**Programowanie Aplikacji Webowych**

Dokumentacja projektu

Bartłomiej Mazurkiewicz,

173670, 3EF-DI/AA, L02

Spis treści

[Backend 3](#_Toc188217994)

[User model 3](#_Toc188217995)

[Announcement model 6](#_Toc188217996)

[Item model 7](#_Toc188217997)

[ShoppingCart model 9](#_Toc188217998)

[User Controller 12](#_Toc188217999)

[Register 12](#_Toc188218000)

[Login 13](#_Toc188218001)

[Metody GetMe, GetById, Delete 14](#_Toc188218002)

[UserService 15](#_Toc188218003)

[ShoppingCart Controller 16](#_Toc188218004)

[CreateCart, AddItem, GetCart 16](#_Toc188218005)

[UpdateItem 17](#_Toc188218006)

[ClearCart 18](#_Toc188218007)

[Announcement Controller 18](#_Toc188218008)

[Create announcement 18](#_Toc188218009)

[GetMyAnnouncements, GetAnnouncements 19](#_Toc188218010)

[Update, Delete 20](#_Toc188218011)

[DbContext 21](#_Toc188218012)

[Frontend 22](#_Toc188218013)

# Założenia projektu

## Wstęp

Projekt sklepu płyt winylowych opiera się na stworzeniu API w języku C# na platformie .NET z wykorzystaniem Entity Frameworks Core oraz frontendu we frameworku React, a także Postgresql bazę danych na dockerze. Stworzone modele w języku C# zawierają takie klasy jak User, Announcement oraz ShoppingCart.

## Funkcjonalności na backendzie:

[Post] Logowanie, Rejestracja, Dodaj ogłoszenie, Dodaj do koszyka

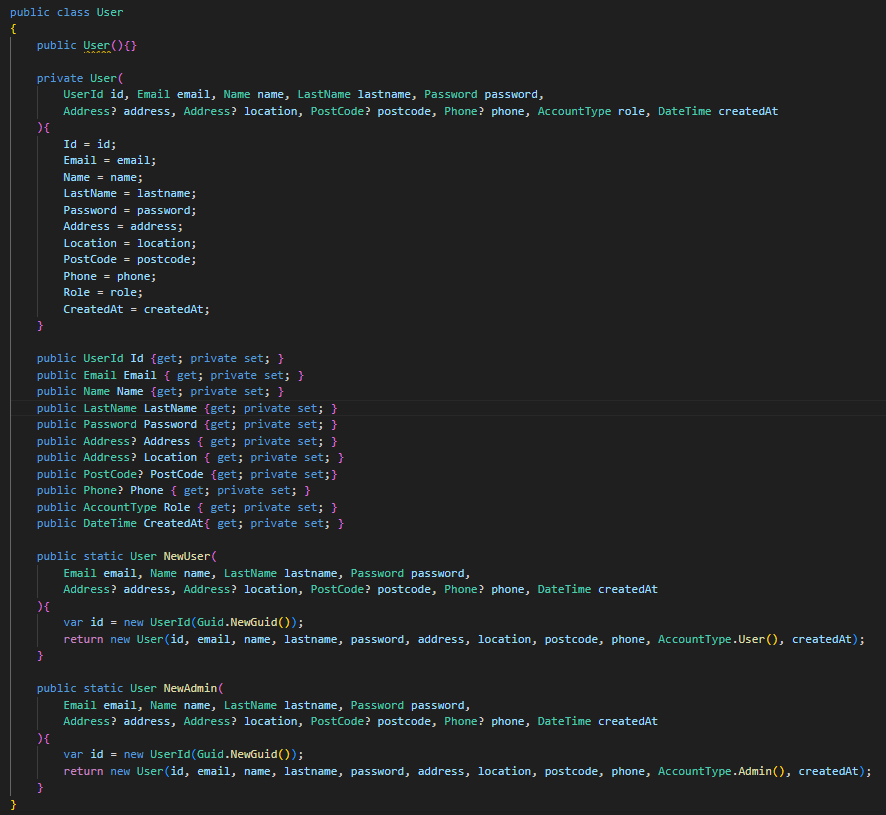
[Get] Wyświetl użytkownika, Wyświetl informacje o moim koncie, Wyświetl ogłoszenie, wyświetl moje ogłoszenia, Wyświetl wszystkie ogłoszenia,  Wyświetl moje zamówienia

[Put] Aktualizuj ogłoszenie (tylko admin lub osoba tworząca ogłoszenie), Aktualizuj przedmiot w koszyku

[Delete] Usuń ogłoszenie (tylko admin lub osoba tworząca ogłoszenie), Wyczyść koszyk

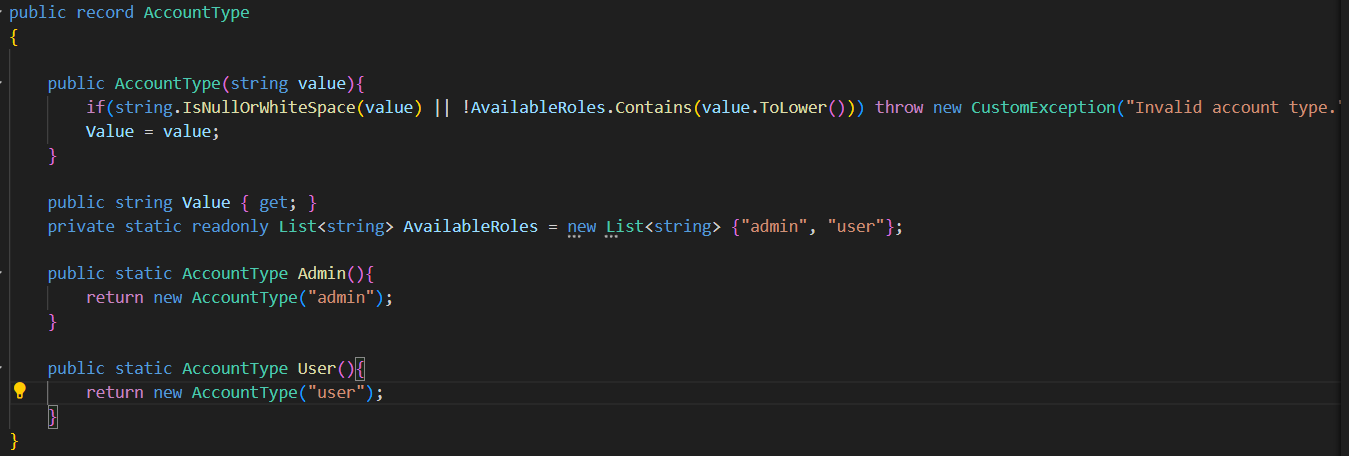
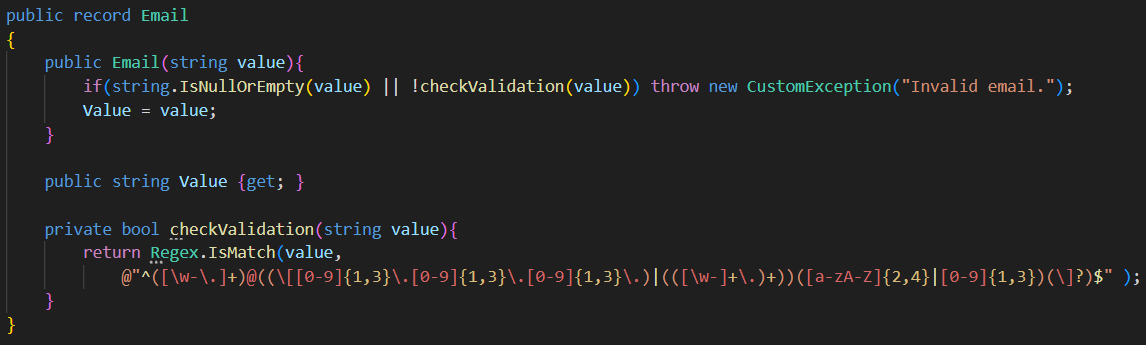
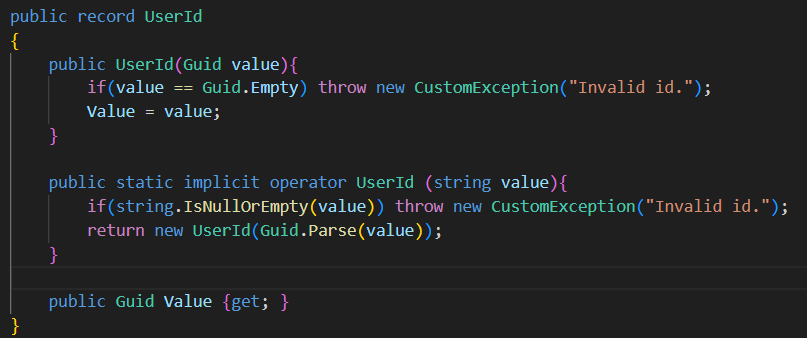
# Backend

