Projenin genel yapısı şu şekilde oluşturulmuştur:

- config
- controller
- exception
- mapper
- model
  - common
  - entity
  - enums
  - request
  - response
- repository
- security
  - jwt
- service
- util

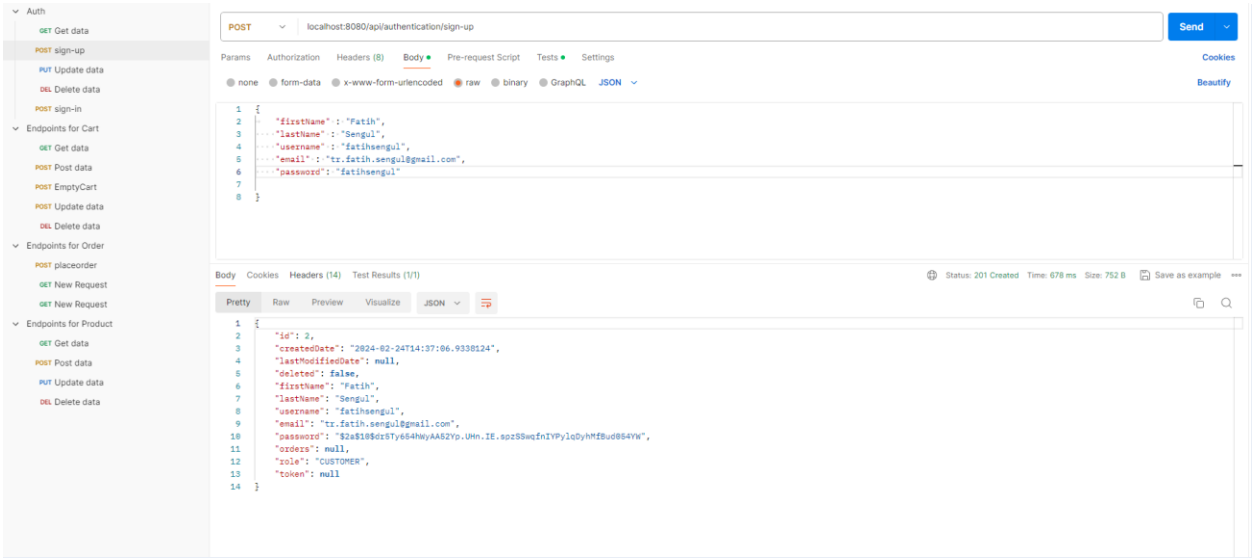
Kullanılan Teknolojiler

- Spring Boot framework
- Spring Data JPA
- Spring Security
- PostgreSQL veritabanı
- Lombok
- MapStruct
- JWT

Endpointler ve Servisler:

