

Dokumentacja Techniczna – Strona internetowa z listą na zakupy

Bartosz Tomczak, Mateusz Strycharz

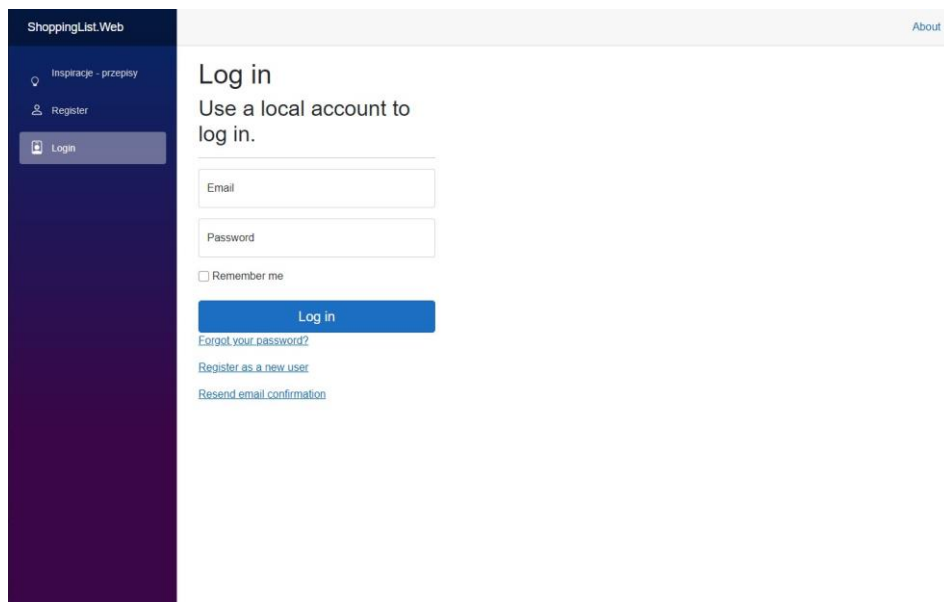
link do strony: <https://shoppinglist-blazor8.azurewebsites.net/>

1.Opis strony:

Strona internetowa jest aplikacją internetową, która umożliwia użytkownikom tworzenie oraz zarządzanie list zakupów. Strona posiada system formularzy, który działa jako punkty na liście. Dane wprowadzane przez użytkownika są przechowywane w bazie danych, do konkretnego konta.

2.System logowania:

Na stronie głównej znajduje się panel logowania.



ShoppingList.Web

About

Inspiracje - przepisy

Register

Login

Log in

Use a local account to log in.

Email

Password

☐ Remember me

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Przy logowaniu użytkownik ma możliwość zapamiętania danych, przywrócenia hasła, przekierowanie do rejestracji oraz ponowne wysłanie weryfikacji e-mailem.

ShoppingList.Web [About](#)

Inspiracje - przepisy

Register

Login

Register

Create a new account.

Email

Password

Confirm Password

[Register](#)

Zarejestrowani użytkownicy mogą się logować na stronę, używając poprawnych danych. Gdy użytkownik jest zalogowany na stronie wyświetla się ich lista zakupów.

Użytkownicy mają również możliwość wylogowania się z konta na stronie.

ShoppingList.Web [About](#)

Inspiracje - przepisy

Moje listy zakupów

Moje ulubione przepisy

[Zarządzaj kontem](#)

Logout

Manage your account

Change your account settings

[Profile](#)

