celu zapewnienia natychmiastowych aktualizacji ceny bez koniecznosci odswiezania strony. Obsluguje uwierzytelnianie JWT oraz walidacje ofert.

9.24 Dokumentacja pliku sklep_internetowy.client/src/pages/Auction/AuctionList.jsx

Komponent wyswietlajacy liste aktywnych aukcji w systemie.

9.24.1 Opis szczegółowy

Komponent wyswietlajacy liste aktywnych aukcji w systemie.
Modul odpowiada za pobieranie danych o trwajacych licytacjach z API, ich prezentacje w formie kart oraz zapewnia nawigacje do szczegolow kazdej aukcji.

9.25 Dokumentacja pliku sklep_internetowy.client/src/pages/Auction/CreateAuction.jsx

Komponent umozliwiajacy administratorowi tworzenie nowych aukcji.

9.25.1 Opis szczegółowy

Komponent umozliwiajacy administratorowi tworzenie nowych aukcji.
Modul pobiera liste produktow z katalogu i pozwala na wystawienie ich na licytacje z okreslona cena startowa.

9.26 Dokumentacja pliku sklep_internetowy.client/src/pages/Cart/Cart.jsx

Komponent widoku koszyka zakupowego aplikacji TechStore.

Zmienne

- export default Cart

9.26.1 Opis szczegółowy

Komponent widoku koszyka zakupowego aplikacji TechStore.
Modul odpowiada za prezentacje przedmiotow dodanych przez uzytkownika, umozliwia modyfikacje ich ilosci, usuwanie pozycji oraz inicjuje proces skladania zamowienia.

9.26.2 Dokumentacja zmiennych

9.26.2.1 Cart

```
export default Cart
```

Dane i metody pobrane z globalnego stanu koszyka.
@description Obsluguje proces przejscia do kasy (Checkout).

Sprawdza obecnosc zalogowanego uzytkownika w localStorage. Jesli uzytkownik nie jest uwierzytelniony, wyswietla komunikat i przekierowuje do logowania.

9.27 Dokumentacja pliku sklep_internetowy.client/src/pages/Forgot↩ Password/ForgotPasswordPage.jsx

Komponent strony odzyskiwania zapomnianego hasla.

9.27.1 Opis szczegółowy

Komponent strony odzyskiwania zapomnianego hasla.
Pozwala uzytkownikowi na wprowadzenie adresu e-mail, na ktory zostanie wyslany link resetujacy dostep do konta.

9.28 Dokumentacja pliku sklep_internetowy.client/src/pages/Home/Home.jsx

Glowny komponent strony glownej aplikacji TechStore.

9.28.1 Opis szczegółowy

Glowny komponent strony glownej aplikacji TechStore.
Modul odpowiada za wyswietlanie strony startowej, agregujac kluczowe elementy interfejsu takie jak karuzela promocyjna oraz siatka produktow.

9.29 Dokumentacja pliku sklep_internetowy.client/src/pages/LoginPage/LoginPage.jsx

Komponent strony logowania uzytkownika.

9.29.1 Opis szczegółowy

Komponent strony logowania uzytkownika.
Modul obsluguje interfejs logowania, walidacje po stronie klienta oraz komunikacje z serwerem w celu uzyskania tokenu autoryzacyjnego JWT.

9.30 Dokumentacja pliku sklep_internetowy.client/src/pages/Payment/PaymentPage.jsx

Komponent obslugujacy proces platnosci elektronicznych Stripe oraz finalizacje zamowienia.

Zmienne

- const **stripePromise** = loadStripe(import.meta.env.VITE_STRIPE_PUBLIC_KEY)

  Inicjalizacja obiektu Stripe przy uzyciu klucza publicznego ze zmiennych srodowiskowych.

9.30.1 Opis szczegółowy

Komponent obslugujacy proces platnosci elektronicznych Stripe oraz finalizacje zamowienia.
Modul integruje sie z bramka platnosci Stripe, zarzadza tworzeniem PaymentIntent na serwerze, potwierdzaniem transakcji oraz automatycznym generowaniem faktury PDF po sukcesie.

9.31 Dokumentacja pliku sklep_internetowy.client/src/pages/Products/ProductDetails.jsx

Komponent widoku szczegolowego produktu przeznaczony dla panelu administracyjnego.