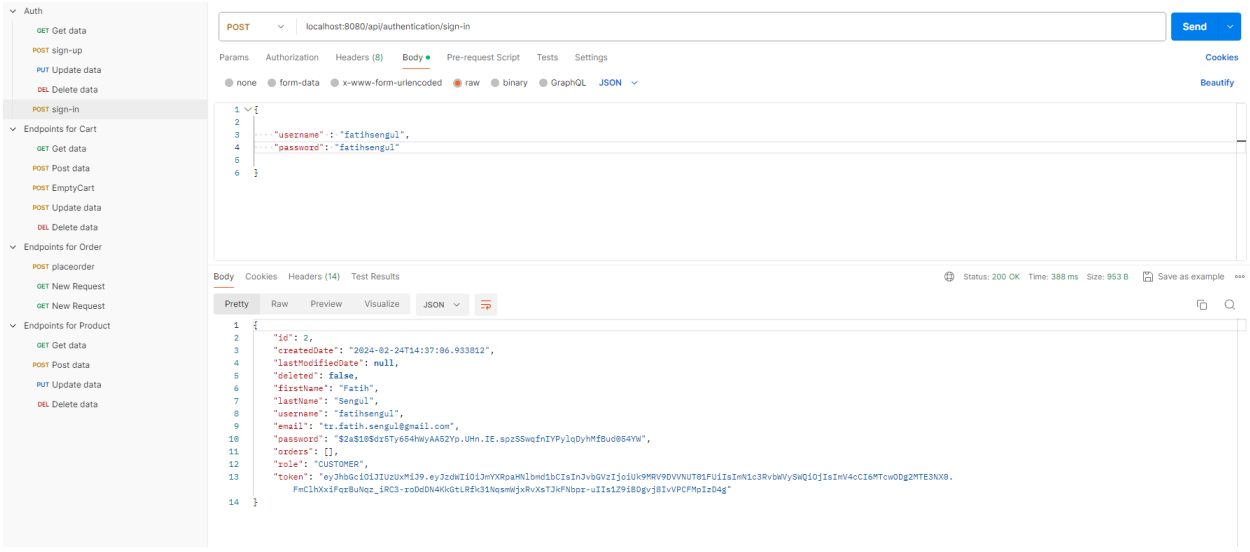
1. **Sign Up (POST /sign-up)**

`localhost:8080/api/authentication/sign-up`



2. **Sign In (POST /sign-in):**

`localhost:8080/api/authentication/sign-in`

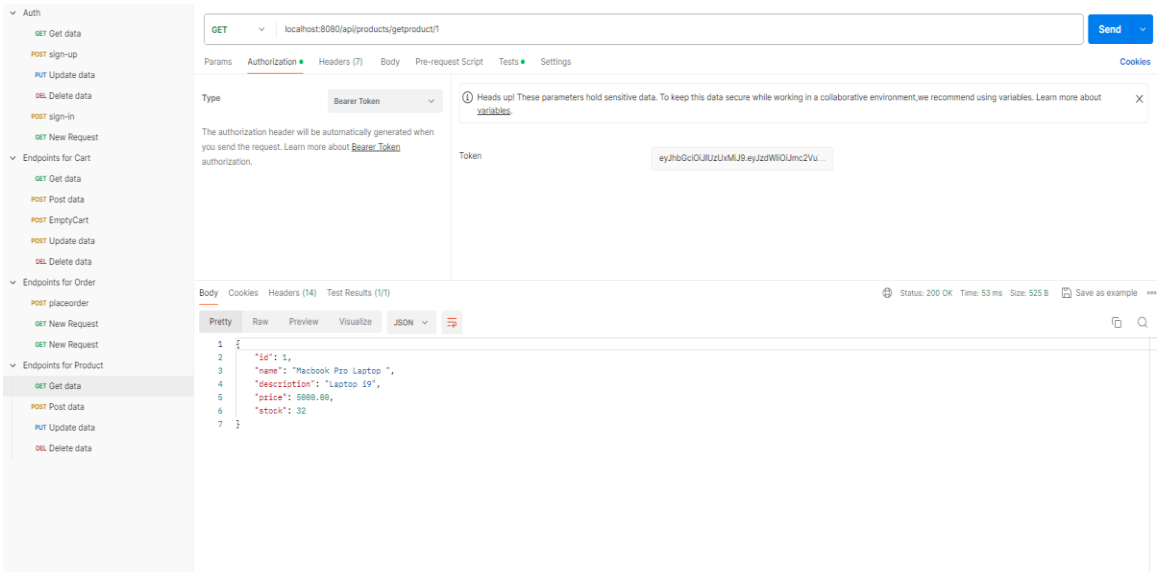


Ayrıca kullanıcıyı Admin yapabilmek için bir servis oluşturulmuştur. Bir ADMIN rolü başka bir kullanıcıyı ADMIN rolü ile yetkilendirebilmesi sağlanmıştır.