## User model



Model użytkownika zawiera takie wartości jak Id, Email, Name, LastName, Password, Role, CreatedAt, które są wymagane przy tworzeniu obiektu. Występują także wartości opcjonalne, takie jak Address, Locaion, PostCode oraz Phone.

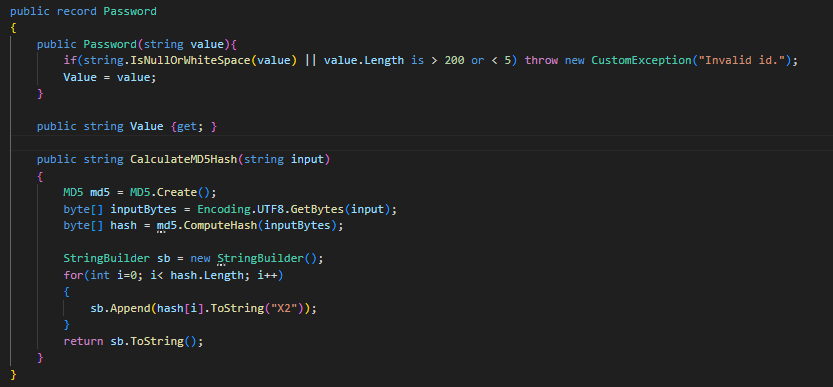
Każda z tych wartości ma swój własny zmiennej, przykładowo zmienna Id jest typu UserId, Email typu Email, Name typu Name, LastName typu LastName, Password typu Password, Address oraz Location typu Address, Phone typu Phone, Role typu AccountType.

Każdy typ umieszczony jest w osobnym pliku, gdzie występuj walidacja lub sprawdzenie, czy wartość przypisywana nie jest pusta. 

W przypadku typu AccountType przypisanie nowej roli użytkownikowi następuje w prosty sposób, jakim jest podanie nazwy rekordu a następnie nazwy roli, przykładowo AnnountType.User(); Rolę można dodać jedynie taką, jaka znajduje się na liście. Obecnie znajdują się na niej dwie pozycje: user oraz admin.

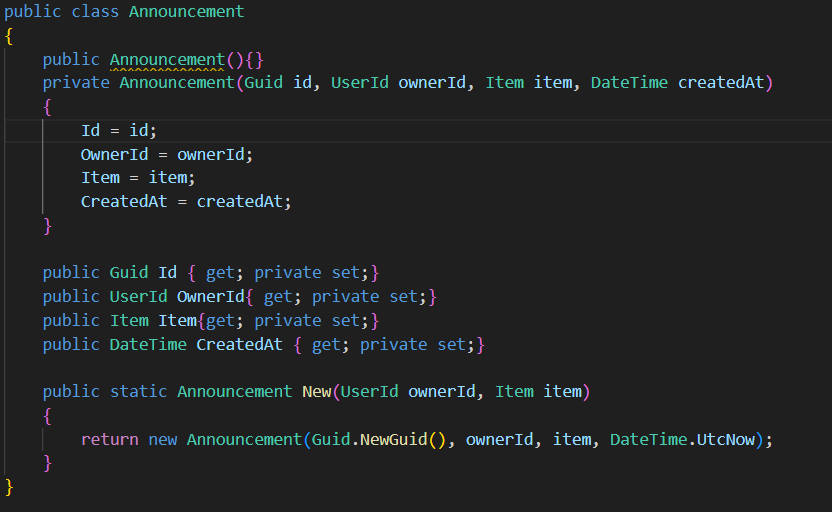
W klasie User występuje konstruktor służący stworzeniu nowego obiektu użytkownika przypisując odpowiednie zmienne.

Zaimplementowane zostały także dwie metody służące do tworzenia użytkownika z odpowiednią rolą. Metoda NewUser() używana jest w kontrolerze do tworzenia nowego użytkownika ze zwykłymi uprawnieniami, natomiast metoda NewAdmin() jest wykorzystywana przy tworzeniu projektu w celu automatycznego stworzenia konta Administratora.