Zmienne

- export default ProductDetails

9.31.1 Opis szczegółowy

Komponent widoku szczegolowego produktu przeznaczony dla panelu administracyjnego.
Modul odpowiada za wyswietlanie pelnych informacji o konkretnym produkcie, umozliwia przelaczenie w tryb edycji przy uzyciu ProductForm oraz wykonanie operacji usuniecia produktu z systemu.

9.31.2 Dokumentacja zmiennych

9.31.2.1 ProductDetails

```
export default ProductDetails
```

Pobranie identyfikatora produktu z parametrow sciezki URL.
Stan przechowujacy szczegolowe dane pobranego produktu.
Flaga kontrolujaca wyswietlanie stanu ladowania.
Przechowuje komunikaty o bledach operacji API.
Flaga okreslajaca widocznosc formularza edycji.
Bazowy adres URL dla punktu koncowego API produktow.

Nota

React useEffect: Inicjalizacja danych. @description Pobiera dane produktu przy kazdym zamontowaniu komponentu lub zmianie ID.

Nota

Funkcja asynchroniczna @description Pobiera z serwera dane konkretnego produktu na podstawie identyfikatora.

9.32 Dokumentacja pliku sklep_internetowy.client/src/pages/Products/Shop/ProductDetailsShop.jsx

Komponent wyswietlajacy szczegolowe informacje o produkcie w interfejsie sklepu.

9.32.1 Opis szczegółowy

Komponent wyswietlajacy szczegolowe informacje o produkcie w interfejsie sklepu.
Zarzadza pobieraniem danych o konkretnym towarze, interaktywna galeria zdjec oraz integracja z systemem koszyka i porownywarki produktow.

9.33 Dokumentacja pliku sklep_internetowy.client/src/pages/Products/Shop/SearchPage.jsx

Komponent wyswietlajacy produkty przypisane do konkretnej kategorii (slug).

9.33.1 Opis szczegółowy

Komponent wyswietlajacy produkty przypisane do konkretnej kategorii (slug).
Komponent strony wynikow wyszukiwania z zaawansowanym filtrowaniem i sortowaniem.
Komponent strony porownywarki produktow.
Modul odpowiada za pobieranie metadanych kategorii oraz powiazanych z nia produktow, umozliwiajac ich przegladanie w formie listy oraz dodawanie do koszyka.
Modul umozliwia uzytkownikowi zestawienie cech dwoch wybranych produktow w formie tabeli. Dane sa pobierane dynamicznie z API na podstawie listy identyfikatorow z kontekstu.
Modul odpowiada za prezentacje wynikow wyszukiwania tekstowego, umozliwiajac uzytkownikowi dynamiczne zawezanie listy produktow wedlug ceny, marki oraz statusu promocji.

9.34 Dokumentacja pliku sklep_internetowy.client/src/pages/Registration/Registration.jsx

Komponent strony rejestracji nowego uzytkownika.

9.34.1 Opis szczegółowy

Komponent strony rejestracji nowego uzytkownika.
Modul obsluguje formularz tworzenia konta, walidacje zgodnosci hasel oraz komunikacje z API uwierzytelniania w celu rejestracji uzytkownika.

9.35 Dokumentacja pliku sklep_internetowy.client/src/pages/UserProfile/UserProfile.jsx

Komponent widoku profilu uzytkownika wraz z historia zamowien.

9.35.1 Opis szczegółowy

Komponent widoku profilu uzytkownika wraz z historia zamowien.
Modul umozliwia zalogowanemu uzytkownikowi edycje danych profilowych (nazwa, email) oraz przegladanie zestawienia archiwalnych zamowien z mozliwoscia pobrania faktur PDF.

9.36 Dokumentacja pliku sklep_internetowy.client/src/pages/WishList/WishlistPage.jsx

Komponent strony listy zyczen (ulubionych produktow).

Zmienne

- export default WishlistPage

9.36.1 Opis szczegółowy

Komponent strony listy zyczen (ulubionych produktow).
Modul umozliwia uzytkownikowi przegladanie produktow zapisanych "na pozniej", usuwanie ich z listy oraz szybkie dodawanie wybranych pozycji do koszyka zakupowego.

9.36.2 Dokumentacja zmiennych

9.36.2.1 WishlistPage

```
export default WishlistPage
```

Pobranie danych i funkcji sterujacych z kontekstu listy zyczen.

Funkcja dodawania produktu do koszyka pobrana z CartContext.
Adres URL domyslnego obrazu uzywanego w przypadku braku zdjec produktu.