`localhost:8080/api/internal/make-admin/2`

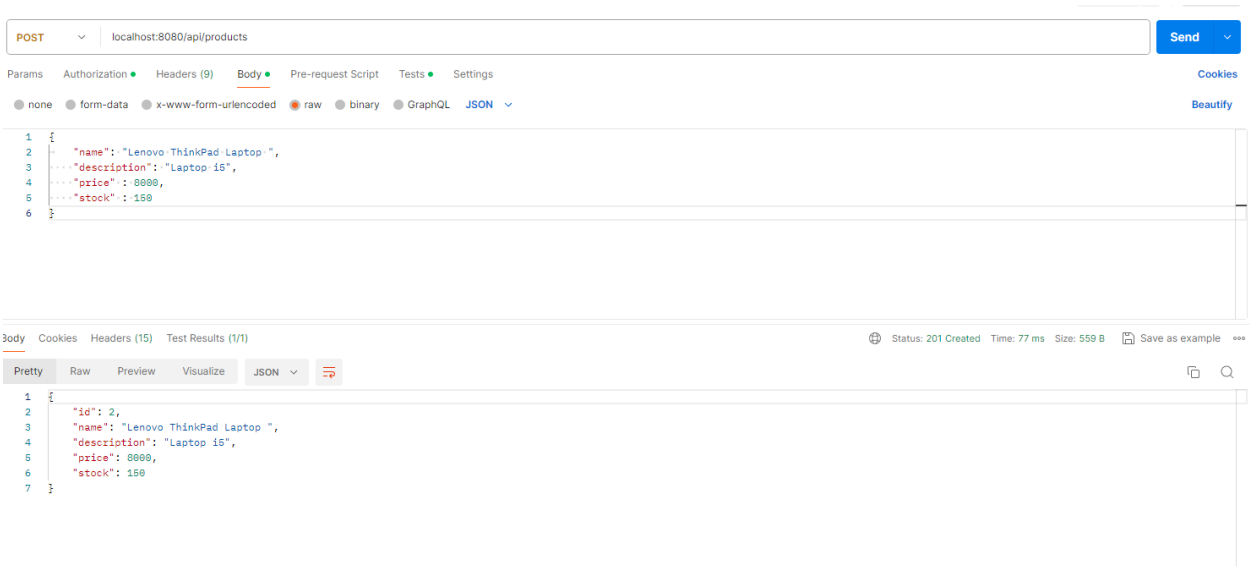
3. GetProduct (GET /getproduct):

`/api/authentication/getproduct/{productId}`



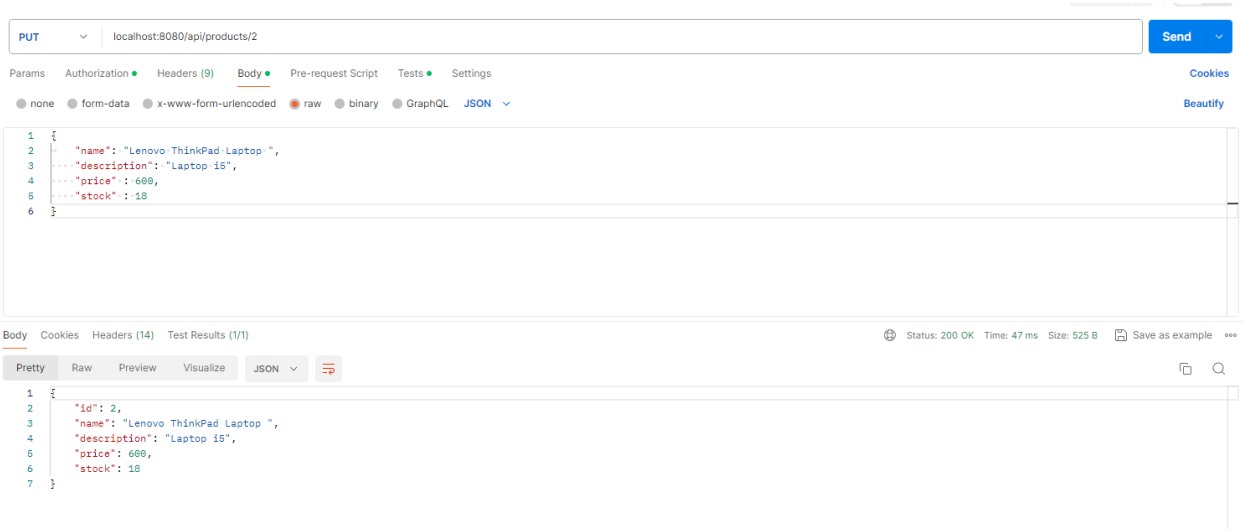
4. CreateProduct (POST/products)

