Національний університет “Острозька академія”

Звіт про виконання лабораторних робіт

з дисципліни

Створення Web за

допомогою WCF

Виконав:

студент групи КН-31

Марчук Микола

Острог 2023

Зміст

[Лабораторна робота №1](#_1fob9te) **1**

[Завдання:](#_3znysh7) 1

[Діаграма:](#_2et92p0) 1

[Авторизація](#_yz0nxzdh16vf) 9

[Висновки:](#_vquvfeb11yvj) 13

[**Лабораторна робота №2**](#_lfqlzac0m2hr) **14**

[Завдання:](#_y1yajedz2da) 14

[Завдання 1](#_9hm74tlxw8yg) 15

[Завдання 2](#_qop6jo4jbd1s) 16

[Висновки:](#_hl11be6xaafz) 20

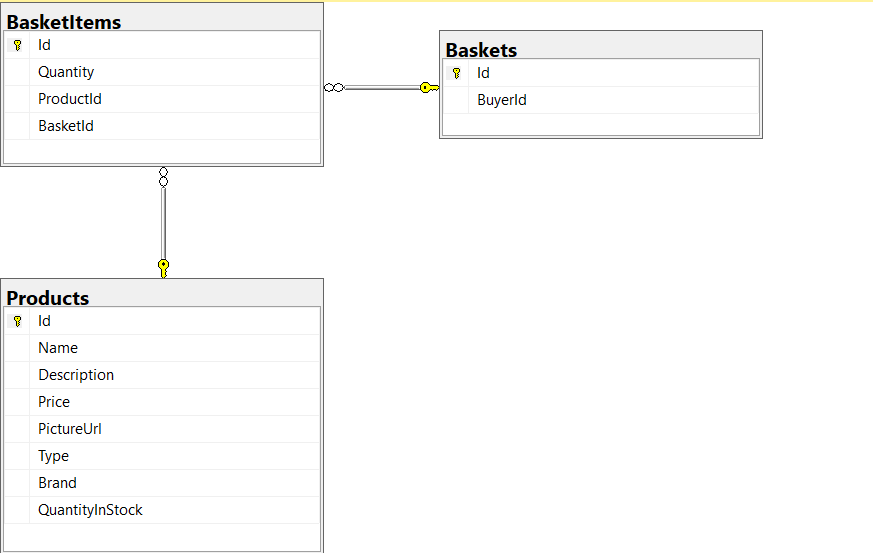
# 

# Лабораторна робота №1

## Завдання:

На основі теми(Ecommerce) з курсу “Програмування на C#” створити WebApi (або одразу WebApi+Blazor WASM) з тестами на swagger.

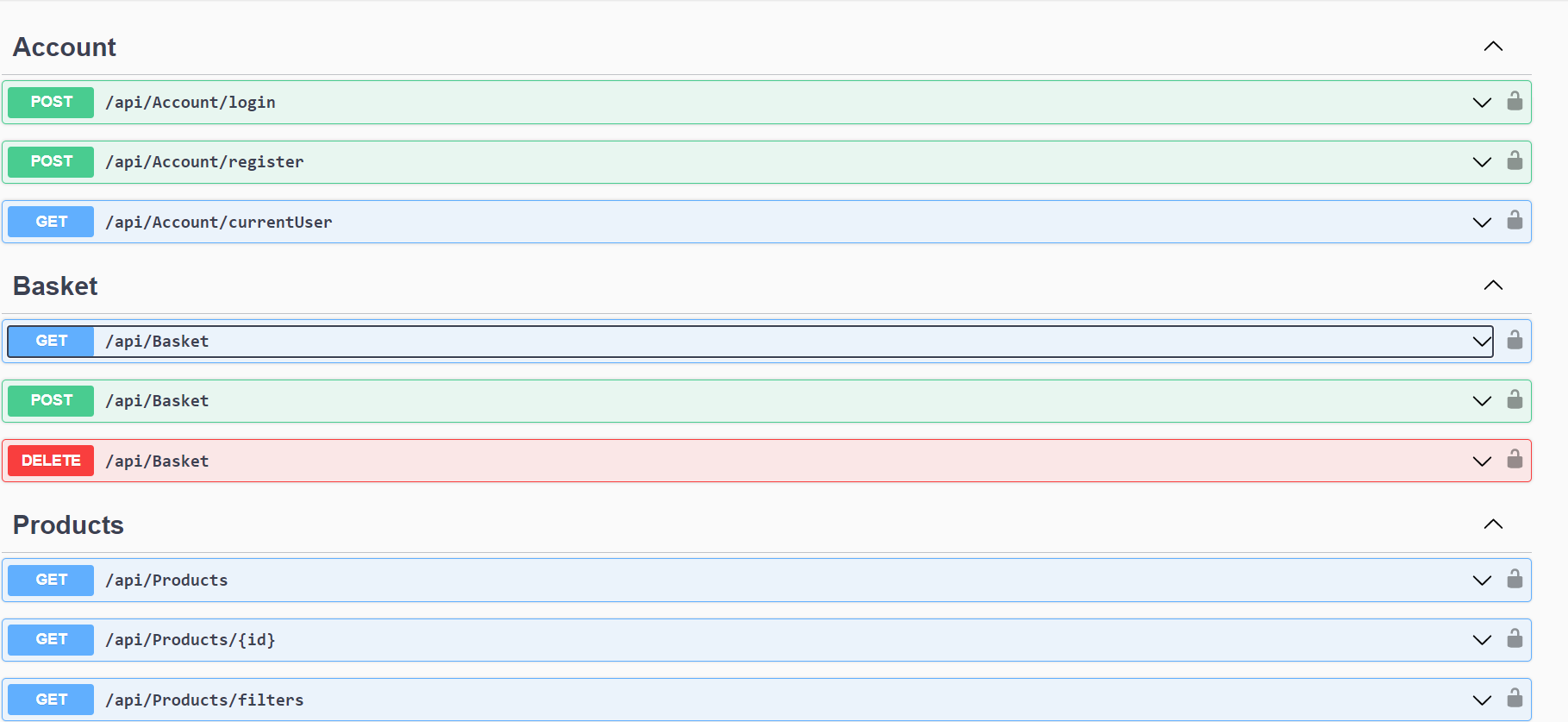
### Діаграма:



