SportStore

Projekt z przedmiotu Techniki Internetu

Konrad Maciałek

26 czerwca 2018

Spis treści

1 Odnośniki

1.1 Aplikacja

1.1.1 Część kliencka

`https://anothersportstore.azurewebsites.net`

1.1.2 Część administracyjna

Login: admin
Hasło: Secret123$

Administracja zamówieniami:
`https://anothersportstore.azurewebsites.net/Order/List`

Administracja produktami:
`https://anothersportstore.azurewebsites.net/Admin/Index`

1.2 Kod źródłowy

`https://github.com/maciakon/SportStore`

2 Cel projektu

Zaprojektowanie i wykonanie aplikacji internetowej spełniającej poniższe warunki:

1. Dowolna tematyka projektu
2. Serwis dostępny przez przeglądarkę internetową
3. Końcową warstwę prezentacji muszą stanowić poprawne składniowo i semantycznie dokumenty HTML
4. Baza danych musi składać się z co najmniej trzech tabel, pomiędzy którymi istnieją relacje
5. Serwis musi umożliwiać wprowadzanie danych do bazy, modyfikację i usuwanie danych poprzez przeglądarkę internetową (interfejs HTML)
6. Musi istnieć część administracyjna serwisu (np. przeznaczona do uzupełniania danych w bazie) zabezpieczona (poprzez mechanizm logowania) przed nieautoryzowanym dostępem.

3 Słowo wstępne

Moim osobistym celem przy realizacji tego projektu, było nauczenie się podstaw projektowania aplikacji internetowych, przy wykorzystaniu jednej z nowoczesnych technologii dostępnych na rynku. Implementując aplikację, opierałem

się na doskonałej książce Adama Freemana "Pro ASP.NET Core MVC". Podążałem zaproponowanym przez autora tutorialem. Z tego powodu kod i wygląd aplikacji może wydać się znajomy, gdyż podręcznik ten obecny jest na rynku już od dłuższego czasu.

4 Założenia projektowe

4.1 Tematyka projektu

Zaprojektowanie prostego serwisu sklepu sportowego, posiadającego kilka kategorii towarów.

Serwis powinien umożliwiać:

1. Przeglądanie listy oferowanych towarów.
2. Dodawanie i usuwanie towarów z koszyka zakupów.
3. Składanie zamówienia wraz z podaniem adresu wysyłki.
4. Dodawanie, usuwanie i modyfikację towarów poprzez zabezpieczoną logowaniem sekcję administracyjną.

4.2 Wybór technologii

Przyjęto poniższe założenia:

4.2.1 Środowisko developerskie

1. System operacyjny: Linux
2. Baza danych: SQLite

4.2.2 Środowisko produkcyjne

1. System operacyjny: Windows
2. Baza danych: Azure SQL Server

4.2.3 Hosting

Microsoft Azure App Service

4.2.4 Wybór technologii

Z uwagi na założoną wieloplatformowość, do realizacji projektu wybrano technologię ASP.NET Core MVC w wersji 2.1.

5 Wykonanie

5.1 Organizacja projektu

Projekt składa się z trzech katalogów:

src- kod źródłowy aplikacji

tests- kod źródłowy testów jednostkowych

doc- dokumentacja projektu

5.2 Ogólny opis implementacji

Zgodnie z kanonem MVC, odpowiedzialność za przetwarzanie żądań klientów rozłożono na kilka kontrolerów, które po pobraniu lub modyfikacji odpowiedniego modelu, generują widoki HTML wysyłane do przeglądarki użytkownika.

5.2.1 Baza danych

W celu ułatwienia implementacji dostępu do bazy danych użyto Entity Framework Core oraz dostarczanego przez nią mechanizmu migracji.
W zależności od środowiska (produkcyjnego lub deweloperskiego) migracje nakładane są na odpowiednie technologie bazodanowe- Azure SQL Server oraz SQLite.