`localhost:8080/api/products`



5. UpdateProduct(PUT/products)

`localhost:8080/api/products`

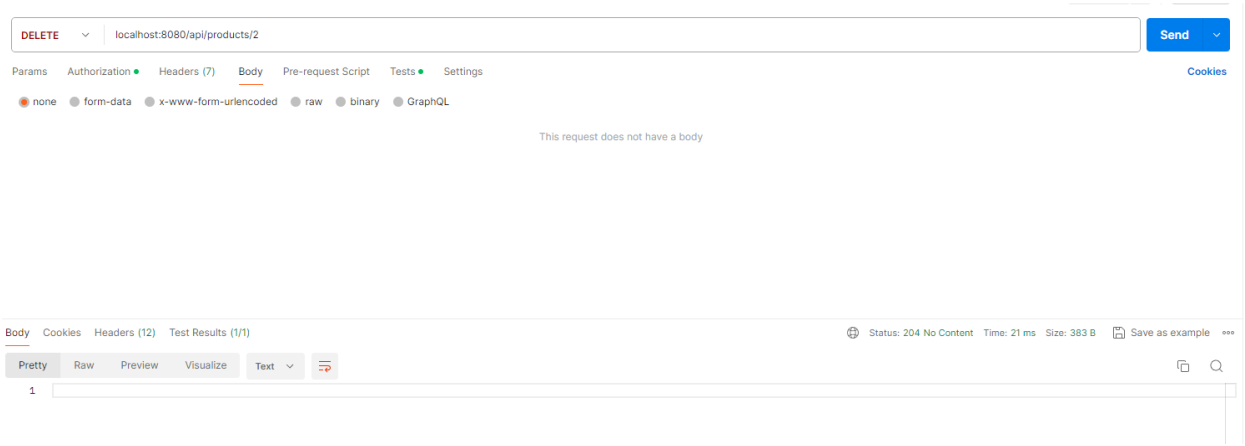


6. DeleteProduct(DELETE/products)

`localhost:8080/api/products/2`

Proje içerisinde Delete işlemi doğrudan db'den kaldırmak yerine deleted column'u true çekilir.