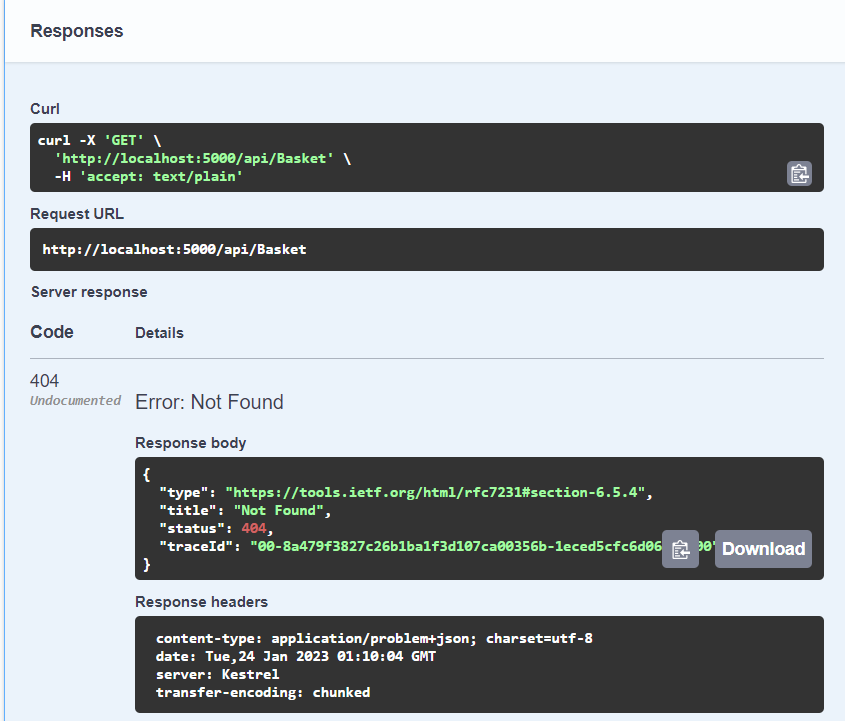
наразі у мене є три таблиці створені мною - продукти, корзина, та речі в корзині. Відношення між baskets та basketItems - one-many -> у нас може бути одна корзина та багато речей у ній. basketItems та products має відношення one-one -> одній речі в корзині відповідає один продукт.

Також у мене є сутність User, яка походить від бібліотеки Asp Net Identity, але про неї згодом.

Запустивши проект через dotnet run watch ми бачимо наступну картину

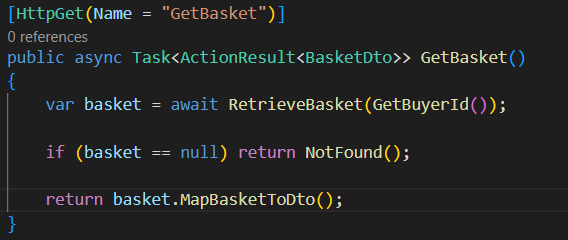


У нас є три контролери - для аутентифікації, для управління корзиною та продуктами

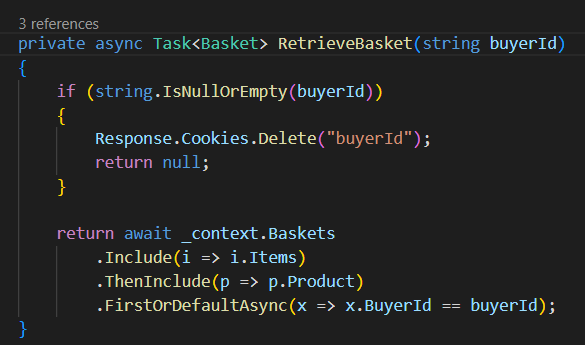


Якщо ми просто попробуємо забрати корзину, у нас нічого не вийде, бо ми не прив’язані ні до чого. Звідки нам знати, що саме цей користувач саме на цьому комп’ютері має таку корзину? Корзина створюється, коли ми додаємо якийсь продукт, а після того ми генеруємо унікальну куку і задаємо її на комп’ютері юзера. Чому саме кука?

