

Dokumentacja Techniczna

Aplikacji internetowej „LegoShop”

Krystian Wawruch
Jakub Walasek
2024

Spis treści:

| | |
|--------|---------------------------------------|
| 1 | Tytuł, autorzy, spis treści |
| 2...3 | Wprowadzenie |
| 4 | Dokumentacja środowiskowa ASP.NET MVC |
| 5...16 | Dokumentacja Techniczna |

Wprowadzenie

1.

Celem projektu LegoShop jest stworzenie interaktywnej platformy internetowej opartej na technologii ASP.NET MVC. Strona umożliwi użytkownikom rejestrację, logowanie, zakup oraz tworzenie produktów - mozaiki z klocków Lego. Projekt ma na celu dostarczenie innowacyjnego doświadczenia zakupowego, które integruje pasję do klocków Lego z interaktywnymi narzędziami do projektowania własnych unikatowych mozaik.

2.

Opis funkcjonalności:

- Rejestracja, logowanie, wylogowanie

Użytkownicy mogą zarejestrować się, podając podstawowe informacje takie jak email oraz hasło. Rejestrację należy potwierdzić klikając w przycisk do potwierdzenia rejestracji. Logowanie następuje poprzez wpisanie swojego adresu email oraz hasła. Istnieją funkcje: zapamiętywania (funkcja jest już dostępna), oraz wstępne funkcje, które wymagają dalszej implementacji tj. zapomnienie hasła i ponowne wysłanie maila z potwierdzeniem rejestracji konta.

- Konfiguracja dostępu do części ukrytych

Aplikacja pozwala na rejestrację użytkowników, jednak dostęp do części ukrytych ma jedynie administrator, którego konto tworzy się automatycznie po uruchomieniu aplikacji i stworzeniu bazy danych. Dane dostępu do konta administratora to:

Email: admin@test.pl Hasło: Password12@

Dane użytkowników jak i zamówienia są przechowywane w bazie danych. Zamówienia będą przypisane do danego konta.

- Tworzenie Produktów

Użytkownicy mogą stworzyć produkt – mozaikę na podstawie linku URL zdjęcia, które jest umieszczone na jakimś serwerze w sieci. Polecamy do tego stronę imgur.com. Po wczytaniu linku, aplikacja wczyta zdjęcie i doda do zamówienia. Podczas tworzenia produktu można nazwać produkt, wybrać typ klocka, rozmiar ramki,

kolor obramowania i wkleić link. Cena, którą trzeba zapłacić za produkt jest adekwatna do rozmiaru ramki produktu.

- [Zamawianie produktu](#)

Użytkownicy po stworzeniu produktu, mogą go zamówić klikając w przycisk „kup produkt” w zakładce produkty. Wyświetla się następnie szczegóły produktu i przycisk do zamówienia produktu.

- [Elementy dostępne tylko dla administratora](#)

Administrator dodatkowo ma podgląd do zamówień wszystkich użytkowników, może podpatrzeć szczegóły zamówienia, edytować zamówienie lub je usunąć, a także anulować. Ponadto ma dostęp do listy użytkowników aplikacji, może edytować dane użytkownika lub go całkowicie usunąć.

Jeśli dane elementy interfejsu front-endu lub back-endu nie zostały wyżej wymienione najprawdopodobniej znajdują się w części technicznej.

Dokumentacja środowiskowa ASP.NET MVC

1.

Wymagania systemowe:

- Visual Studio (zalecane jest pobranie najnowszej wersji ze strony Microsoft)
- Zestaw SDK platformy .NET
- SQL Server, baza danych, w której przechowywane będą dane aplikacji.

2.

Projekt jest podzielony na kontrolery, widoki, modele zgodnie z wzorcem architektonicznym MVC. Widoki (folder Views) odpowiadają za prezentację danych użytkownikowi, natomiast kontrolery (folder Controllers) obsługuje żądania użytkownika, współpracując z modelem i widokiem.