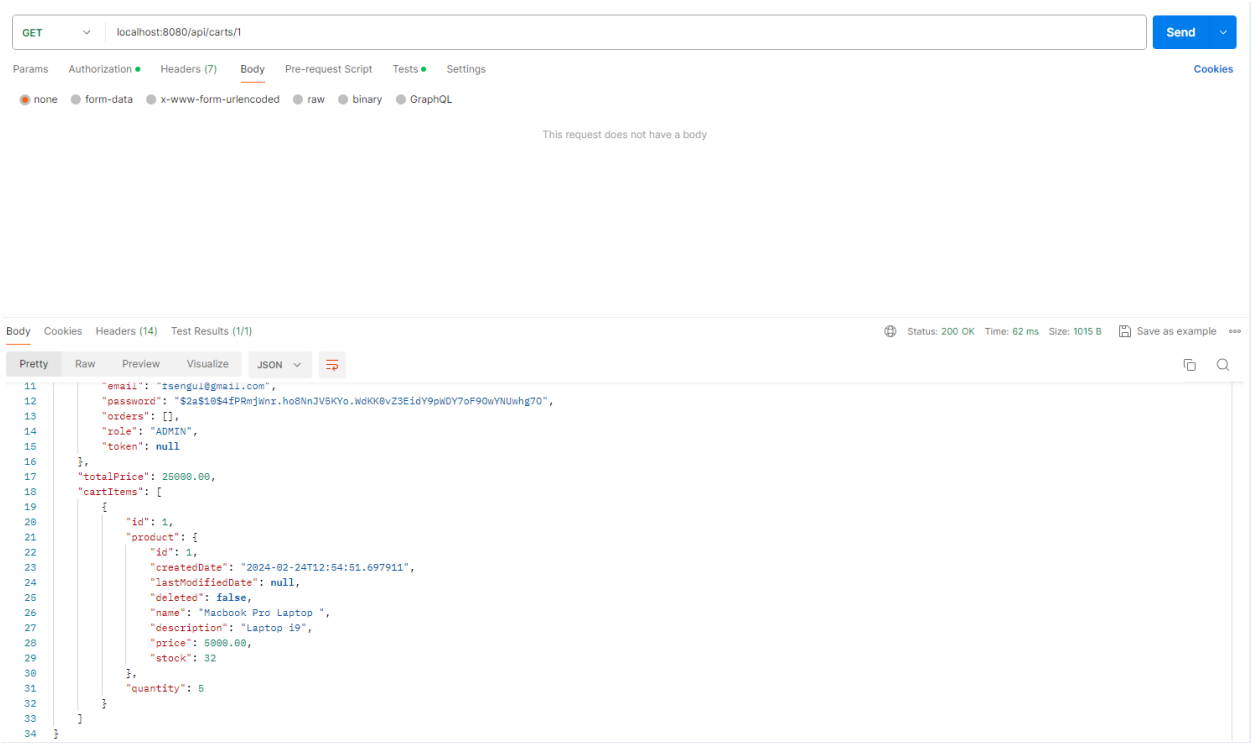
`localhost:8080/api/products/2`



	id [PK] bigint	created_date timestamp without time zone	deleted boolean	last_modified_date timestamp without time zone	description character varying (255)	name character varying (255)	price numeric (19,2)	stock integer
1	1	2024-02-24 12:54:51.697911	false	[null]	Laptop i9	Macbook Pro Laptop	5000.00	32
2	2	2024-02-24 14:56:35.831456	true	2024-02-24 15:01:56.618672	Laptop i5	Lenovo ThinkPad Laptop	600.00	18

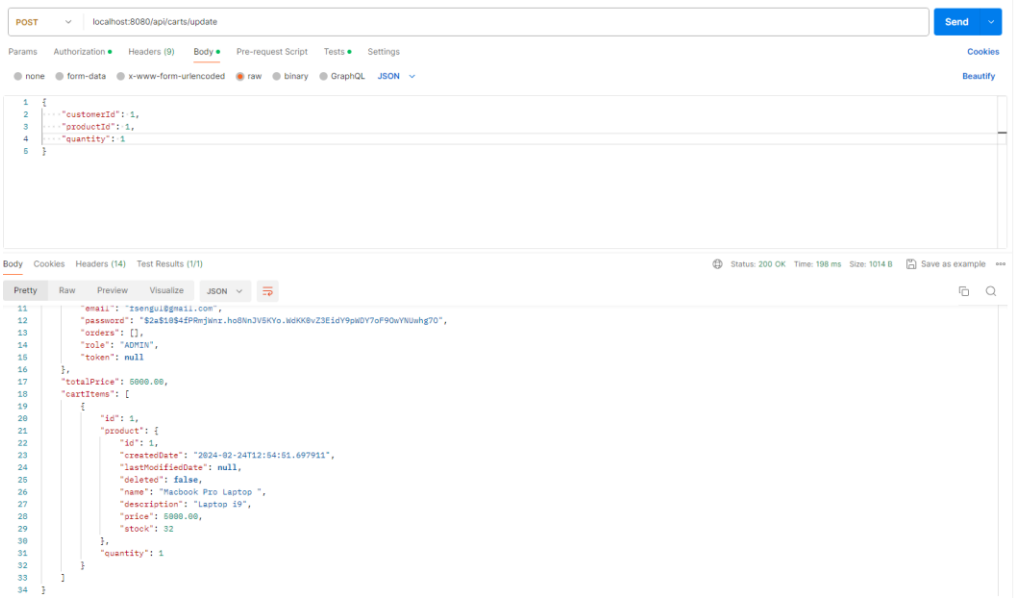
7. GetCart(GET/{customerId})