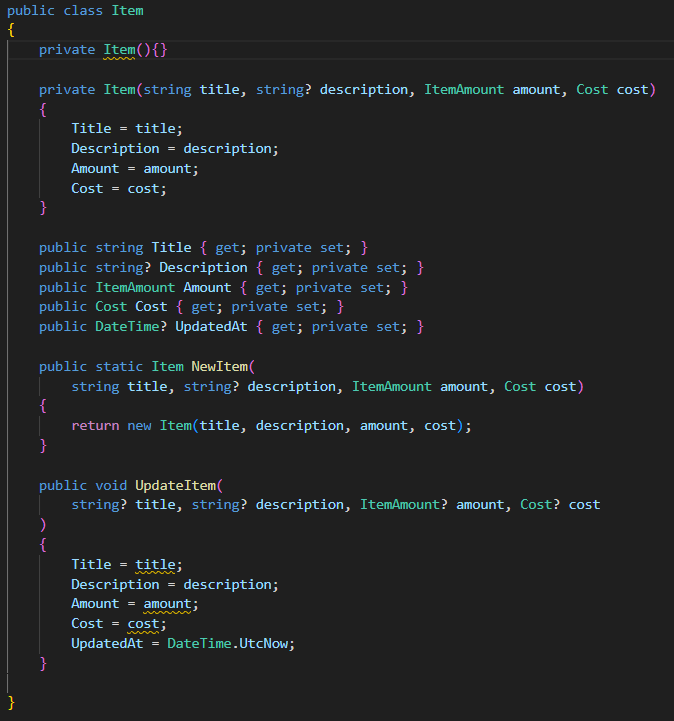
Rekord Password zawiera metodę CalculateMD5Hash służącą do hashowania haseł w celu bezpieczeństwa.

## Announcement model

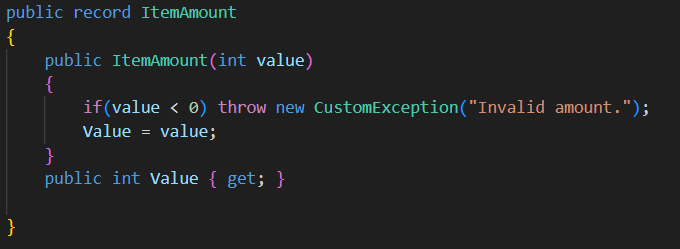


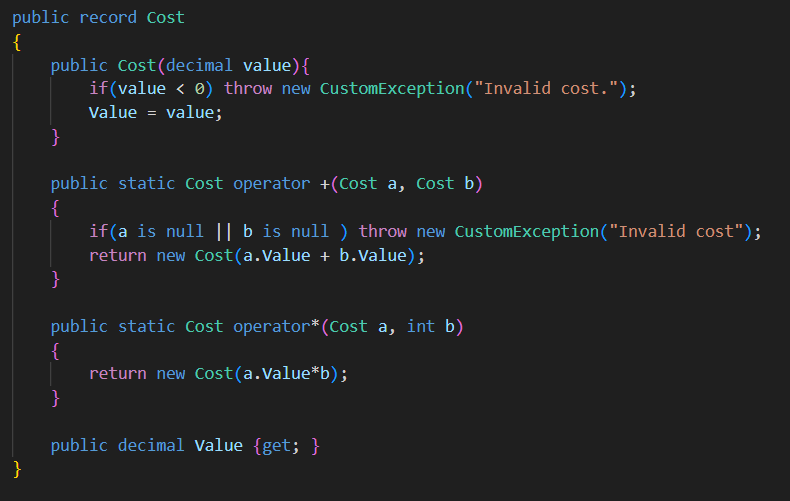
Model ogłoszenia zawiera wartości takie jak Id, OwnerId typu UserId, obiekt Item, CreatedAt. W celu stworzenia obiektu wykorzystywany jest konstruktor przypisujący odpowiednie pola. Klasa zawiera metodę New() wykorzystywaną do tworzenia nowego ogłoszenia.

## Item model



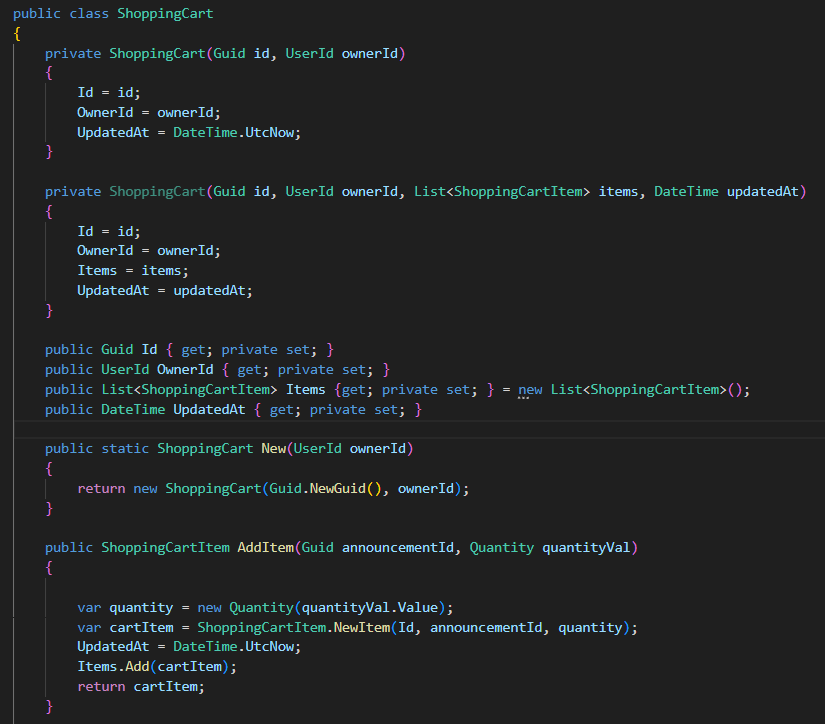
Obiekt Item jest wykorzystywany w klasie Announcement. Zwawiera pola Title, Description, Amount typu ItemAmount, Cost typu Cost oraz UpdatedAt. Zawarta jest tutaj metoda NewItem() tworząca nowy przedmiot oraz metoda UpdatedItem wykorzystywana do aktualizowania przedmiotu.

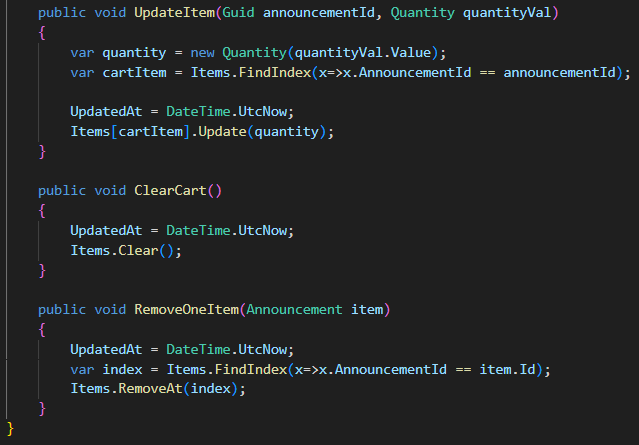




Zarówno w rekordzie ItemAmount jak i Cost jest sprawdzane, czy wartości nie są ustawione poniżej 0. W rekordzie Cost zaimplementowane zostały operatory dodawania i mnożenia, aby można było dodawać lub mnożyc dwa obiekty typu Cost.