Profile

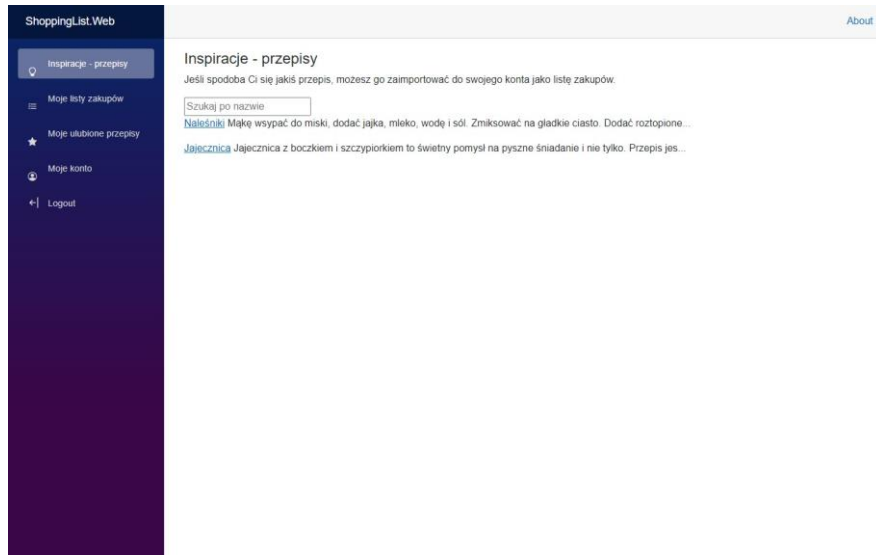
Email

Password

Username

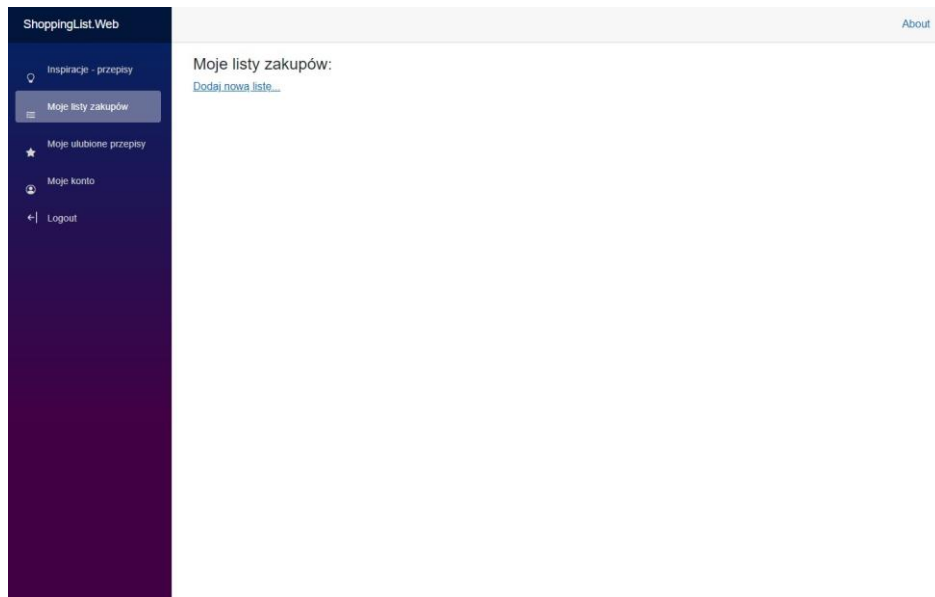
3.Użytkowanie:

Aplikacja posiada podstronę dla listów publicznych, w której znajdują się listy zakupów na różnego rodzaju przepisy.



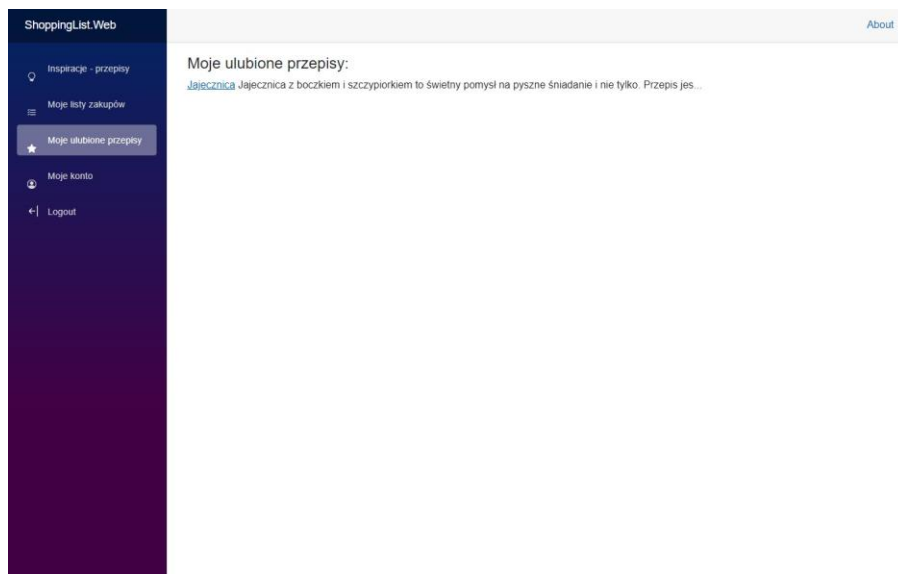
Nie można zmieniać danych tych listów, lecz można je dodać do swojej własnej listy zakupów z ustawioną przez użytkownika nazwą. Można również wyszukać listy za pomocą nazwy. Przy kliknięciu nazwy listy, wyświetli się cały przepis oraz opcję stworzenia listy ze składnikami oraz możliwość dodania strony do ulubionych.