`localhost:8080/api/carts/2`



8. UpdateCart(POST/update)

`localhost:8080/api/carts/update`



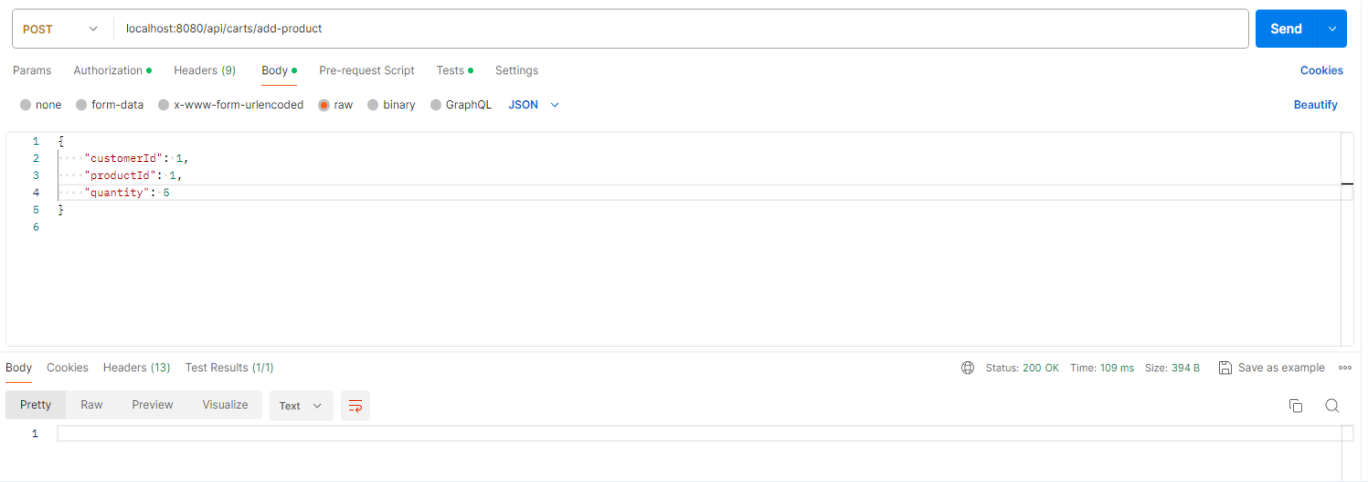
Her cart'a ekleme, çıkarma, miktar arttırma yapıldığında Total Price alanında güncellendiği görülmektedir.