1. Вона може зберігатись до 30-ти днів.
2. Для того, щоб наш умовний бекенд отримав цю куку, достатньо використати лише один метод на .net



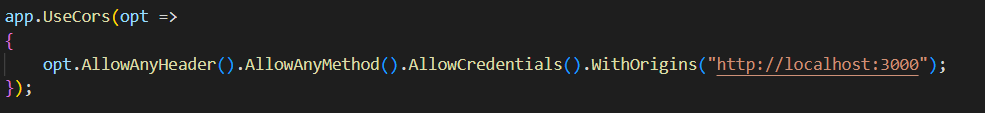
DTO - Data transfer objects - ми задаємо вигляд того, як хочемо повертати наші дані з запиту.



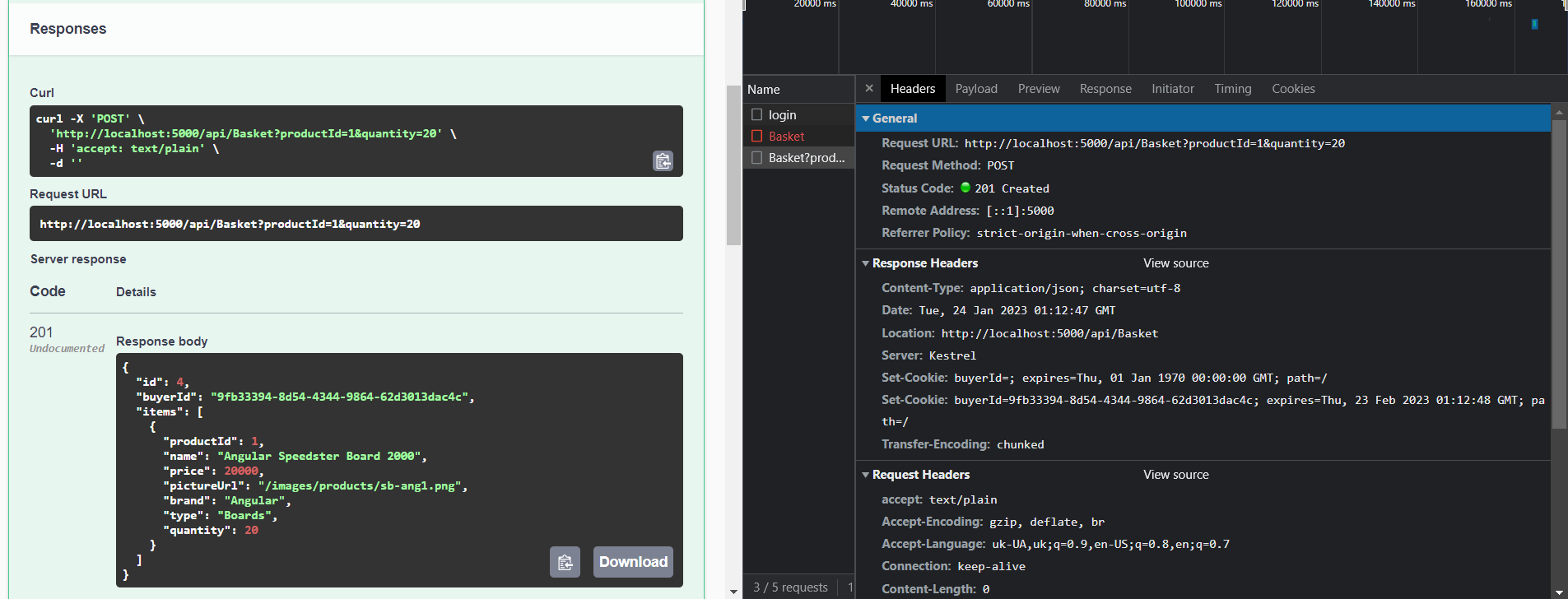
## 



Важливо - для того, щоб коректно працювала така логіка, потрібно додати деякі налаштування для CORS



А саме AllowCredentials(), а також подібна логіка має бути на фронті. Swagger таке підтримує з коробки, а наприклад для реакту потрібні подібні налаштування до allowCredentials