Lista zakupów użytkownika jest podstroną, która jest dostępna jedynie dla użytkownika.

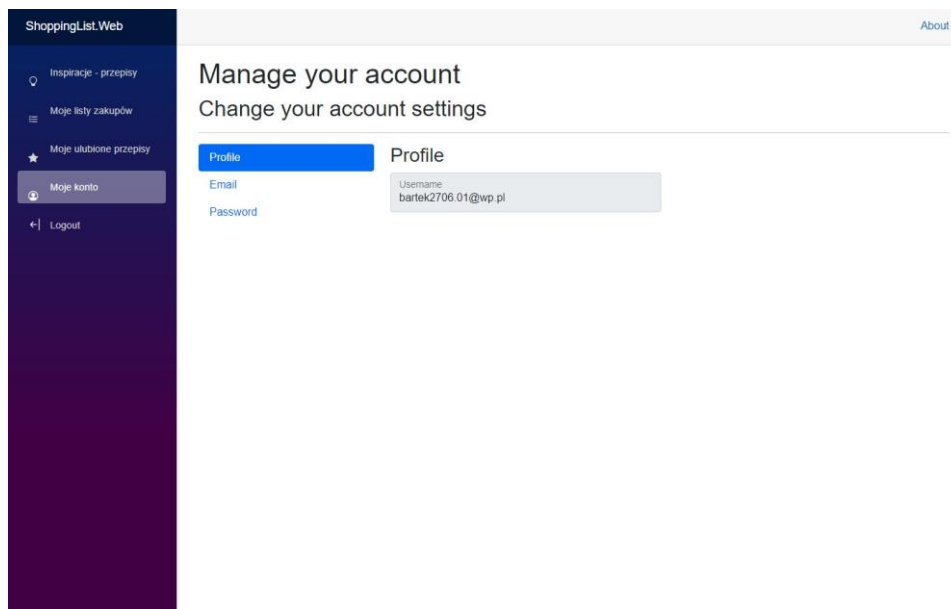


Przy naciśnięciu “Dodaj nową listę”, pojawi się miejsce na nazwę listy. Po zapisaniu nazwy, wyświetli się pierwszy punkt naszej listy, w którym można dodać nazwę produktu, ilość produktu z jednostką miary oraz opcjonalny opis. Gdy produkt zostanie zapisany do listy, pojawia się następny wiersz listy oraz check-box przy poprzednim produkcie, który opisuje czy produkt jest już kupiony. Strona posiada również możliwość usunięcia produktu z listy i usunięcie całej listy zakupów.

Strona posiada również zakładkę z ulubionymi przepisami.



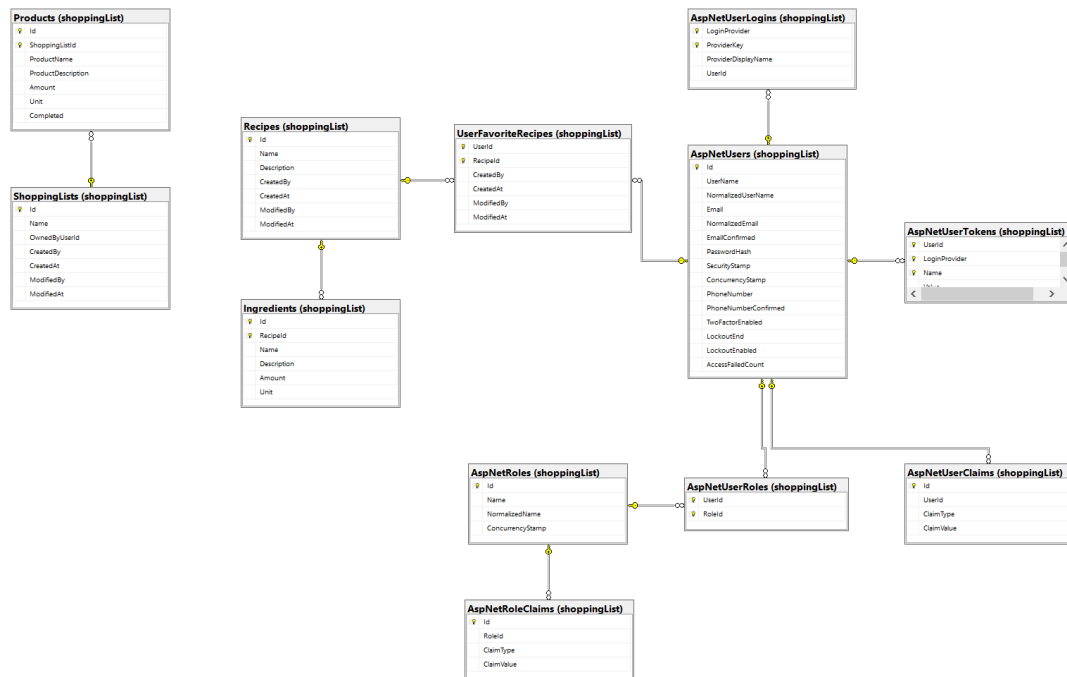
Ostatnią podstroną na stronie jest konto.