Dokumentacja techniczna

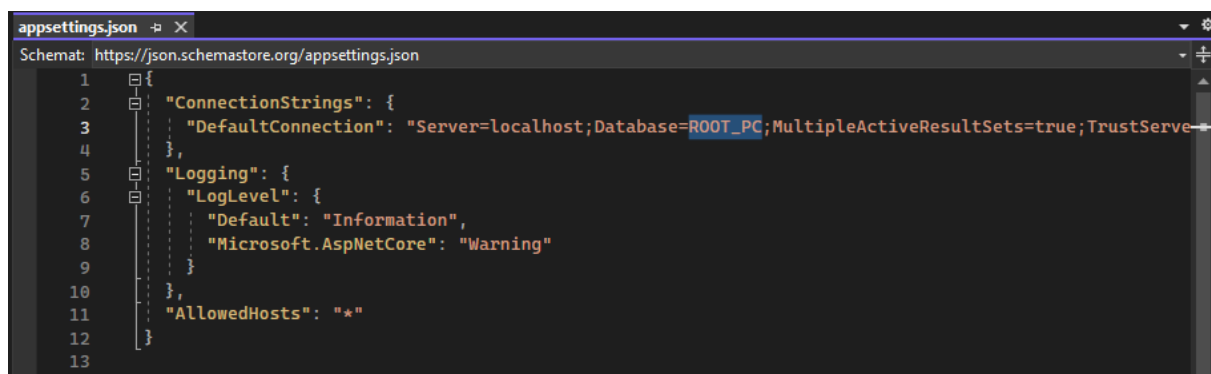
1.

Wstępna konfiguracja aplikacji:

Aby aplikacja została uruchomiona poprawnie na dowolnym komputerze ważna jest podstawowa znajomość MySQL oraz konfiguracji MySQL u siebie na komputerze. Konieczna jest zmiana wartości parametru „Database” w pliku konfiguracyjnym appsettings.json

Ścieżka do pliku - LegoShop\apssettings.json

Linijka nr 3 powinna zostać zmieniona o nazwę Twojej bazy w miejsce w zamian za „ROOT_PC”, przykład został umieszczony poniżej. W tym wypadku nazwa bazy u kogoś kto testował aplikację nazywa się ROOT_PC.



```
1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Server=localhost;Database=ROOT_PC;MultipleActiveResultSets=true;TrustServerCertificate=true",
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Information",
8       "Microsoft.AspNetCore": "Warning"
9     }
10  },
11  "AllowedHosts": "*"
12 }
13
```

2.

W aplikacji zostały zainstalowane następujące pakiety NuGet:

- microsoft.aspnetcore.diagnostics.entityframeworkcore
- microsoft.aspnetcore.identity.entityframeworkcore
- microsoft.aspnetcore.identity.ui
- microsoft.entityframeworkcore.sqlserver
- microsoft.entityframeworkcore.tools
- microsoft.visualstudio.web.codegeneration.design

3.

Logika działania pasku nawigacji

Pasek nawigacji aplikacji LegoShop z poziomu konta administratora zawiera następujące podstrony: Strona główna, Produkty, Zamówienia, Statusy zamówień, Użytkownicy, Zarejestruj się, Zaloguj się. Po logowaniu możemy zauważyć zmianę w pasku nawigacji. Dostajemy opcję wylogowania oraz widać nasz email. Użytkownik z wiadomych przyczyn nie ma dostępu do Statusu zamówień i listy użytkowników.

Część widokowa paska nawigacji składa się między innymi z pliku _Layout.cshtml

LegoShop\Views\Shared_Layout.cshtml

Widok użytkownika niezalogowanego



Widok administratora



Widok użytkownika



Między 32 a 49 liniijką w pliku _Layout.cshtml zawarta jest część logiczna dostępu do zakładki widocznych tylko dla administratora

```
@if(User.Identity.IsAuthenticated)
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Products" asp-action="Index">Produkty</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Orders" asp-action="Index">Zamówienia</a>
    </li>
    @if(User.IsInRole("Admin"))
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="OrderStatus" asp-action="Index">Statusy zamówień</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="ApplicationUser" asp-action="Index">Użytkownicy</a>
        </li>
    }
}
```

Część widokowa logowania w pasku nawigacji składa się z pliku `_LoginPartial.cshtml`