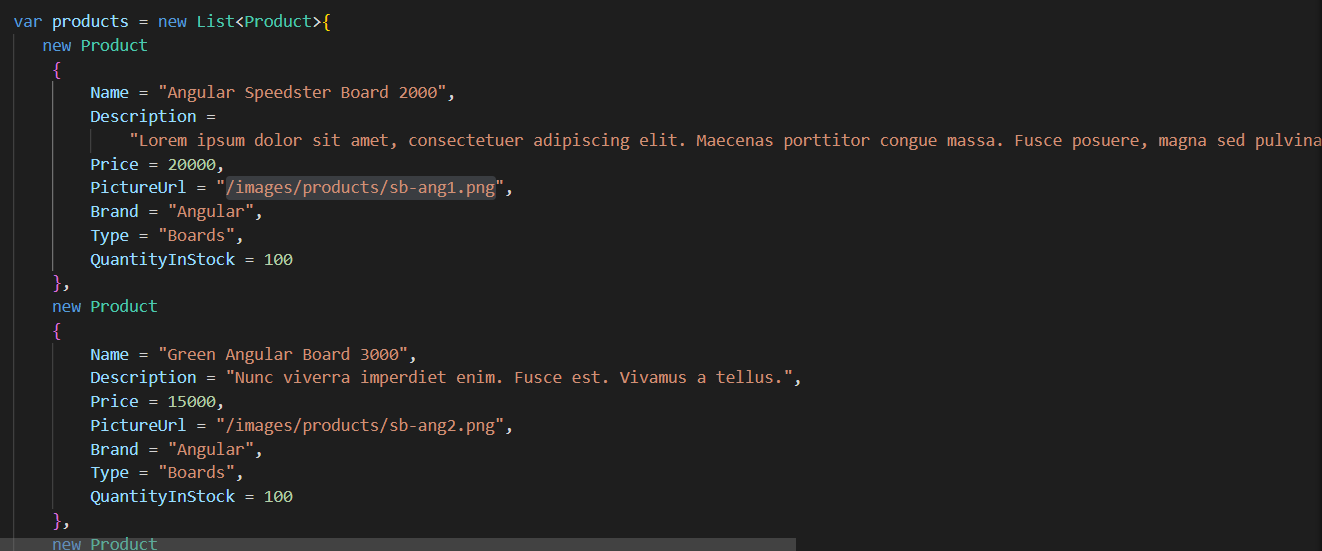
На фото можна побачити в response headers хедер Set-Cookie: ось таким чином через запит ми додаємо куку з бека на фронт

## 

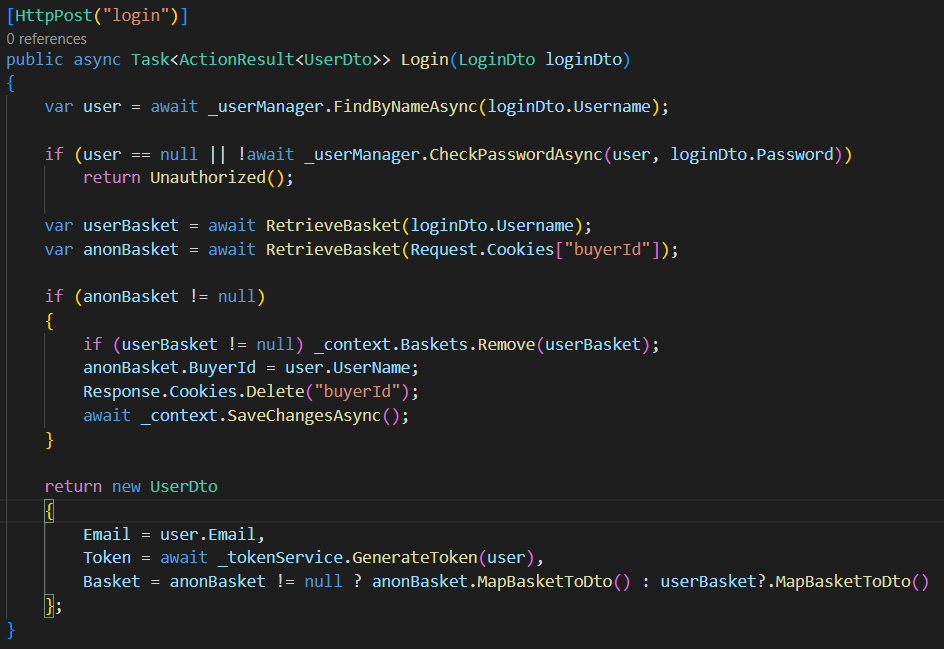
Логіка додавання речі в корзину -

Намагаємось дістати куку з buyerId, якщо її немає, генеруємо новий basket з унікальним buyerId. Потім намагаємось дістати з бази продукт. Якщо він існує, додаємо його в таблицю basket і зберігаємо зміни.