9. EmptyCart(POST/empty)

`localhost:8080/api/carts/empty`

10. AddProductToCart(POST/add-product)

`localhost:8080/api/carts/add-product`

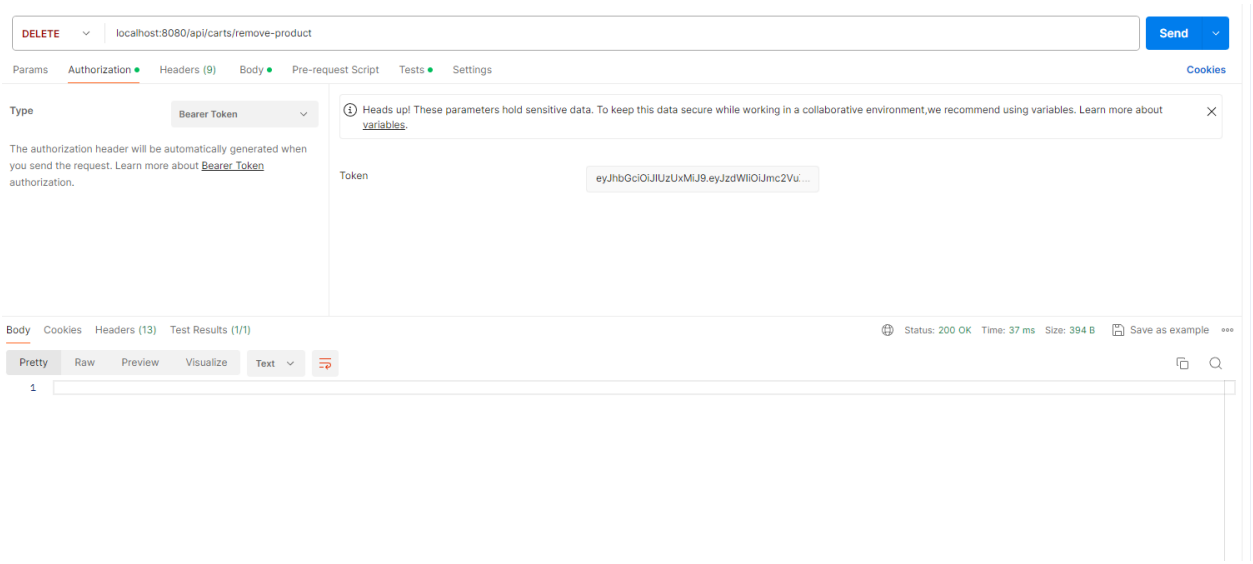


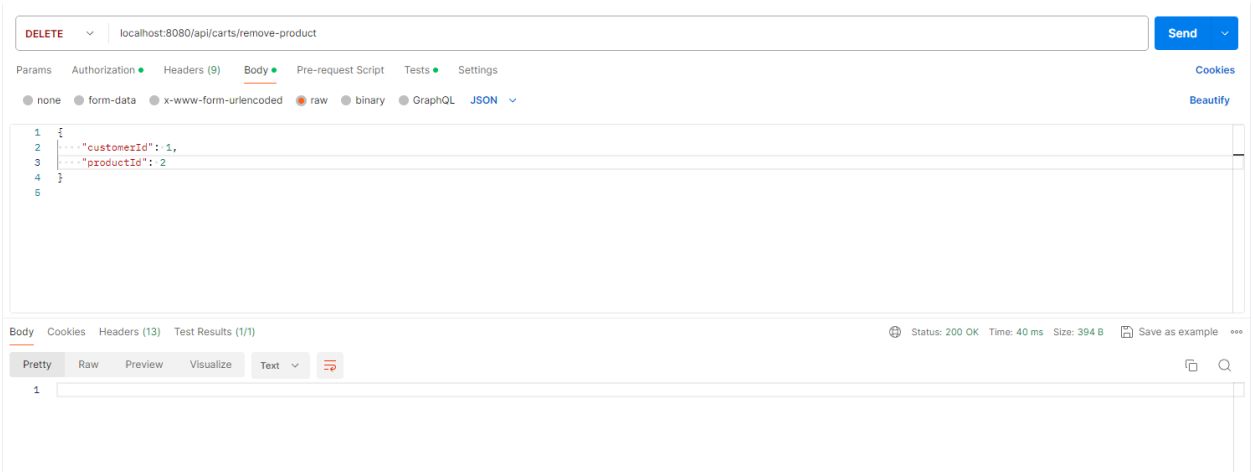
Aynı üründen tekrardan sepete eklenmek istenirse aynı üründe olup olmadığı kontrol edilerek o ürünün eklenen miktar kadar toplam ürün miktarı arttırılmıştır. Product ile CartItem arasında OneToOne ilişkisi kurulmuştur.

```java
public class CartItem extends BaseEntity {


    @ManyToOne
    @JoinColumn(name = "cart_id")
    @JsonBackReference
    private Cart cart;

    @OneToOne
    @JoinColumn(name = "product_id")
    private Product product;

    private Integer quantity;

    @ManyToOne
    @JsonIgnore
    private Order order;
```

11. RemoveProductToCart(DELETE/remove-product)

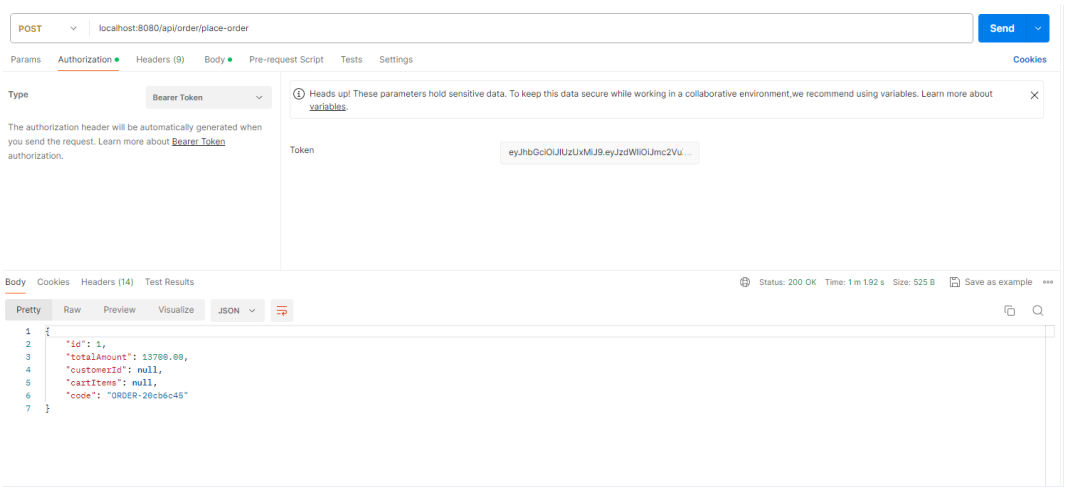
`localhost:8080/api/carts/remove-product`





12. PlaceOrder(POST/place-order)

`localhost:8080/api/order/place-order`



Place Order işlemi tamamlandıktan sonra cart'ın boşatılması sağlanmıştır.