`LegoShop\Views\Shared_LoginPartial.cshtml`

```
<ul class="navbar-nav">
@if (SignInManager.IsSignedIn(User))
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/* Comments can contain keywords such as
for and while without generating errors. */t/Manage/Index" title="Manage">Witaj @User.Identity?.Name!</a>
    </li>
    /* WYLOGUJ SIĘ */
    <li class="nav-item">
        <form class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">
            <button type="submit" class="nav-link btn btn-link text-dark">Wyloguj się</button>
        </form>
    </li>
    /* WYLOGUJ SIĘ */
}
else
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Zarejestruj się</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Zaloguj się</a>
    </li>
}
</ul>
```

3.1

Zabezpieczenie przed nieautoryzowanym wejściem w podstronę dostępną tylko dla administratora działa w prosty sposób. Kiedy przykładowo chcemy skopiować link do schowka ze ścieżką do podstrony dostępnej tylko dla administratora, wyskoczy nam błąd – Access Denied.

Strona główna Produkty Zamówienia

Witaj użytkownik@wp.pl! [Wyloguj się](#)

Access denied

You do not have access to this resource.

4.

Logika działania tworzenia produktu

Do tworzenia produktu możemy przejść na 2 sposoby, klikając w przyciski:

- „Rozpocznij” na stronie głównej
- „Zacznij teraz” na stronie z listą produktów

Strona główna Produkty Zamówienia Statusy zamówień Użytkownicy

Witaj admin@test.pl! [Wyloguj się](#)

ROZPOCZNIJ

Stwórz swoją mozaikę z LEGO

ZACZNIJ TERAZ!

Po przejściu do strony tworzenia produktu mamy pola do uzupełnienia lub wybrania.

Możliwe jest wprowadzenie:

- Nazwy produktu
- Linku do zdjęcia (najlepiej gdy jest to zdjęcie w rozmiarach 100x100,500x500,1000x1000 etc.)
- Opisu produktu

Oraz możliwe jest wybranie:

- Typu klocka mozaiki (okrągłego lub kwadratowego)
- Rozmiaru ramki (Small, medium i big)
- Koloru ramki (Red, Green, Blue, Yellow, White, Black, Brown)

Na samym dole widnieje cena którą musimy zapłacić za produkt, zmienia się ona adekwatnie do wybranego rodzaju ramki.

- Small – 30cmx30cm – Cena – 50PLN
- Medium – 45cmx45cm – Cena – 75PLN
- Big – 60cmx60cm – Cena – 100PLN

Po wciśnięciu przycisku Create, produkt zostaje przypisany do konta oraz widnieje w liście produktów.

Produkty stworzone przez jednego użytkownika nie będą widoczne dla drugiego użytkownika chyba, że jest nim administrator.

Stwórz swoją mozaikę z LEGO

ZACZNIJ TERAZ!

| Nazwa produktu | Opis | Cena | Typ klocka | Rozmiar ramki | Kolor ramki | Twój obrazek | |
|----------------|------------------------|-------|------------|---------------|-------------|---|---|
| Grogu | Grogu from Mandalorian | 75,00 | Round | Medium | Black |  | KUP PRODUKT SZCZEGÓŁY |

Po stworzeniu produktu wyświetlają się nam dwa przyciski:

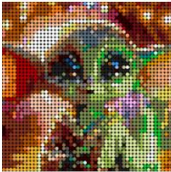
- Kup produkt (Funkcja ta zostanie omówiona w punkcie 5)
- Szczegóły

Po kliknięciu w szczegóły mamy opis produktu, który stworzyliśmy. Zawiera on wszystkie informacje jakie wpisaliśmy poprzednio w formularzu. Wyświetla się również podgląd zdjęcia, które wkleiliśmy za pomocą linku URL.

Strona główna Produkty Zamówienia Statusy zamówień Użytkownicy Witaj admin@test.pl! Wyloguj się

Details

Product

| | |
|----------------|--|
| Nazwa produktu | Grogu |
| Description | Grogu from Mandalorian |
| Price | 75,00 |
| MosaicType | Round |
| FrameSize | Medium |
| FrameColor | Black |
| ImageUrl |  |

[Edit](#) | [Delete](#) [Back to List](#)