Продукти наразі задаються вручну, є список при ініціалізації бази даних

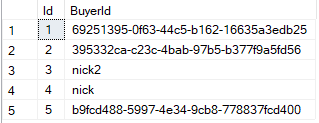


### Авторизація

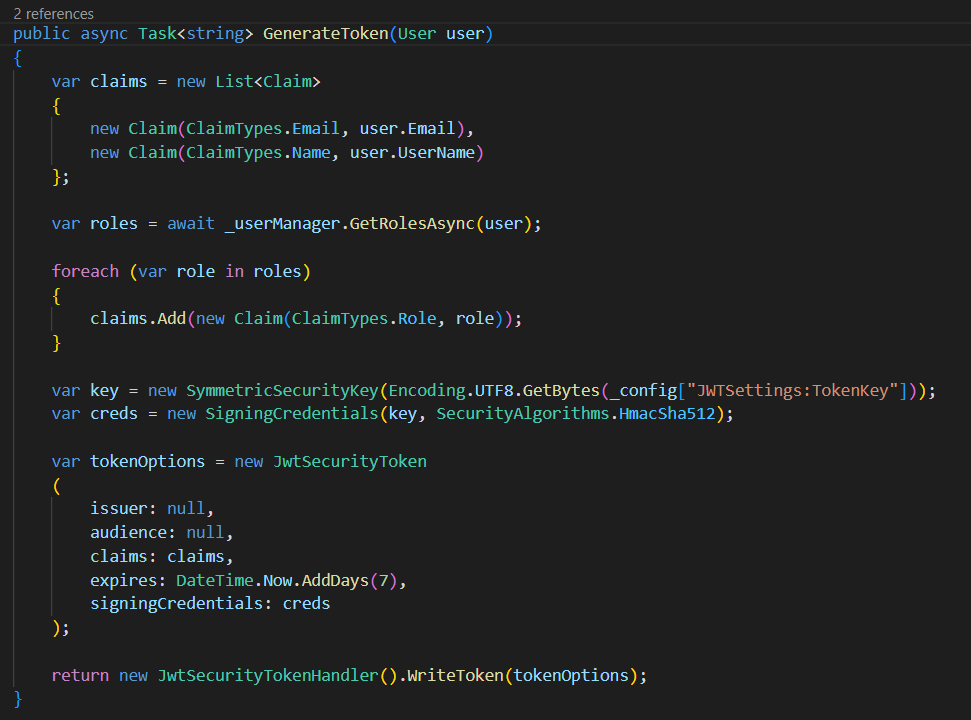


Спочатку ми намагаємось за допомогою userManager знайти чи такий юзер існує у базі. Якщо його немає або пароль не збігається - кидаємо unathorized.

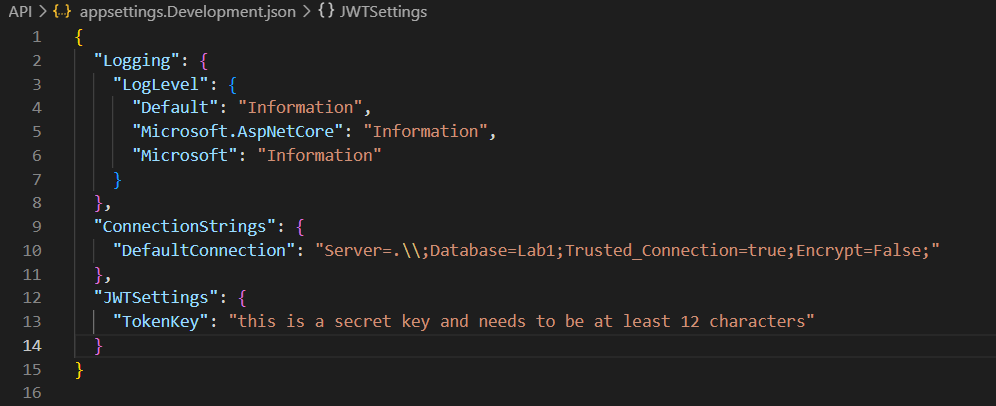
Далі у нас є наступна логіка - Юзер міг не авторизовуватись і створювати корзину через згенерований buyerId. Корзина закріплена одна за кожним юзером. Якщо у нього уже є попередня корзина ми її замінюємо на корзину до логіну, а корзину з buyerId видаляємо. В кінці повертаємо у респонсі емейл, токен, та корзину



запис у базі

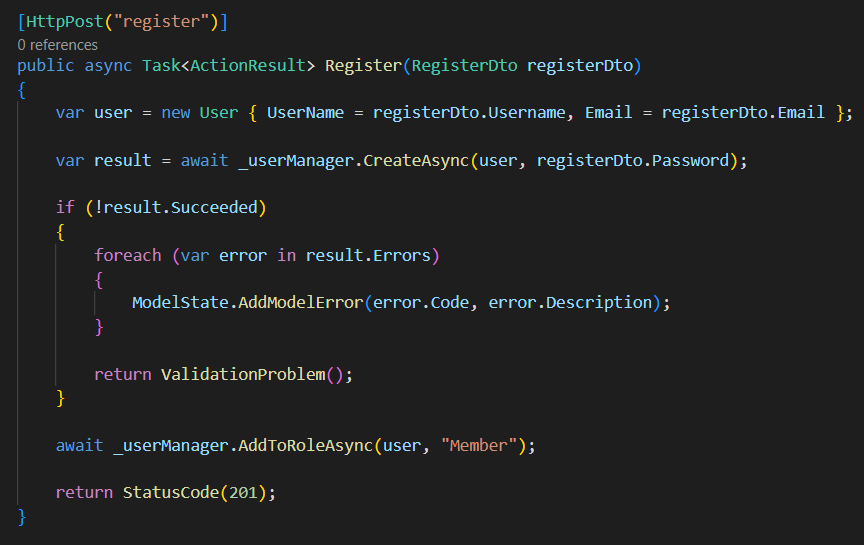


Токен генерується лише при успішній валідації. Токен має декілька компонентів - claims(Твердження), час тривалості токена та секретний ключ, у моєму прикладі такий



Ми стверджуємо, що у нас є такий юзер з таким ім’ям та емейлом, та має роль Member(користувач) або Admin(адмін) та генерується ключ. В подальшому ми будемо використувати цей ключ для того щоб передавати у хедер Authorization параметр та ідентифікувати юзера, але наразі цієї реалізації немає.