13. GetOrderForCode(GET/order{ORDER-})

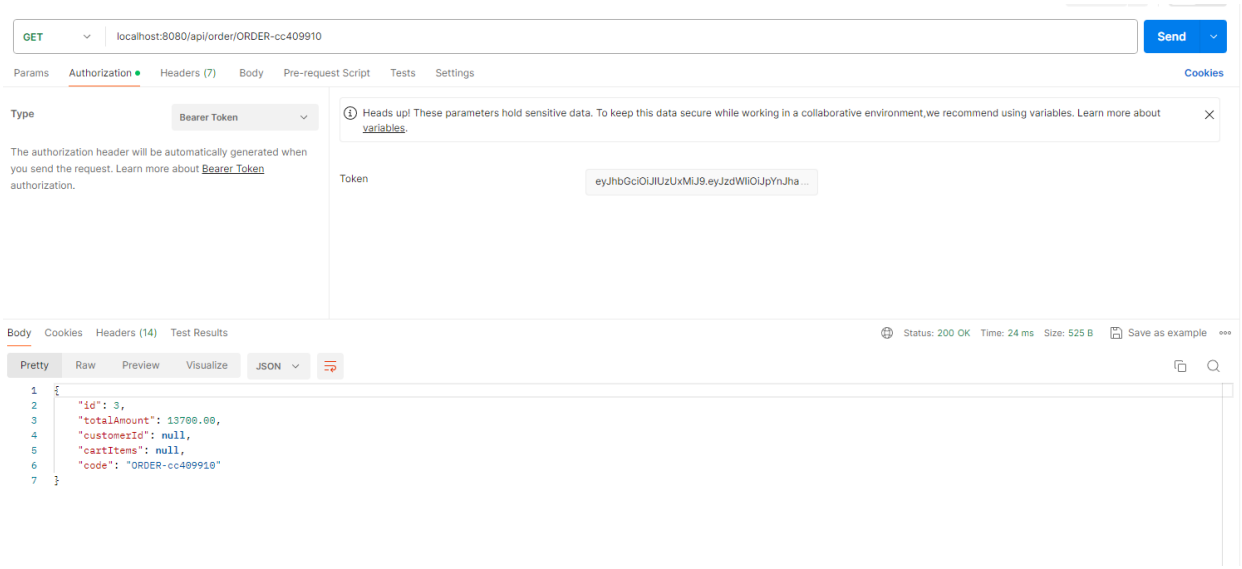
`localhost:8080/api/order/ORDER-f8e60e23`

OrderHistory Tablosu oluşturularak geçmişe dönük siparişlerin ve sipraiş sırasındaki fiyatların tutulması sağlanmıştır. Ayrıca bir sipariş code oluşturularak bunların sorgulanması sağlanmıştır. Order Code oluşturma için UUID kütüphanesi kullanılmıştır.

```
1  select * from commerce.order_history
```

Data Output | Messages | Notifications

	id [PK] bigint	created_date timestamp without time zone	deleted boolean	last_modified_date timestamp without time zone	price numeric (19,2)	quantity integer	order_id bigint	product_id bigint
1	1	2024-02-24 15:52:42.668906	false	[null]	1000.00	4	1	4
2	2	2024-02-24 15:52:42.684909	false	[null]	500.00	7	1	3
3	3	2024-02-24 15:52:42.688908	false	[null]	600.00	2	1	2
4	4	2024-02-24 15:52:42.69591	false	[null]	5000.00	1	1	1



14. GetAllOrdersForCustomer

`localhost:8080/api/order/customer/3`