5.2.2 Stylizacja widoków

Zastosowanie biblioteki CSS Bootstrap 4.1.1 umożliwiło:

- ukształtowanie spójnego wyglądu aplikacji
- dostosowanie interfejsu do wyświetlania na urządzeniach o różnej szerokości ekranu

Biblioteka Font Awesome została wykorzystana do prezentacji ikony koszyka zakupów.

Wszystkie biblioteki CSS i JavaScript wykorzystane w projekcie dostarczane są klientowi przez odpowiednie serwery Content Delivery Network. Pozwala to na założenie z dużym prawdopodobieństwem, że skrypty będą wczytywane z lokalnego cache'u przeglądarki.

5.2.3 Dodatkowe biblioteki

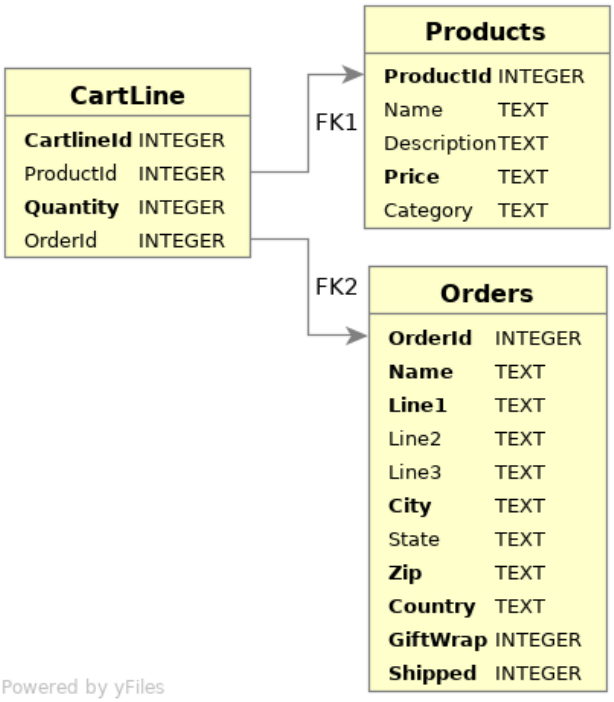
Dodatkowo użyto biblioteki jQuery w celu walidacji niektórych formularzy po stronie klienta.

5.3 Niektóre szczegóły implementacyjne

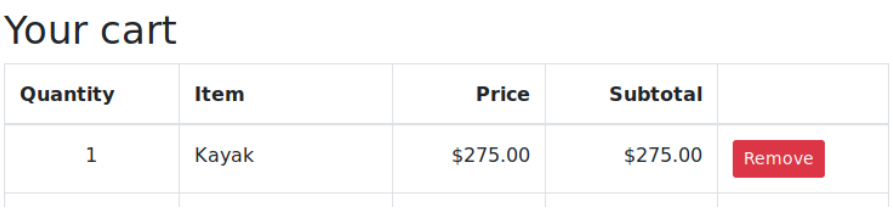
5.3.1 Bazy danych: tabele i relacje

Aplikacja używa dwóch baz danych:

- **Sportstore.db** dla przechowywania informacji biznesowej związanej z produktami, koszykiem i zamówieniami



Rysunek 1: Tabele i relacje bazy Sportstore.db



Rysunek 2: Linia zamówienia koszyka zakupów

- **Identity.db** dla przechowywania informacji biznesowej związanej z produktami, koszykiem i zamówieniami

Schemat bazy **Sportstore.db** został pokazany na Rysunku 1. Baza **Identity.db** została utworzona automatycznie przez mechanizm ASP.NET Core Identity, dlatego umieszczanie jej schematu nie ma sensu.

Tabele `Orders` oraz `Products` nie wymagają wyjaśnień co do przetrzymywanych w nich informacji. Tabela `CartLine` przechowuje pojedynczą "linię zamówienia" z koszyka zakupów. Jedno zamówienie może zawierać wiele takich linii: Rysunek 2.