Реєстрація юзера



Майже все те саме, але ми задаємо при створенні юзера роль Member по дефолту

## Висновки:

На цій лабораторній роботі я навчився створювати Restful api для комунікації бекенду з фронтом, розібрався з тим як працює Cors, розібрався з авторизацією на asp net

Зони для покращення

1. Створення, видалення, редагування продуктів, а також захищення ендпоїнтів таким чином, щоб лише адмін міг редагувати продукти
2. Створення нових сутностей з CRUD операціями, а саме категорії та підкатегорії
3. Створення нової сутності order для оформлення замовлення

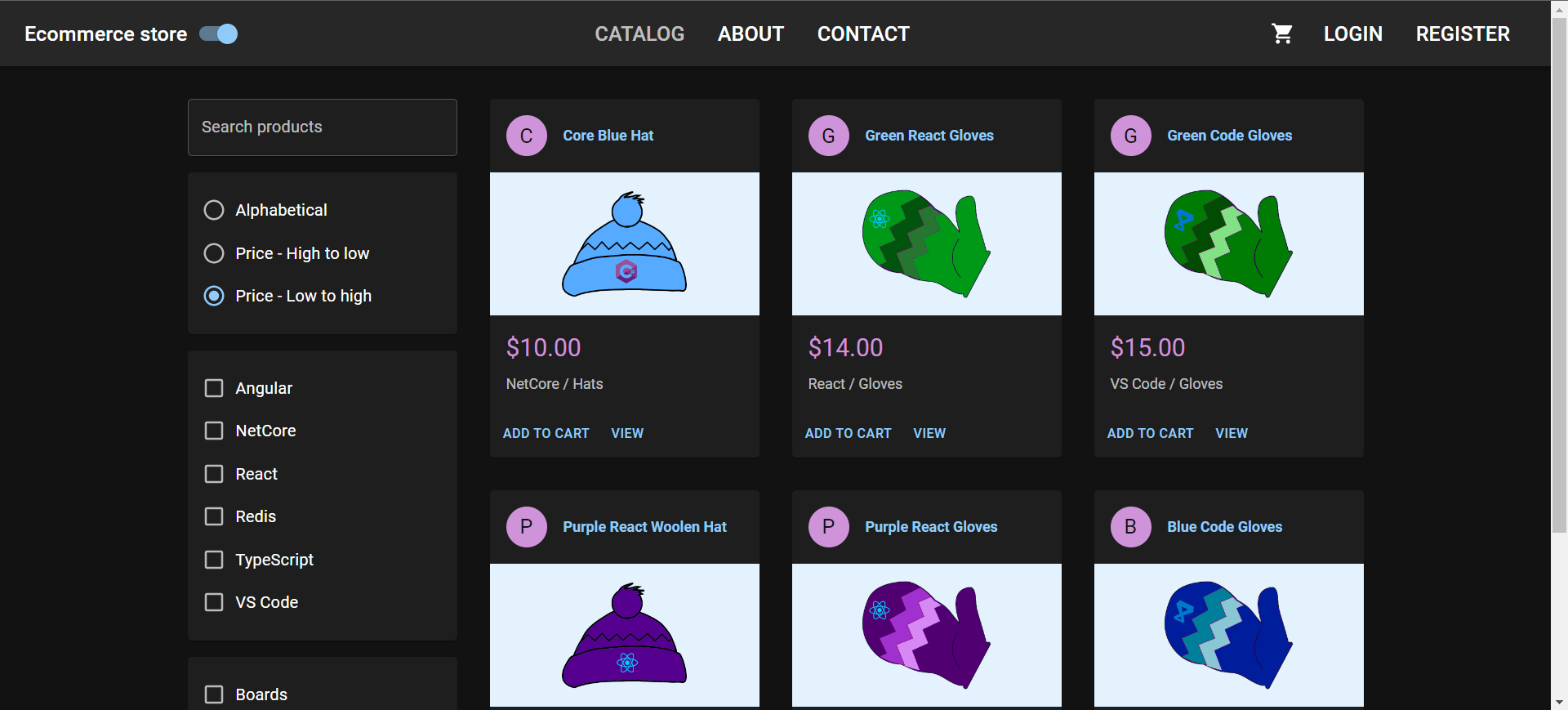
# Лабораторна робота №2

## Завдання:

Потрібно виконати наступні завдання:

1. Створити проєкт UI на будь-якій технології (Angular, React, Vue, Blazor Webassembly тощо, можна навіть html / css / js).
2. Реалізувати пошук або фільрацію “центральної” сутності по багатьох полях, як на сайтах аналогах *Наприклад, для каталогу фільмів це буде пошук фільмів по жанрах, роках випуску у прокак тощо.*
3. Реалізувати базові функції роботи з “центральною” сутністю БД:
   1. Виведення списку елементів
   2. Додавання елементів
   3. Редагування
   4. Видалення
4. Для **найсильніших** :)
   1. Додати JWT Auth.
   2. Додати обробку помилок.
   3. Реалізувати Clean Architecture.