Część logiczna tworzenia produktu zawarta jest w pliku

LegoShop\Views\Products\Create.cshtml

oraz niezbędny do działania jest też kontroler ProductsController.cs

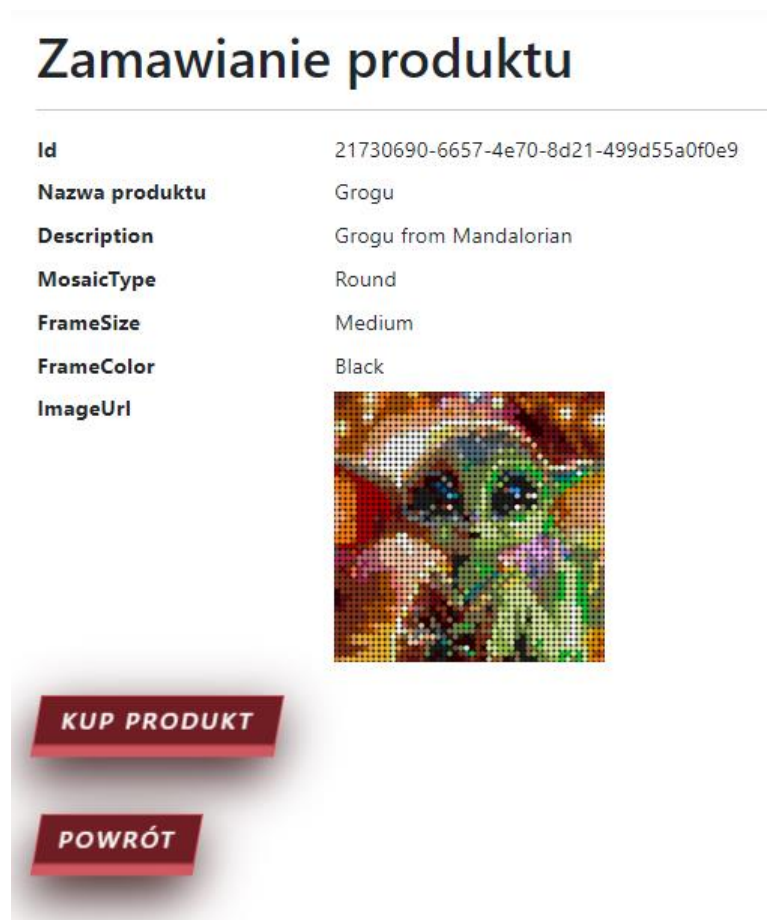
LegoShop\Controllers\ProductsController.cs

Opisy kontrolerów znajdują się na stronie nr 15

5.

Logika działania kupowania produktu

Produkt, który został stworzony możemy kupić klikając przycisk „Kup produkt” w zakładce „Produkty”. Można również powrócić do listy produktów klikając w przycisk „Powrót”. W zakładce kupowania produktu są również podane szczegóły produktu wraz z podglądem obrazka.



Część logiczna kupowania produktu zawarta jest w pliku

LegoShop\Views\Products\CreateOrder.cshtml

oraz niezbędny do działania jest też kontroler OrderController.cs

LegoShop\Controllers\OrderController.cs

Opisy kontrolerów znajdują się na stronie nr 15

Twoje zamówienia

| OrderDate | TotalPrice | Product | User | OrderStatus | |
|---------------------|------------|-----------------------|---------------|-------------|---|
| 13.01.2024 23:39:20 | 75,00 | Grogu | admin@test.pl | New | Anuluj zamówienie Szczegóły Edytuj Usuń |

Po kupieniu produktu w zakładce „Zamówienia” powinno się wyświetlić twoje zamówienie zawierające:

- Datę zamówienia
- Cenę
- Nazwę produktu
- Użytkownika, który zamówił produkt
- Status zamówienia
- Listę czynności
 - Anulowanie zamówienia
 - Szczegóły
 - Edycję
 - Usuwanie

5.

Logika działania statusów zamówień

Lista statusów zamówień

Dostęp tajny

[Stwórz nowy status](#)

| ConstId | Name | |
|---------|-----------------|---|
| 1 | New | Edytuj status |
| 2 | Payment recived | Edytuj status Usuń status |
| 3 | Payment failed | Edytuj status Usuń status |
| 4 | In progress | Edytuj status |
| 5 | Completed | Edytuj status |
| 6 | Closed | Edytuj status |
| 7 | Canceled | Edytuj status |

Na stronie dodana została strona, do której dostęp ma tylko administrator. Są to „Statusy zamówień”. Zawiera ona statusy, które można przydzielić do zamówienia. Status „New” przydziela się do zakupionego produktu automatycznie.