## ShoppingCart model





Model koszyka zawiera pola Idm OwnerId typu UserId, listę Items typu ShoppingCartITem oraz UpdatedAt.

Zaimplementowana metoda New() służy do stworzenia nowego koszyka dla użytkownika.

AddItem to metoda dodająca do koszyka nowy przedmiot.

UpdateItem służy do aktualizowania przedmiotów w koszyku.

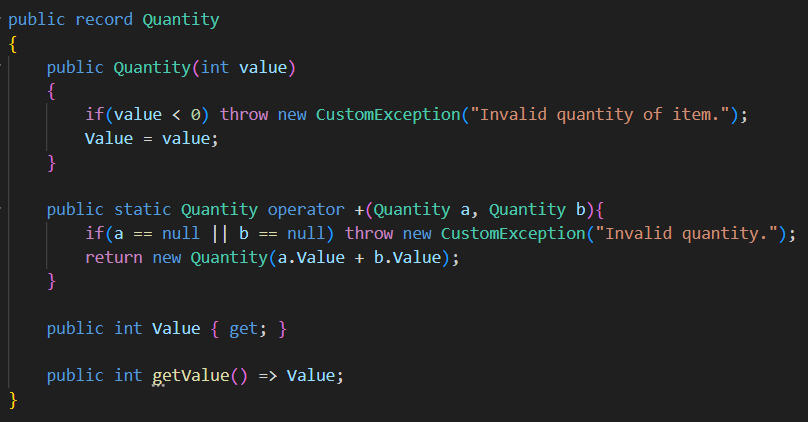
ClearCart służy do wyczyszczenia koszyka ze wszystkich przedmiotów.

RemoveOneItem usuwa wybrany przedmiot z koszyka.



ShoppingCartItem to klasa zawierająca Id, ShoppingCartId, AnnouncementId oraz Quantity typu Quantity. Znajduje się tutaj metoda NewItem() do tworzenia nowego przedmiotu dla koszyka.

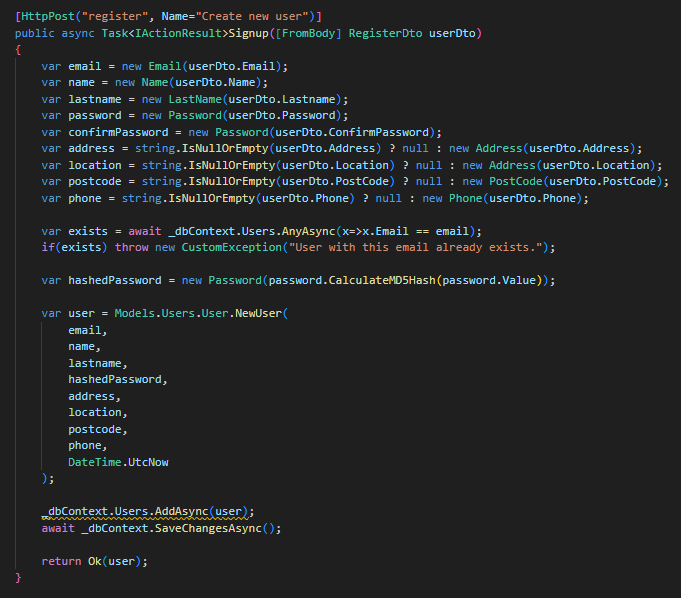
Metoda update aktualizuje ilość wybranego przedmiotu.



Quantity to rekord zawierający operator dodawania oraz metodę do zwracania wartości nazwaną getValue().

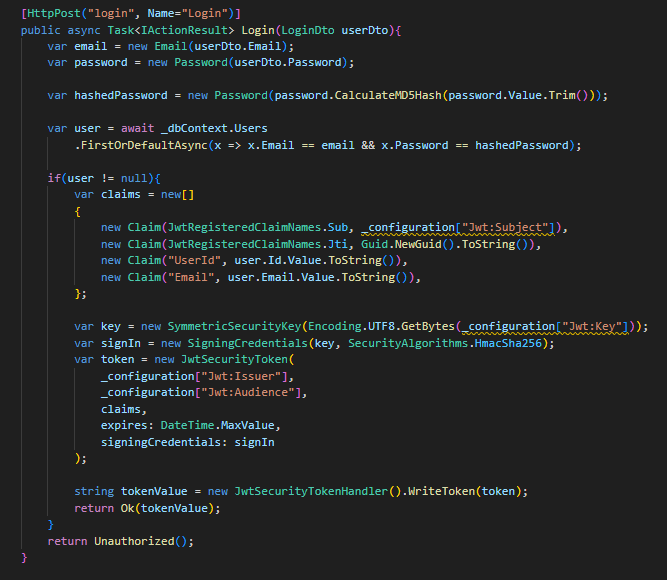
## User Controller

### Register



Customowa metoda Register służy do rejestracji nowego użytkownika. Wykorzystywane jest DTO w celu pobrania wartości od użytkownika i przypisania ich do odpowiednich zmiennych. Hasło przed stworzeniem nowego obiektu jest hashowane, a dopiero następnie przesyłane do metody torzącej użytkownika.

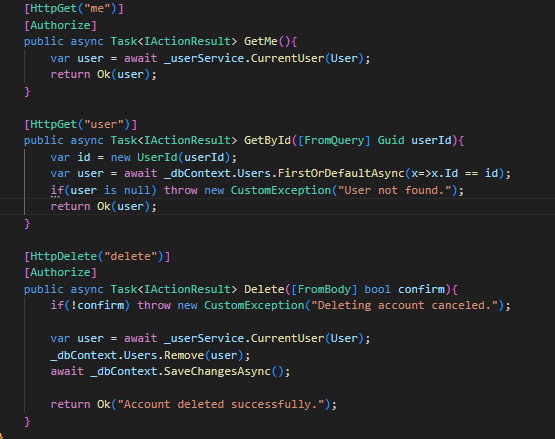
### Login



Własnoręcznie zaimplementowana metoda służąca do logowania pobiera użytkownika z bazy danych na podstawie emailu oraz porównaniu zahashowanych haseł. W przypadku braku rekordu w bazie danych zwracany jest kod Unauthorized.

Jeśli obiekt istnieje w bazie, tworzony jest token posiadający wartość UserId oraz Email. Na potrzeby projektu token nigdy nie wygasa. Po udanym logowaniu token jest zwracany i gotowy do pobrania.