### Завдання 1

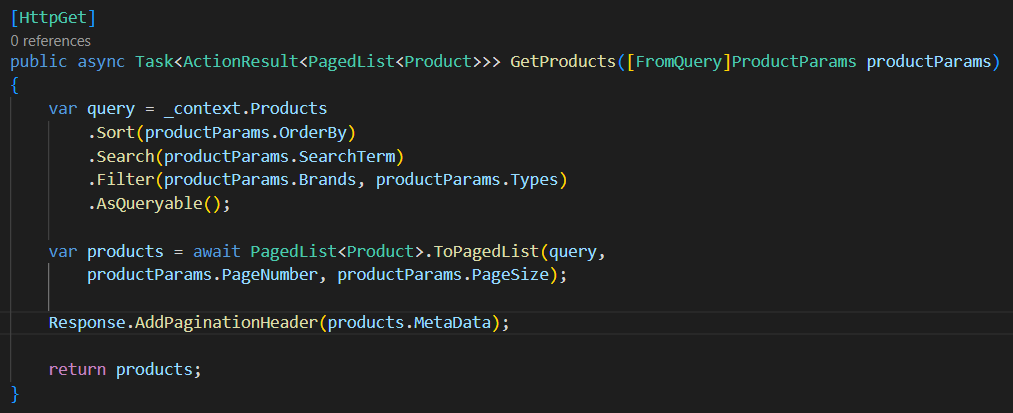


Сайт створений на базі реакту з додаванням багатьох бібліотек, основні із них -

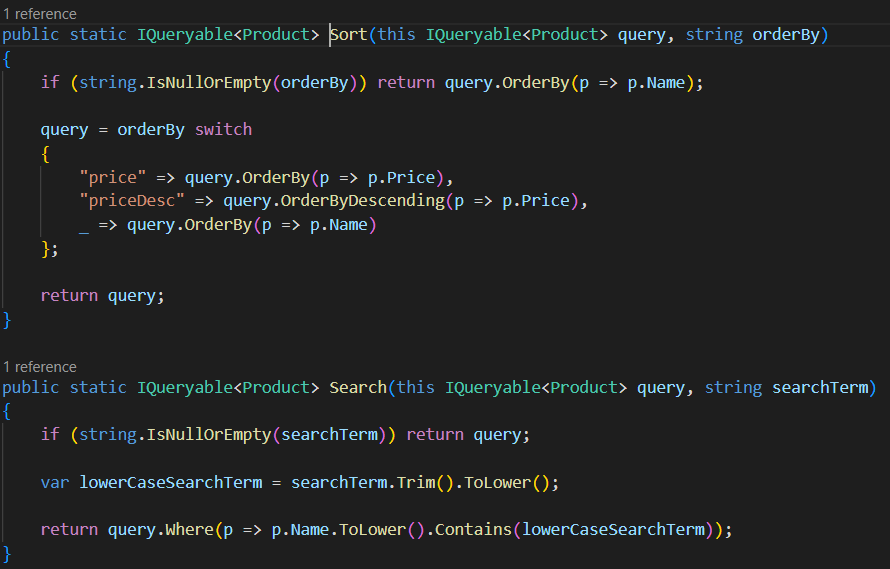
1. mui - бібліотека з багатьма заготовленими компонентами, спрощує розробку, не потрібно все робити з нуля
2. react-router-dom - роутинг по сторінках
3. redux - централізоване внутрішнє сховище на фронті, прив’язане до сесії, дозволяє отримувати дані з кожного компонента окремо
4. reduxjs - надбудова redux, спрощує структуру та поведінку взаємодії зі сховищем
5. typescript - строга типізація
6. axios - бібліотека для спрощення створення запитів на бекенд та отримання відповіді

## Завдання 2

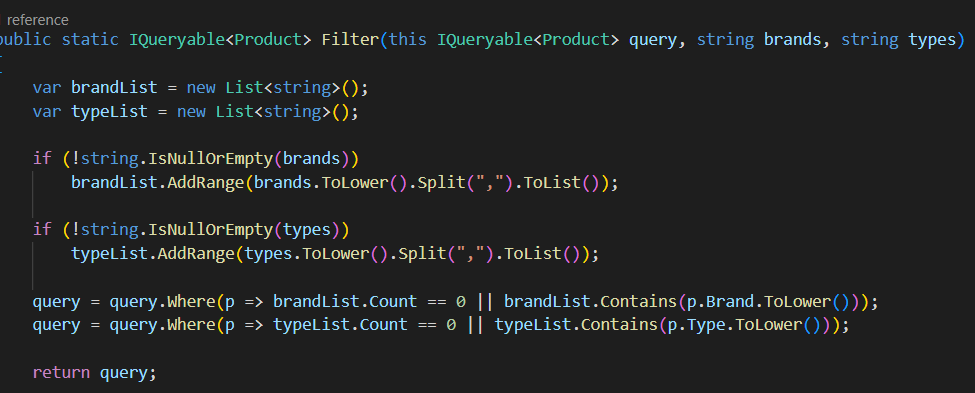
У нас є фільтрація по Алфавіту, по цінах, по Бренду та Типу продукту, а також по пошуку, також у нас реалізована пагінація.