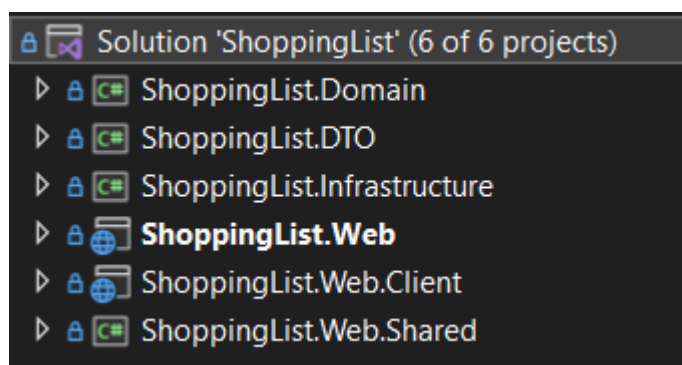
Na tej podstronie mamy możliwość zmiany e-maila konta oraz zmiany hasła.

4.Baza danych:

Aplikacja posiada związaną z nią bazę danych, w której przechowywane są wiadomości o koncie użytkownika, listach zakupów użytkownika oraz dane listów publicznych. Strona umożliwia wstawianie nowych danych listów zakupów użytkownika oraz kont, które są z nimi powiązane.



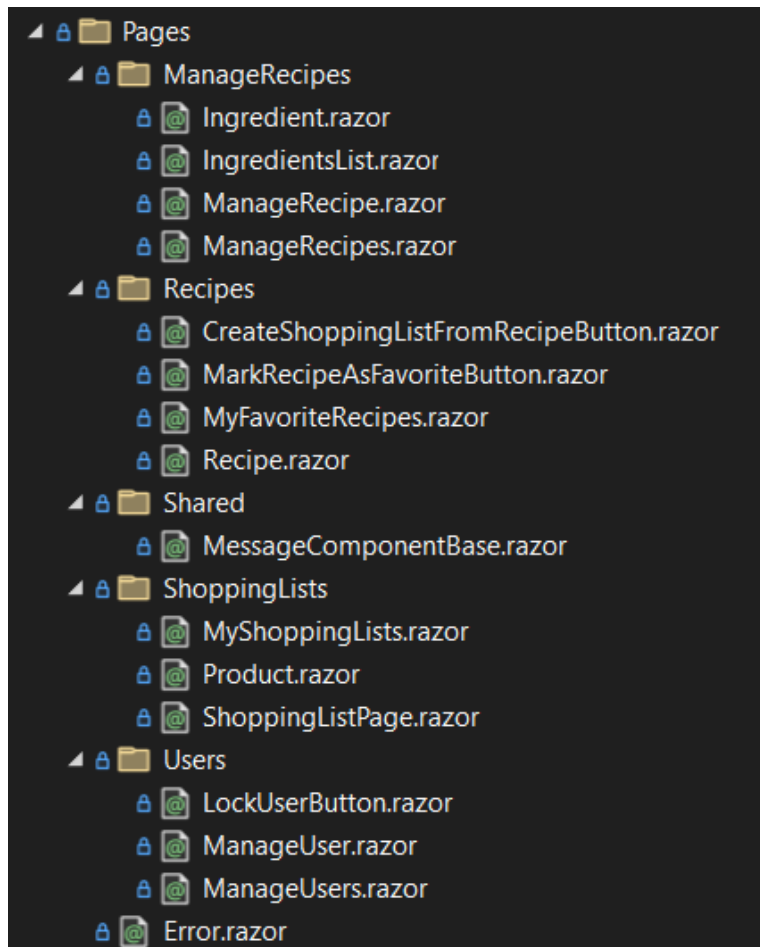
5. Struktura projektu



Solucja składa się z powyższych projektów. Punktem startowym aplikacji jest ShoppingList.Web. Działanie aplikacji opiera się o wzorzec CQRS (bazując na paczce MediatR).