5.3.2 Autoryzacja użytkownika administracyjnego

Część administracyjna serwisu jest dostępna tylko po zalogowaniu. Do autoryzacji wykorzystano ASP.NET Core Identity, ograniczając dostęp do niektórych kontrolerów lub poszczególnych metod za pomocą atrybutu `[Authorize]`.

5.3.3 Schemat wyglądu serwisu

W celu ukształtowania spójnego wyglądu serwisu, wykorzystano poniższe mechanizmy dostępne w ASP.NET Core:

- Layouts - współdzielone schematy wyglądu (`Layout.cshtml` dla części klienckiej oraz `AdminLayout.cshtml` dla części administracyjnej
- ViewComponents - samodzielne komponenty niezależne od kontrolerów (np. `NavigationMenuViewComponent` odpowiadający za wyświetlanie menu wyboru kategorii).

5.3.4 Wyliczenie ilości stron w widoku przeglądania towarów

Aby prawidłowo wyliczyć liczbę stron pomiędzy którymi możliwa jest nawigacja w przeglądaniu listy towarów, zaimplementowano własny TagHelper. Jest to nowy mechanizm, który pojawił się dopiero w ASP.NET MVC 6 (czyli ASP.NET Core), umożliwiający tworzenie w C# kodu działającego po stronie HTMLa. Pozwala to na łatwe testowanie jednostkowe powstałego kodu oraz jego reużywalność.

5.3.5 Testy jednostkowe

Część kodu została przetestowana jednostkowo, wykorzystując technologie `xUnit` oraz `Moq`. Niektóre testy zgłaszają błędy i wymagają naprawy, jednakże celem było poznanie sposobu testowania jednostkowego aplikacji internetowej, sam wynik testów jest sprawą drugorzędną.