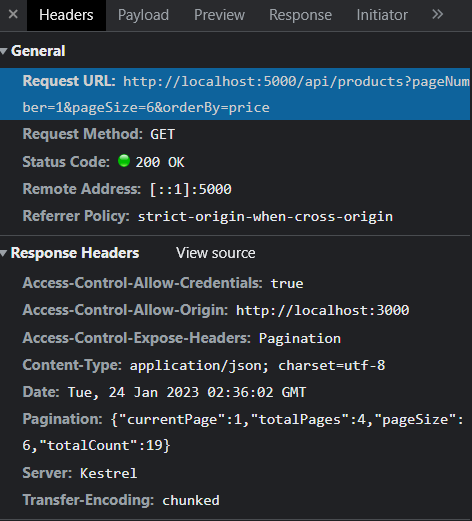


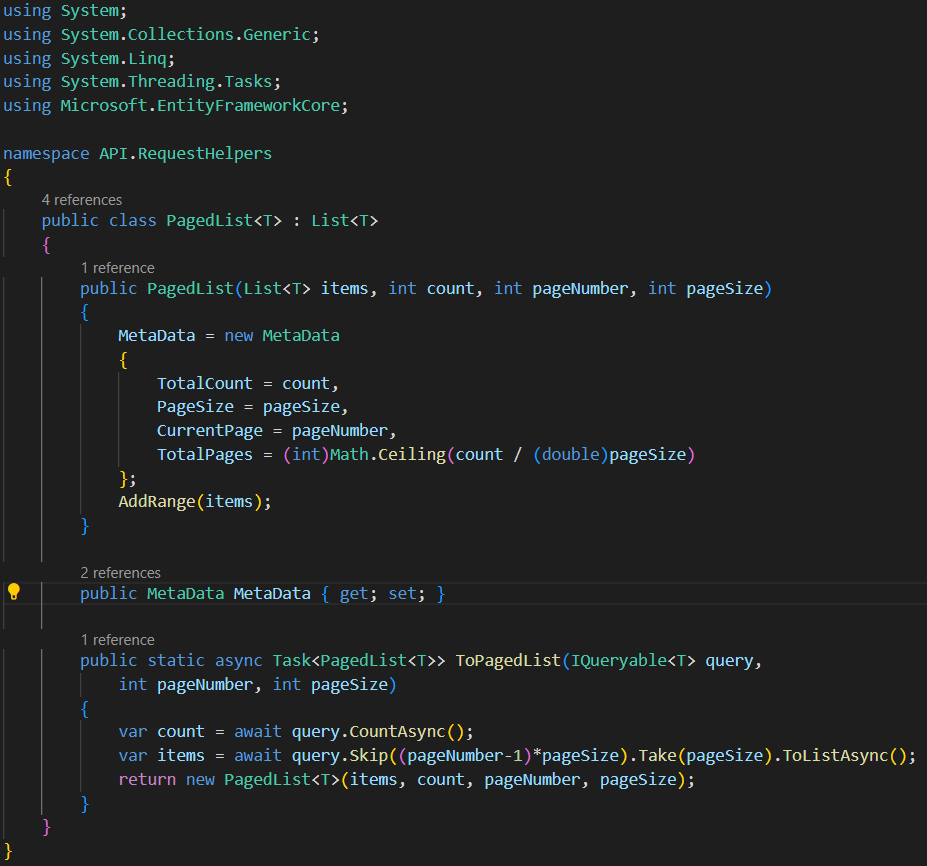
Для фільтрації використовуємо IQueryable -



Власні задані функції Сортування, Пошуку та Фільтрів



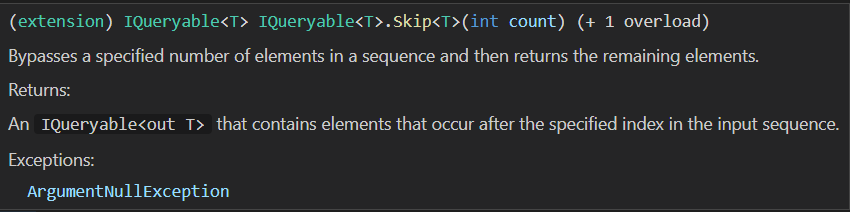




Приклад пагінації знайшов у інтернеті

Основний на мою думку функціонал знаходиться у

var items = await query.Skip((pageNumber-1)\*pageSize).Take(pageSize).ToListAsync();



Ми розділяємо весь масив відфільтрованих об`єктів, достаємо кількість сторінок, дістаємо параметр сторінки з запиту, а потім повертаємо масив об`єктів для заданої сторінки

Третій та четвертий пункт частково виконаний та описаний у першій лабораторній

### Висновки:

На цій лабораторній роботі я зробив справжню фільтрацію за багатьма параметрами, а також створив вивід та фільтрацію продуктів на клієнті

Використані джерела

* <https://khalilstemmler.com/articles/object-oriented/programming/4-principles/>
* <https://jakeydocs.readthedocs.io/en/latest/security/authentication/identity.html>
* <https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-7.0>
* <https://jwt.io/introduction>
* <https://learn.microsoft.com/ru-ru/dotnet/api/system.linq.iqueryable-1?view=net-7.0>
* <https://code-maze.com/paging-aspnet-core-webapi/>
* <https://mui.com/>
* <https://redux-toolkit.js.org/>