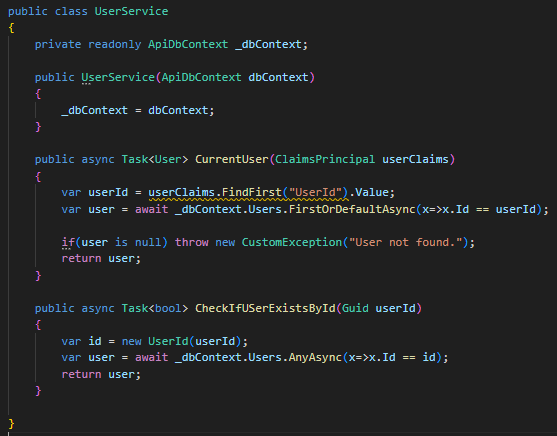
### Metody GetMe, GetById, Delete



Metody GetMe() oraz Delete() posługują się repozytorium UserService zawierające informacje na temat zalogowanego użytkownika. GetMe() to metoda służąca do wyświetlenia informacji o obecnie zalogowanym użytkowniku. Metoda Delete() usuwa obecnie zalogowaneg użytkownika. W obu przypadkach wymagana jest wcześniejsza autoryzacja za pomocą tokenu.

Metoda GetById() pozwala na pobranie informacji użytkownika, którego Id zostanie podane.

### UserService

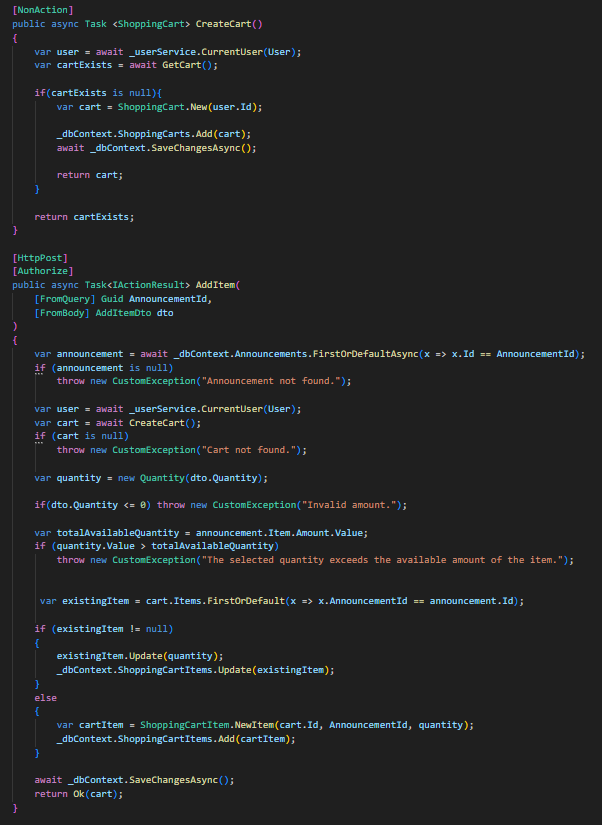


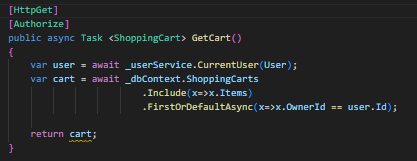
UserService posiada metodę CurrentUser, która zwraca obiekt użytkownia pobranego z bazy danych na podstawie użytego tokenu. Odczytywany jest z niego Id, a następnie pobiera obiekt User z bazy danych i zwraca go.

Metoda CheckIfUSerExistsById() sprawdza, czy dany użytkownik istnieje na podstawie podanego Id.

## ShoppingCart Controller

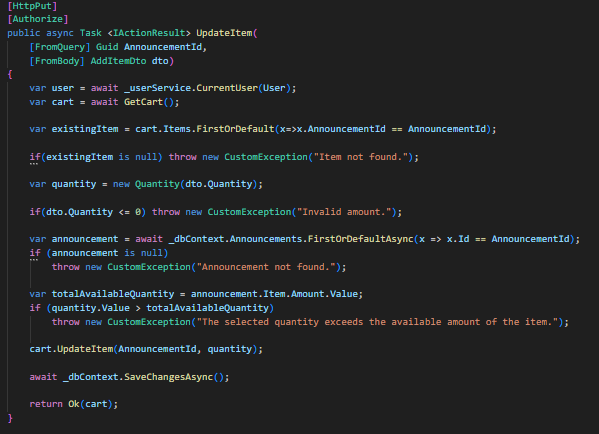
### CreateCart, AddItem, GetCart





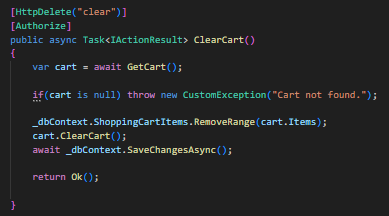
Kontroller posiada metodę AddItem, gdzie dodaje przedmiot do koszyka użytkownika. Jeśli koszyk nie istnieje, wykorzystuje metodę CreateCart do stworzenia go, natomiast jeśli koszyk istnieje, to wykorzystuję tę motodę do zwrócenia obiektu cart wywołując metodę GetCart(). Jeśli przy dodwaniu przedmiotu dany przedmiot już istnieje w koszyku, wykorzystywana jest metoda do aktualizacji ilości przedmiotu za pomocą metody Update().

### UpdateItem



Metoda UpdateItem() służy do aktualizacji ilości przedmiotu w koszyku. Sprawdzany jest tutaj warunek, czy podana wartość nie przekracza ogólnej ilości dostępnego towaru.

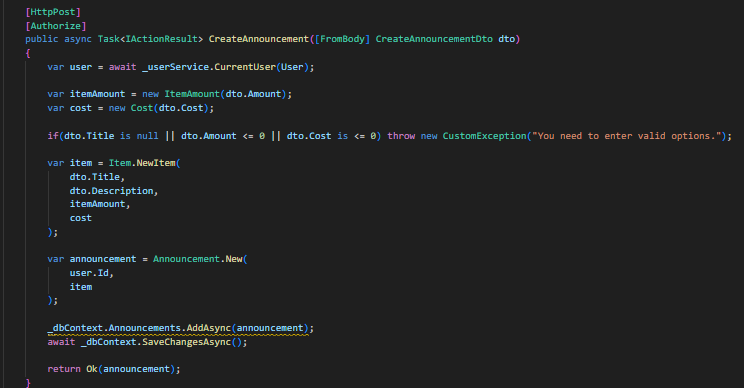
### ClearCart



Metoda ClearCart słuzy do wyczyszczenia zawartości koszyka. Użyta zostaje metoda ClearCart znajdująca się w modelu koszyka.