-

9.36.3 EmptyState

Obsluga wyswietlania komunikatu w przypadku braku produktow na liscie.
Logika wyboru glownego zdjecia produktu.

9.37 Dokumentacja pliku Sklep_internetowy.Server/Program.cs

Główny punkt wejścia aplikacji TechStore, odpowiedzialny za konfigurację usług i potoku HTTP.

Funkcje

- builder.Services. AddCors (options=> { options.AddPolicy("AllowReactApp", policy=> { policy .With↩ Origins("http://localhost:5173", "https://localhost:5173", "http://localhost:5174", "https://localhost:5174", "http://localhost:5084", "https://localhost:5084") .AllowAnyMethod() .AllowAnyHeader() .AllowCredentials();});})
- builder.Services. AddScoped< AuctionService > ()
- builder.Services. AddControllers () .AddJsonOptions(options
- builder.Services. AddIdentity< User, IdentityRole > (options=> { options.Password.RequireDigit=false;options.↩ Password.RequireLowercase=false;options.Password.RequireUppercase=false;options.Password.Require↩ NonAlphanumeric=false;options.Password.RequiredLength=6;options.User.RequireUniqueEmail=true;}) .AddEntityFrameworkStores< StoreDbContext >() .AddDefaultTokenProviders() .AddRoles< IdentityRole >()
- builder.Services. AddSignalR () .AddHubOptions< AuctionHub >(options

Zmienne

- var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")
- var secretKey

9.37.1 Opis szczegółowy

Główny punkt wejścia aplikacji TechStore, odpowiedzialny za konfigurację usług i potoku HTTP.
Plik ten zawiera definicję kontenera wstrzykiwania zależności (Dependency Injection), konfigurację systemów bezpieczeństwa (CORS, Identity, JWT), rejestrację usług biznesowych oraz ustawienia middleware sterującego przepływem żądań.

9.37.2 Dokumentacja funkcji

9.37.2.1 AddControllers()

```
builder.Services. AddControllers ()  [inline]
```

9.37.3 Controllers

Konfiguracja kontrolerów API z obsługą cykli w serializacji JSON.

9.37.3.1 AddCors()

```
builder.Services. AddCors (
            options ,
            { options.AddPolicy("AllowReactApp", policy=> { policy .WithOrigins("http:/↩
/localhost:5173", "https://localhost:5173", "http://localhost:5174", "https://localhost:↩
5174", "http://localhost:5084", "https://localhost:5084") .AllowAnyMethod() .AllowAnyHeader()
.AllowCredentials();});} )
```

9.37.4 CORS

Konfiguracja polityki Cross-Origin Resource Sharing (CORS).

Nota

MUSI być zarejestrowana przed innymi usługami, aby umożliwić komunikację z frontendem React.

9.37.4.1 AddIdentity< User, IdentityRole >()

```
builder.Services. AddIdentity< User, IdentityRole > (
            options ,
            { options.Password.RequireDigit=false;options.Password.RequireLowercase=false;options.↩
Password.RequireUppercase=false;options.Password.RequireNonAlphanumeric=false;options.Password.↩
RequiredLength=6;options.User.RequireUniqueEmail=true;} )
```

9.37.5 Identity

Konfiguracja systemu ASP.NET Core Identity dla autoryzacji opartej na użytkownikach i rolach.

9.37.5.1 AddScoped< AuctionService >()

```
builder.Services. AddScoped< AuctionService > ()
```

9.37.6 BusinessServices

Rejestracja usług logicznych i pomocniczych aplikacji.

9.37.6.1 AddSignalR()

```
builder.Services. AddSignalR ()  [inline]
```

9.37.7 SignalR

Rejestracja usług czasu rzeczywistego (WebSockets).

9.37.8 Dokumentacja zmiennych

9.37.8.1 connectionString

```
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")
```

9.37.9 Database

Rejestracja kontekstu bazy danych PostgreSQL (Entity Framework Core).

9.37.9.1 secretKey

```
var secretKey
```

Wartość początkowa:

```
= builder.Configuration.GetValue<string>("AuthSettings:Key") ??
  builder.Configuration.GetValue<string>("AuthSettings:SecretKey")
```

9.37.10 Authentication

Konfiguracja uwierzytelniania opartego na tokenach JWT oraz obsługa protokołu SignalR.

Skorowidz