Dokumentacja techniczna projektu sklep

Szymon Wachowiak

Projekt to prosty sklep internetowy. Projekt został stworzony w oparciu o platformę .NET 8.0 i wykorzystuje zestaw narzędzi dostępnych w ekosystemie .NET, takich jak Microsoft Identity do rejestracji użytkowników oraz SQLite wraz z Entity Framework do obsługi bazy danych.

1. Cel dokumentacji

Niniejsza dokumentacja techniczna ma na celu zapewnienie kompleksowego zrozumienia struktury, działania oraz konfiguracji projektu. Przeznaczona jest zarówno dla programistów pracujących nad projektem, jak i dla osób, które chciałyby zapoznać się z technicznymi aspektami działania aplikacji.

2. Zakres dokumentacji

Dokumentacja zawiera szczegółowe informacje na temat architektury projektu, struktury katalogów, wykorzystanych technologii, konfiguracji środowiska deweloperskiego oraz sposobu uruchomienia projektu na własnym komputerze. Omówione też zostały kluczowe funkcjonalności oraz sposoby ich implementacji.

3. Ogólny opis projektu

Projekt umożliwia użytkownikom przeglądanie i zakup przedmiotów w sklepie, rejestrację i logowanie użytkowników. Każdy zarejestrowany użytkownik jest w stanie sprawdzić listę swoich zamówień.

4. Technologie wykorzystane

w projekcie Projekt został zbudowany przy wykorzystaniu następujących technologii:

- .NET 8.0 z ASP.NET Core: Framework programistyczny wykorzystywany do budowy aplikacji internetowych w języku C#. Aplikacja wykorzystuje technologię Model-View-Controller (MVC).

- Microsoft Identity: Biblioteka umożliwiająca i ułatwiająca obsługę procesu uwierzytelniania i autoryzacji użytkowników.

- SQLite: Baza danych relacyjna wykorzystywana do przechowywania danych aplikacji,

- Entity Framework: Narzędzie Object-Relational Mapping (ORM) służące do mapowania obiektów aplikacji na struktury danych w bazie danych.

5. Najważniejsze pliki i katalogi występujące w projekcie

w projekcie Projekt został zbudowany przy wykorzystaniu następujących technologii:

- .NET 8.0 z ASP.NET Core: Framework programistyczny wykorzystywany do budowy aplikacji internetowych w języku C#. Aplikacja wykorzystuje technologię Model-View-Controller (MVC).

- Microsoft Identity: Biblioteka umożliwiająca i ułatwiająca obsługę procesu uwierzytelniania i autoryzacji użytkowników.

- SQLite: Baza danych relacyjna wykorzystywana do przechowywania danych aplikacji,

- Entity Framework: Narzędzie Object-Relational Mapping (ORM) służące do mapowania obiektów aplikacji na struktury danych w bazie danych.

- ProjektSklep.sln: Plik solucji Visual Studio.

- ProjektSklep.csproj: Główny plik projektu aplikacji.

- Program.cs: Zawiera kod źródłowy głównej klasy aplikacji, która definiuje metodę Main i konfiguruje uruchomienie aplikacji.

- appsettings*.json: Pliki konfiguracyjne aplikacji, które zawierają ustawienia dotyczące środowiska, takie jak połączenie z bazą danych.

- app.db: Plik bazy danych SQLite, w którym są przechowywane dane aplikacji.

- Controllers: Katalog zawierający kontrolery aplikacji,

- Data: Katalog zawierający pliki związane z dostępem do danych, takie jak klasy kontekstu bazy danych.

- Models: Katalog zawierający modele danych aplikacji,

- Migrations: Katalog zawierający pliki migracji dla bazy danych,generowane przez Entity Framework,

- Views: Katalog zawierający widoki aplikacji,

- Migrations: Katalog zawierający pliki migracji dla bazy danych.

6. Struktura Katalogów i Plików

1. **Controllers**
   - Pliki kontrolerów odpowiadające za logikę aplikacji oraz obsługę żądań użytkowników. Przykłady:
     - HomeController.cs: Obsługuje żądania dotyczące strony głównej.
     - OrdersController.cs: Obsługuje żądania dotyczące zamówień.
     - CartController.cs: Obsługuje żądania dotyczące koszyka zakupowego.
2. **Models**
   - Pliki modeli reprezentujące dane aplikacji oraz logikę biznesową. Przykłady:
     - Product.cs: Model reprezentujący produkt.
     - Order.cs: Model reprezentujący zamówienie.
     - Cart.cs: Model widoku reprezentujący dane w koszytku.
3. **Views**
   - Pliki widoków (szablonów) odpowiedzialne za generowanie interfejsu użytkownika. Widoki są podzielone na katalogi odpowiadające kontrolerom. Przykłady:
     - Home/Index.cshtml: Widok dla strony głównej.
     - Orders/Index.cshtml: Widok dla historii zamówień.
     - Cart/Index.cshtml: Widok dla koszyka zakupowego.
     - Cart/Success.cshtml: Widok poprawsnego złożenia zamówienia.
4. **Data**
   - Pliki związane z dostępem do danych oraz konfiguracją bazy danych. Przykłady:
     - ApplicationDbContext.cs: Klasa kontekstu bazy danych używana przez Entity Framework.
5. **Migrations**
   - Pliki migracji generowane przez Entity Framework, które pozwalają na zarządzanie schematem bazy danych.

7. Podział na Komponenty

- **Model (Models)** Modele reprezentują dane aplikacji oraz logikę biznesową. Przykładowe modele to Product, Order, Cart. Modele są używane przez Entity Framework do mapowania danych na bazę danych.
- **Widok (Views)** Widoki są odpowiedzialne za prezentację danych użytkownikowi. Są to pliki Razor (.cshtml), które generują HTML na podstawie danych przekazanych przez kontrolery. Widoki są zorganizowane w katalogach odpowiadających kontrolerom.
- **Kontroler (Controllers)** Kontrolery obsługują żądania HTTP, przetwarzają dane, wywołują odpowiednie metody w modelach i zwracają odpowiednie widoki. Przykładowe kontrolery to HomeController, OrdersController, CartController.

8. Opis Użytkowników Systemu

1. **Zarejestrowany Użytkownik**
    - **Opis:** Użytkownik, który utworzył konto i jest zalogowany w aplikacji.
    - **Uprawnienia:**
        - Może przeglądać produkty dostępne w sklepie.
        - Może dodawać produkty do koszyka i składać zamówienia.
        - Może przeglądać historię swoich zamówień.
        - Może modyfikować swój koszyk (dodawać, usuwać produkty, zmieniać ilość).
2. **Niezarejestrowany Użytkownik**
    - **Opis:** Użytkownik, który nie utworzył konta ani nie jest zalogowany.
    - **Uprawnienia:**
        - Może przeglądać produkty dostępne w sklepie.
        - Nie może dodawać produktów do koszyka ani składać zamówień.
        - Nie ma dostępu do historii zamówień ani innych funkcji dostępnych dla zarejestrowanych użytkowników.

9. Uruchomienie projektu

Wymagania:

- .NET 8.0 SDK: Wymagane do kompilacji jak i uruchomienia projektu.

- Visual Studio: do uruchomienia projektu

Instrukcja:

1.Sklonuj repozytorium przez stronę na GitHubie lub przez komendę git clone

2.Uruchom projekt w visual studio

Gdyby wyskoczył błąd kliknij na górze kontynuuj, a na strone apply migrations i odśwież

Po uruchomieniu powinna być widoczna strona główna projektu.