## Announcement Controller

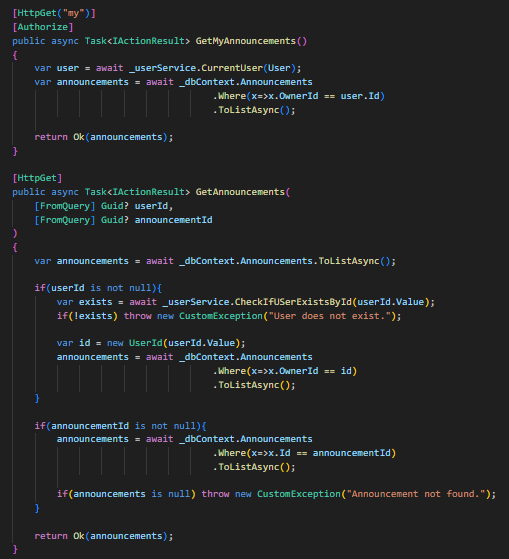
### Create announcement



Create Announcement służy do stworzenia nowego ogłoszenia, gdzie użytkownik podaje tytuł, opis, koszt oraz ilość przedmiotów.

Tworzony jest tutaj obiekt przedmiotu, a następnie ogłoszenie wraz z nowym obiektem item.

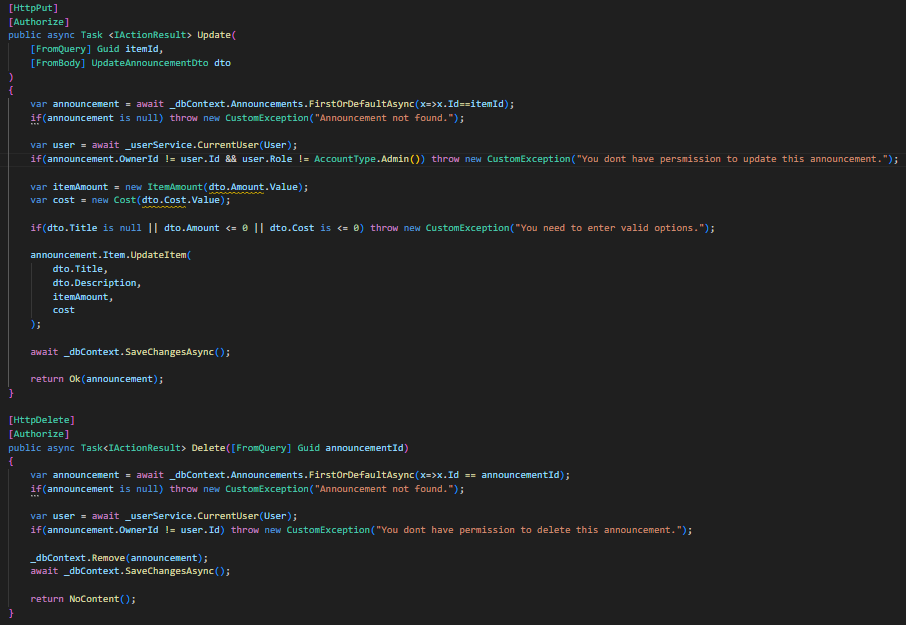
### GetMyAnnouncements, GetAnnouncements



Pierwsza metoda pobiera dla zalogowanego użytkownika jego własne stworzone ogłoszenia i wyświetla informacje o nich.

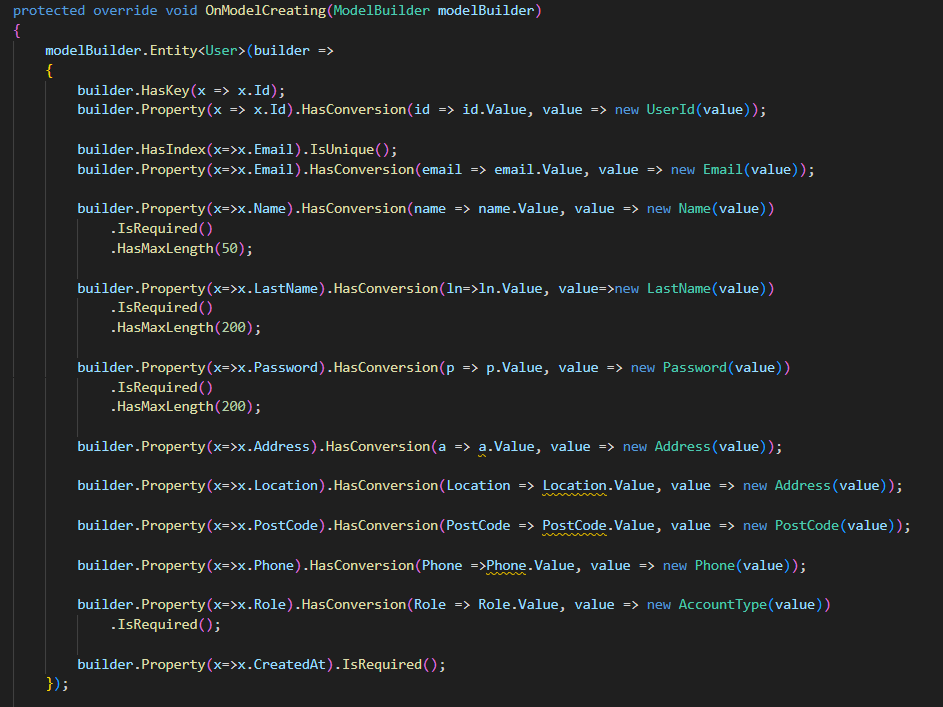
Druga metoda sprawdza, czy zostało podane id userId lub announcementId. Jeśli userId nie jest puste, to wyświetla ogłoszenia dla podanego użytkownika. Jeśli announcementI nie jest puste, to wyświetla ogłoszenie na podstawie podanego Id, natomiast jeśli obie wartości są puste, to wyświetlone zostają wszystkie istniejące ogłoszenia.

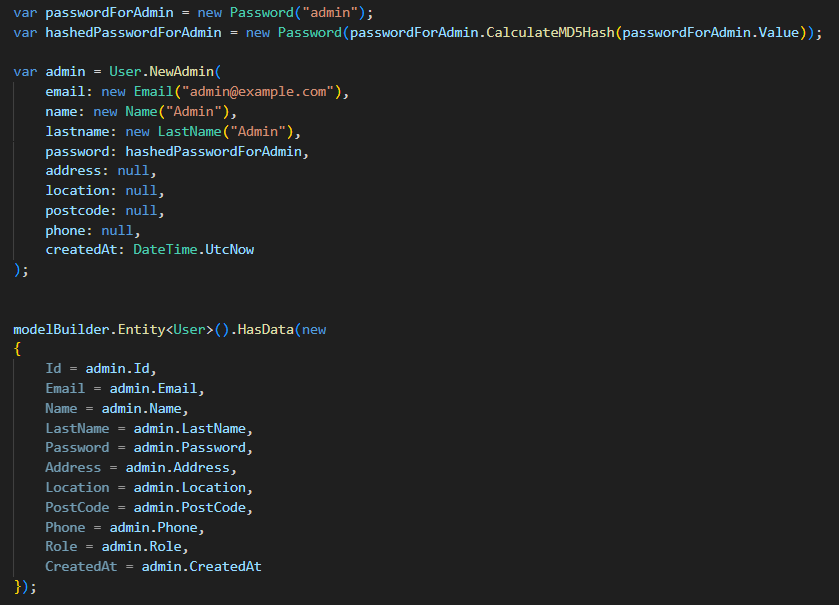
### Update, Delete



Metody służą do aktualizacji ogłoszenia oraz jest usuwania. Aktualizacji może dokonać jedynie użytkownik, który stworzyć ogłoszenie lub administrator.