6 Opis działania aplikacji

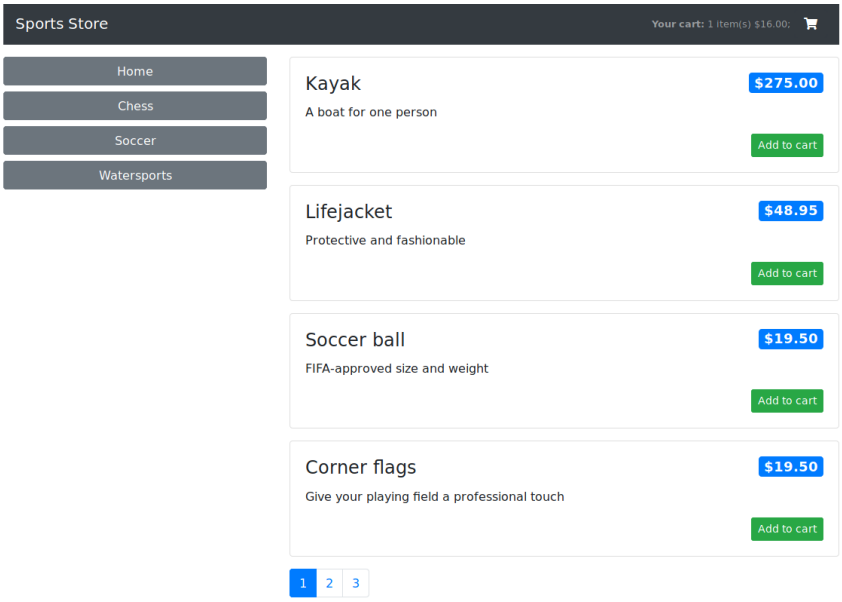
6.1 Część dostępna dla użytkownika

6.1.1 Przeglądanie zawartości sklepu

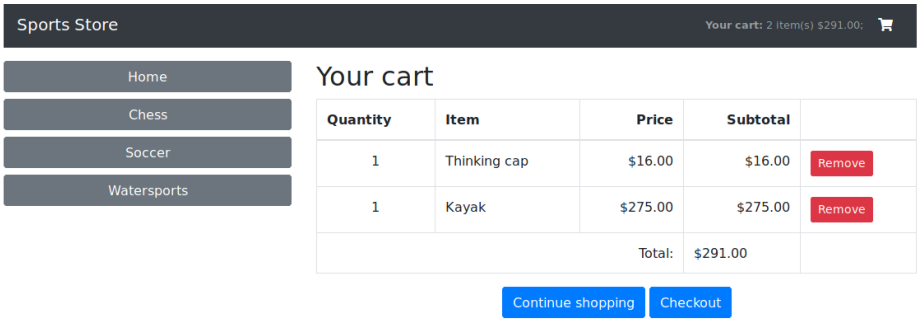
Przeglądanie zawartości sklepu odbywa się poprzez wybór odpowiedniej kategorii towarów z menu po lewej stronie. W przypadku wybrania `Home` pokazywane są wszystkie kategorie towarów. Z uwagi na szeroki asortyment sklepu oraz zwiększenie czytelności ograniczono liczbę wyświetlanych towarów do 4. Użytkownik ma możliwość przewijania kolejnych stron listy towarów, jeśli takie istnieją (Rysunek 3).

6.1.2 Koszyk zakupów

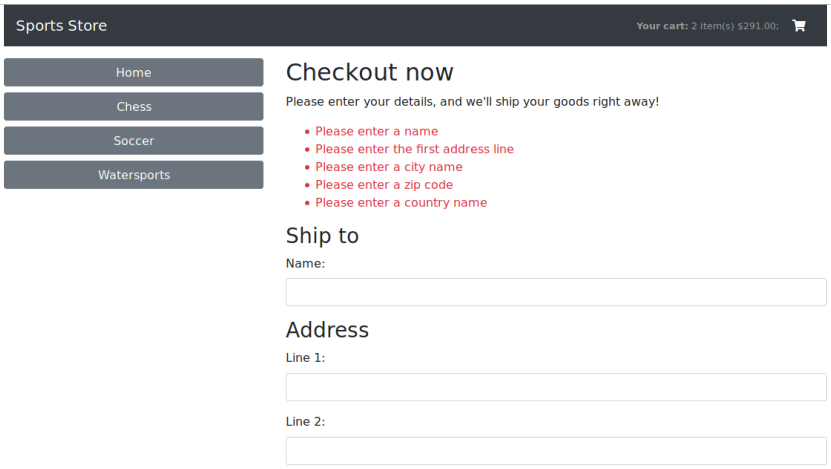
Po dodaniu towaru do koszyka, wyświetlana jest jego zawartość wraz z możliwością usunięcia towarów. Użytkownik ma możliwość kontynuowania zakupów lub przejścia do widoku złożenia zamówienia (Rysunek 4).



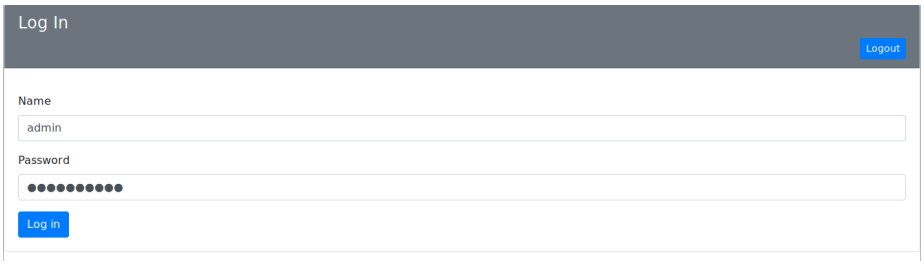
Rysunek 3: Widok przeglądania towarów



Rysunek 4: Koszyk zakupów



Rysunek 5: Widok składania zamówienia (skrócony)



Rysunek 6: Formularz logowania

6.1.3 Złożenie zamówienia

Składanie zamówienia polega na wypełnieniu danych adresowych oraz opcjonalnych dotyczących pakowania towaru. Formularz jest walidowany po stronie serwera w celu sprawdzenia wypełnienia koniecznych pól (Rysunek 5). Jeśli złożenie zamówienia zakończone jest powodzeniem, wyświetlane jest potwierdzenie przyjęcia zamówienia.