Krótkie omówienie struktury:

- Domain - domena aplikacji, zawiera encje trzymane w bazie danych
- DTO - modele, commands oraz queries
- Infrastructure - implementacje commands i queries, połączenie z bazą danych, pliki konfiguracyjne encji
- Web - strony renderowane po stronie serwera
- Web.Client - strony renderowane po stronie klienta
- Shared - pliki współdzielone między projektami Web

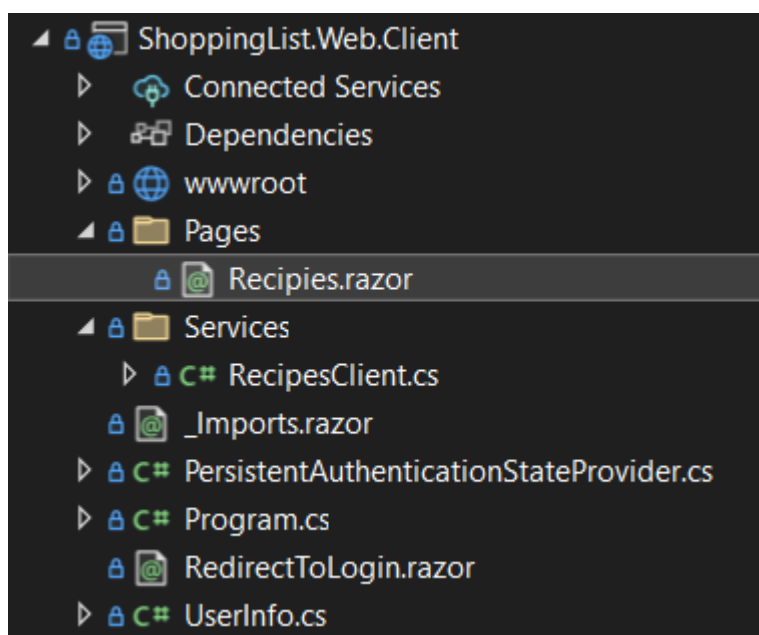


Powyższe pliki zawierają najważniejsze podstrony aplikacji. W zależności od sytuacji użyte są różne sposoby renderowania stron:

- Static Server (Static SSR) - kiedy nie podano żadnego atrybutu w pliku, oraz nie dziedziczy po komponencie `MessageComponentBase`. Sposób ten zwraca gotowy do wyświetlenia HTML bezpośrednio w przeglądarce.
- Interactive Server - przeglądarka otrzymuje gotowy HTML,

jednak wciąż jest możliwa interakcja użytkownika bezpośrednio w aplikacji, bez konieczności ciągłego przesyłania nowych widoków z serwera. Komunikacja w tym przypadku przebiega za pomocą WebSocketów.

- Static Server, wraz z atrybutem StreamRendering - podobny do trybu w punkcie 1. z tą różnią, że przeglądarka otrzymuje wstępnie wygenerowany plik HTML, a następnie przesyłany jest kolejny plik po pobraniu potrzebnych informacji z bazy danych.



Projekt zawierający stronę generowaną po stronie klienta posiada jeden tryb renderowania:

- InteractiveWebAssembly - cały widok jest generowany po stronie przeglądarki, a ponadto dalsze akcje również są

zarządzane przez przeglądarkę. Komunikacja z serwerem występuje za pomocą requestów HTTP do kontrolera API po stronie serwera.

Z racji, że aplikacja Web.Client renderowana jest po stronie klienta, to strona Recipes.razor nie korzysta bezpośrednio z komend CQRS, a jedynie z serwisu, który wykonuje zapytania HTTP do serwera.

```
5
6 1 reference | Mateusz Strycharz, 3 days ago | 1 author, 1 change
7 public class RecipesClient(HttpClient httpClient) : IRecipesClient
8 {
9     2 references | Mateusz Strycharz, 3 days ago | 1 author, 1 change
10    public async Task<IReadOnlyCollection<RecipeShort>> GetRecipes(string? searchName)
11    {
12        var recipes = await httpClient.GetAsync("recipes");
13        return await recipes.Content.ReadFromJsonAsync<IReadOnlyCollection<RecipeShort>>() ?? [];
14    }
15 }
```