## DbContext

  
OnModelCreating zawiera model tworzenia obiektów. W tym przypadku model User wymaga konwersji własnych typów na typy proste. Wykorzystywana jest w tym celu metoda HasConversion, gdzie przykładowo wartość typowego guid id zamieniona zstaje na UserId.



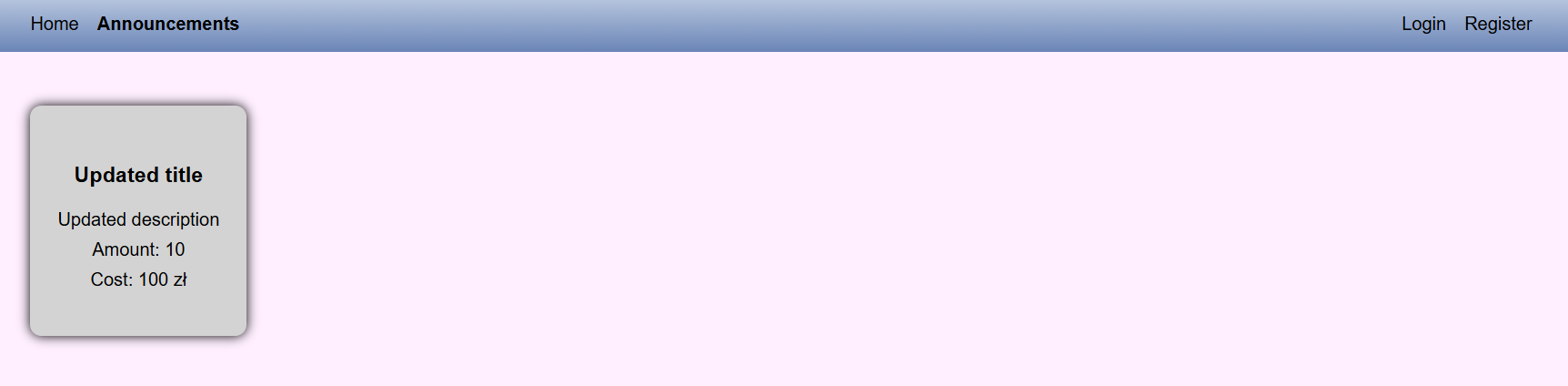
Występuje tutaj także implementacja tworzenia konta administratora przy tworzeniu projektu z emailem admin@example.com oraz zahashowanym hasłem admin.

# Frontend

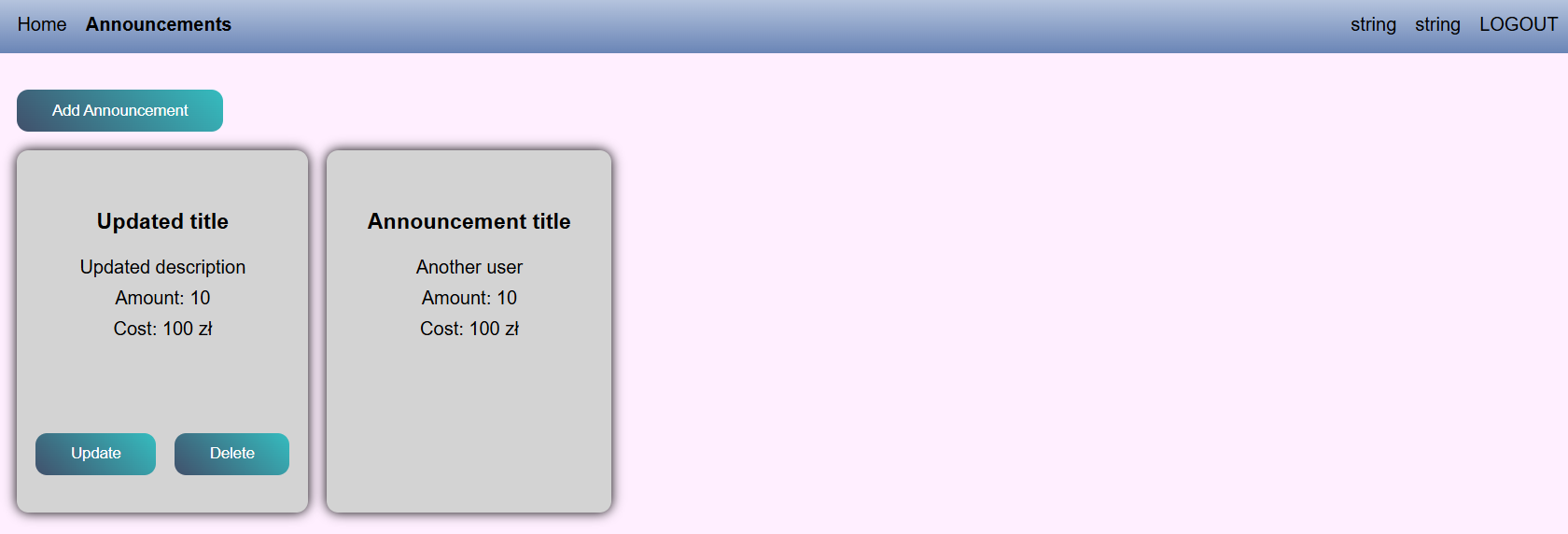


Po wejściu na stronę pojawia się komunikat „Log in to see your profile”.

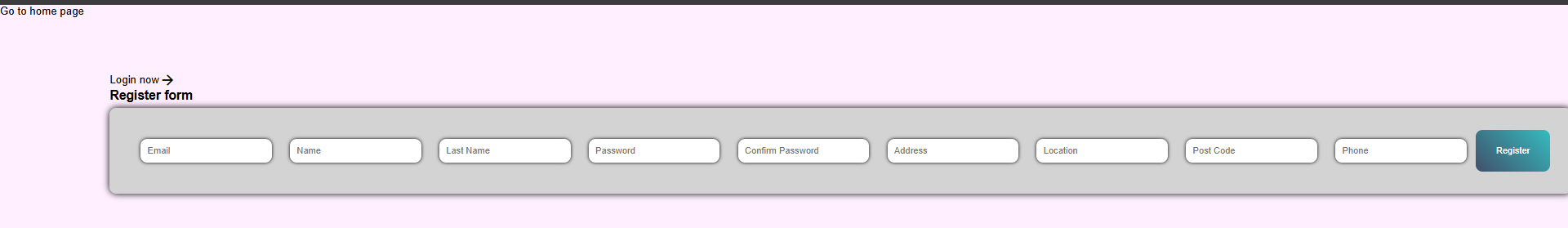
Z tego miejsca można wyświetlić ogłoszenia w zakładce Announcements.