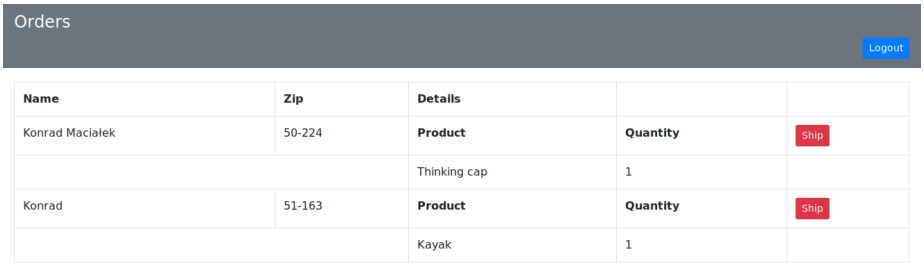
Zamówienia zapisywane są w bazie danych, do późniejszego administrowania wysyłką.

6.2 Część administracyjna

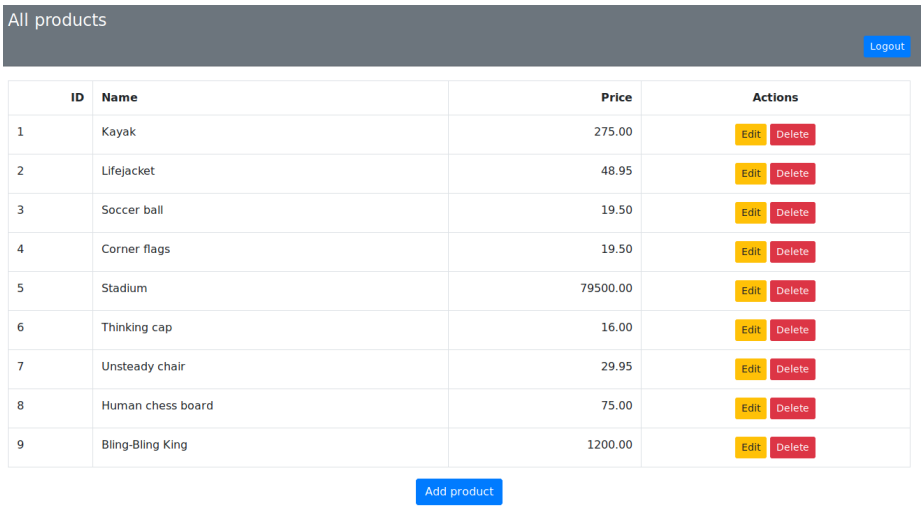
Część administracyjna serwisu jest dostępna tylko po ręcznym wpisaniu odpowiednich adresów.

6.2.1 Logowanie

Widok formularza logowania przedstawia Rysunek 6. Dane zalogowanego użytkownika przetrzymywane są w sesji.



Rysunek 7: Administracja zamówieniami



Rysunek 8: Administracja produktami

6.2.2 Administracja zamówieniami

Widok administracji zamówieniami (Rysunek 7) umożliwia przegląd aktualnie złożonych zamówień i oznaczenie ich jako wysłanych.

6.2.3 Administracja produktami

Widok administracji produktami zapewnia proste operacje CRUD na bazie produktów (Rysunek 8).

Literatura

[1] Adam Freeman, *Pro ASP.NET Core MVC*. Apress, 6th Edition, 2016.