Część logiczna kupowania produktu zawarta jest w pliku index.cshtml

LegoShop\Views\OrderStatus\index.cshtml

oraz niezbędny do działania jest też kontroler OrderStatusController.cs

LegoShop\Controllers\OrderStatusController.cs

Dane, które są wyświetlane w tabeli tj. statusy, pobierane są z tabeli bazy danych, która tworzona jest za pomocą pliku OrderStatus.cs w folderze:

LegoShop\Data\Entities\OrderStatus.cs

6.

Logika działania listy użytkowników

Użytkownicy

| Id | UserName | Email | EmailConfirmed | |
|--------------------------------------|------------------|------------------|--------------------------|---------------------------------|
| d577258e-1fa4-40a2-80ee-e23c8a08ffc8 | uzytkownik@wp.pl | uzytkownik@wp.pl | <input type="checkbox"/> | Szczegóły konta |
| df6a122e-cbf9-45c8-a569-07cc179f7f0a | admin@test.pl | admin@test.pl | <input type="checkbox"/> | Szczegóły konta |

Na stronie dodana została strona, do której dostęp ma tylko administrator. Są to „Użytkownicy”. Zawiera ona listę użytkowników.

Część logiczna kupowania produktu zawarta jest w pliku Index.cshtml

LegoShop\Views\ApplicationUser\Index.cshtml

oraz niezbędny do działania jest też kontroler ApplicationUser.cs

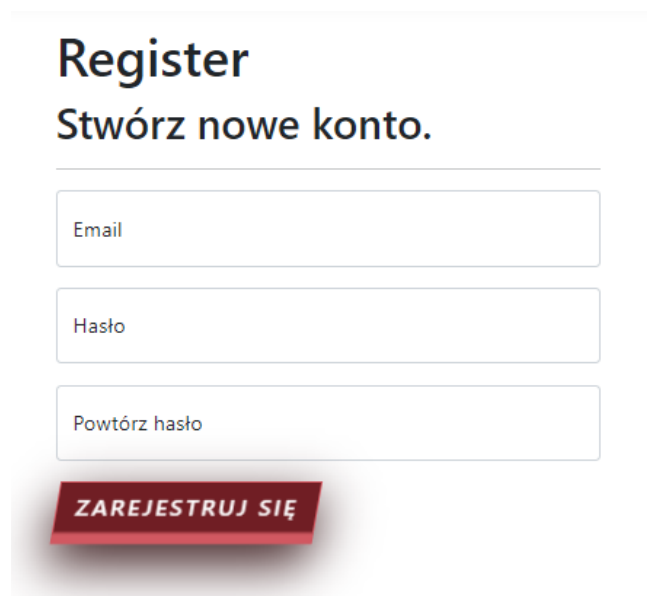
LegoShop\Controllers\ApplicationUserController.cs

Dane, które są wyświetlane w tabeli tj. użytkownicy, pobierane są z tabeli bazy danych, która tworzona jest za pomocą pliku ApplicationUser.cs w folderze:

LegoShop\Data\Entities\ApplicationUser.cs

7.

Logika działania rejestracji i logowania



Register
Stwórz nowe konto.

Email

Hasło

Powtórz hasło

ZAREJESTRUJ SIĘ

Strona zawiera opcję rejestracji. Można się zarejestrować wpisując email, hasło i potwierdzając hasło.

Po zarejestrowaniu wyskoczy nam komunikat dot. Zatwierdzenia konta. Obowiązkowo należy kliknąć zielony przycisk potwierdzający założenie konta.



Potwierdź rejestrację

KLIKNIJ TUTAJ ABY POTWIERDZIĆ REJESTRACJĘ

Część logiczna rejestracji konta zawarta jest w pliku Register.cshtml

LegoShop\Areas\Identity\Pages\Account\Register.cshtml

Strona główna

Zaloguj się

☐ Zapamiętaj mnie

ZALOGUJ SIĘ

Strona zawiera opcję logowania . Można się zalogować wpisując email oraz hasła.
Jest możliwość zapamiętania hasła.

8.