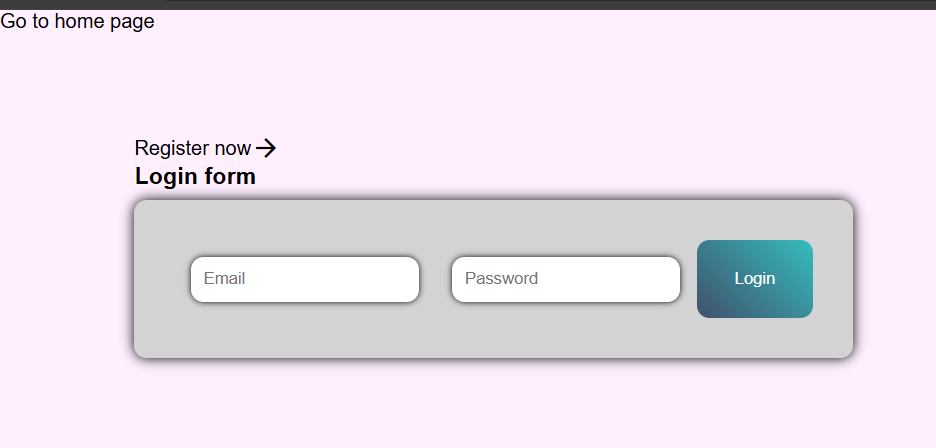
Natomiast po zalogowaniu wygląd się nieco zmienia. Dochodzi przycisk Add Announcement, a jeśli użytkownik jest właścicielem ogłoszenia, pojawiają się dwa przyciski służące do usunięcia lub zaktualizowania ogłoszenia.



Możliwe jest zarejestrowanie nowego konta w formularzu rejestracji

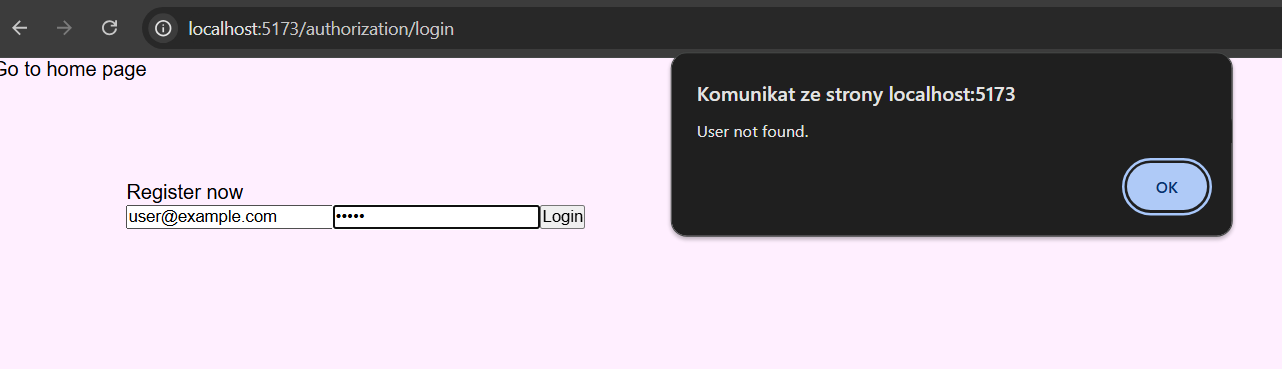


Można także przejść do panelu logowania klikając „Login now”

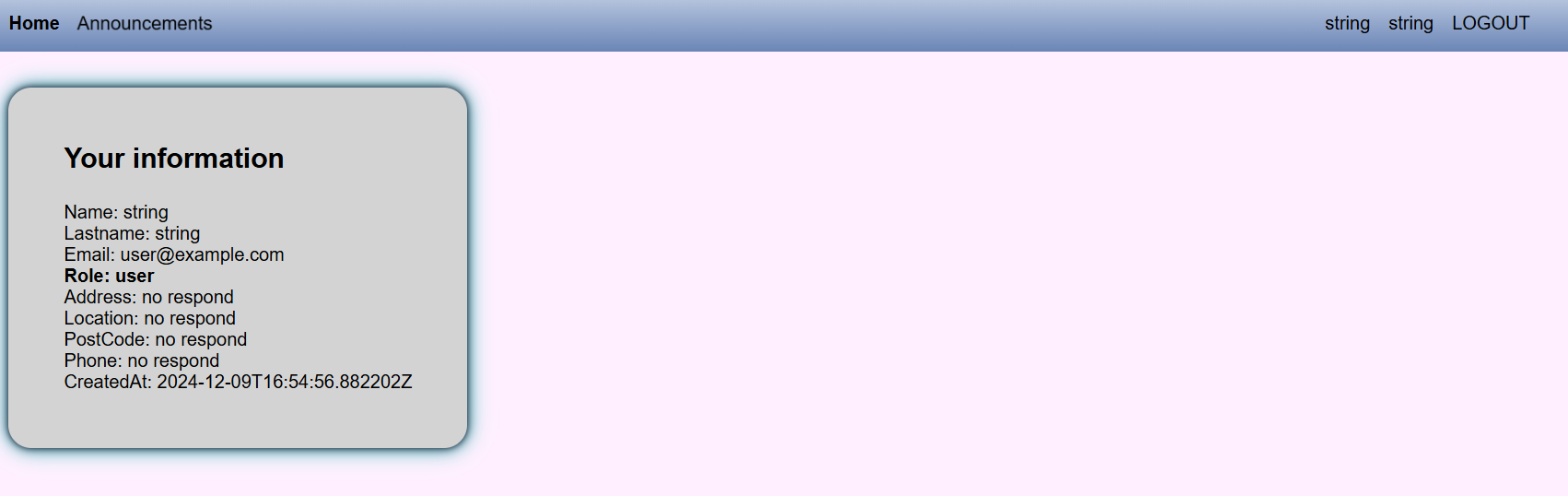


Klikając „register now” wrócimy do panelu rejestracji.

Po nieudanym logowaniu wyświetlony zostaje komunikat o błędzie.



Po udanym logowaniu wyświetlona zostają informacje o koncie na stronie głównej, a w nawigacji wyświetlane jest imię oraz nazwisko, a także opcja wylogowania.



Użytkownik został stworzony poprzez dodanie kontekstu UserContext, dlatego wszystkie metody są zawarte w jednym miejscu i mogą zostać użyte poprzez destrukturyzację.







Metody są zatem łatwo dostępne i proste w konfiguracji.