Backend aplikacji

Aplikacja zawiera 5 tabel, których implementacja znajduje się w folderze LegoShop\Data\Entities

- ApplicationUser - rozszerzenie klasy IdentityUser z paczki Microsoft.AspNet.Identity
- Address - zawierające dane adresowe
- Order - zawiera zamówienia
- OrderStatuses - zawiera możliwe statusy zamówień
- Product - zawiera produkty (mozaiki lego)

Aplikacja zawiera 5 kontrolerów znajdujących się w folderze Controllers

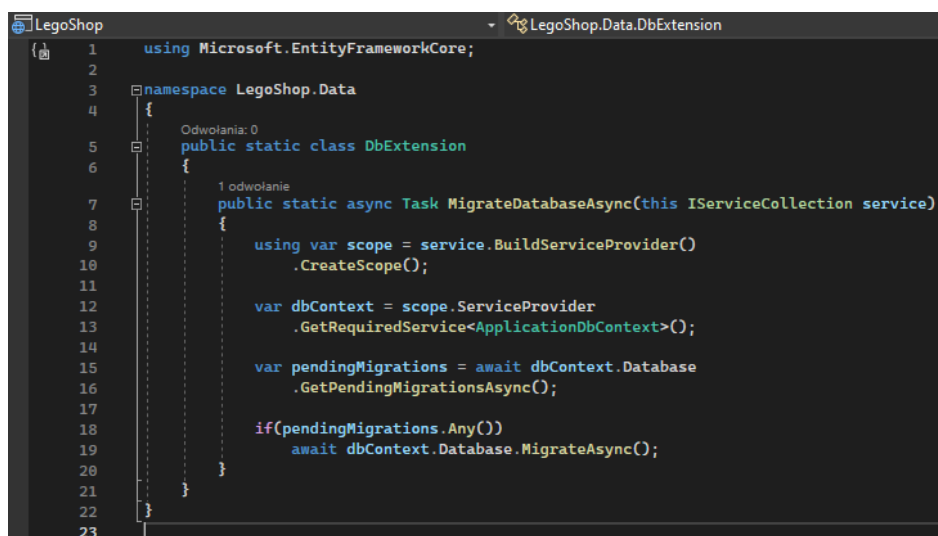
- ApplicationController - odpowiada za zarządzanie użytkownikiem
- HomeController - odpowiada za stronę główną oraz stronę plityki prywatności
- OrdersController - odpowiada za zarządzanie zamówieniami
- OrderStatusController - odpowiada za zarządzanie statusami zamówień
- ProductController - odpowiada za zarządzanie produktami

Aplikacja zawiera również folder z metodą odpowiedzialną za wykonywanie automatycznych migracji po uruchomieniu aplikacji dzięki czemu nie jest wymagane wpisywanie polecenia update-database podczas nanoszenia zmian na bazę danych z poziomu kodu.

Implementacja klasy znajduje się w

LegoShop\LegoShop\Data\DbExtension.cs

jako metoda rozszerzająca IServiceCollection – MigrateDatabaseAsync



```
1 using Microsoft.EntityFrameworkCore;
2
3 namespace LegoShop.Data
4 {
5     public static class DbExtension
6     {
7         public static async Task MigrateDatabaseAsync(this IServiceCollection service)
8         {
9             using var scope = service.BuildServiceProvider()
10                .CreateScope();
11
12             var dbContext = scope.ServiceProvider
13                .GetRequiredService<ApplicationDbContext>();
14
15             var pendingMigrations = await dbContext.Database
16                .GetPendingMigrationsAsync();
17
18             if(pendingMigrations.Any())
19                 await dbContext.Database.MigrateAsync();
20         }
21     }
22 }
23
```

Aplikacja zawiera również folder z klasami odpowiedzialnymi za Seedowanie danych (Automatycznie generowanie podstawowych danych) w folderze LegoShop\Data\Seeders

gdzie w podfolderze Seeds znajdują się klasy (ApplicationUserSeeder, AspNetRoleSeeder oraz OrderStatusesSeeder)implementujące interfejs IEntitySeeder z metodą Seed. W kolejnym podfolderze SeedService znajduje się klasa SeedService z metodą ExecuteSeeds wywołującą seeder dla każdej zarejestrowanej klasy poprzez klasę RegisterSeeds.

