

Projenin genel yapısı şu şekilde oluşturulmuştur:

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

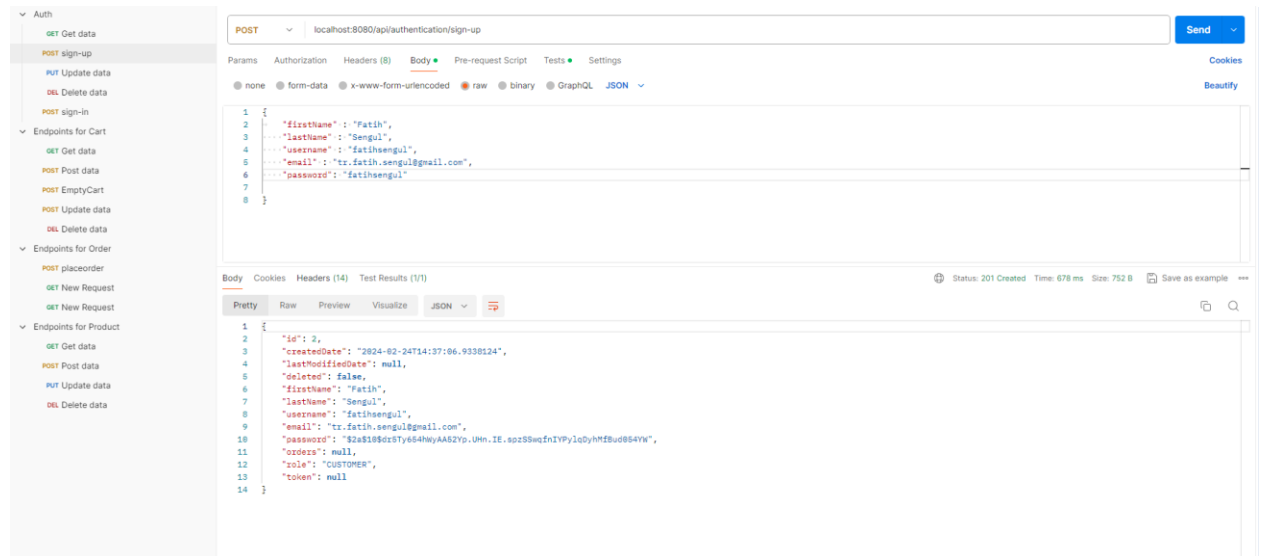
Kullanılan Teknolojiler

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

Endpointler ve Servisler:

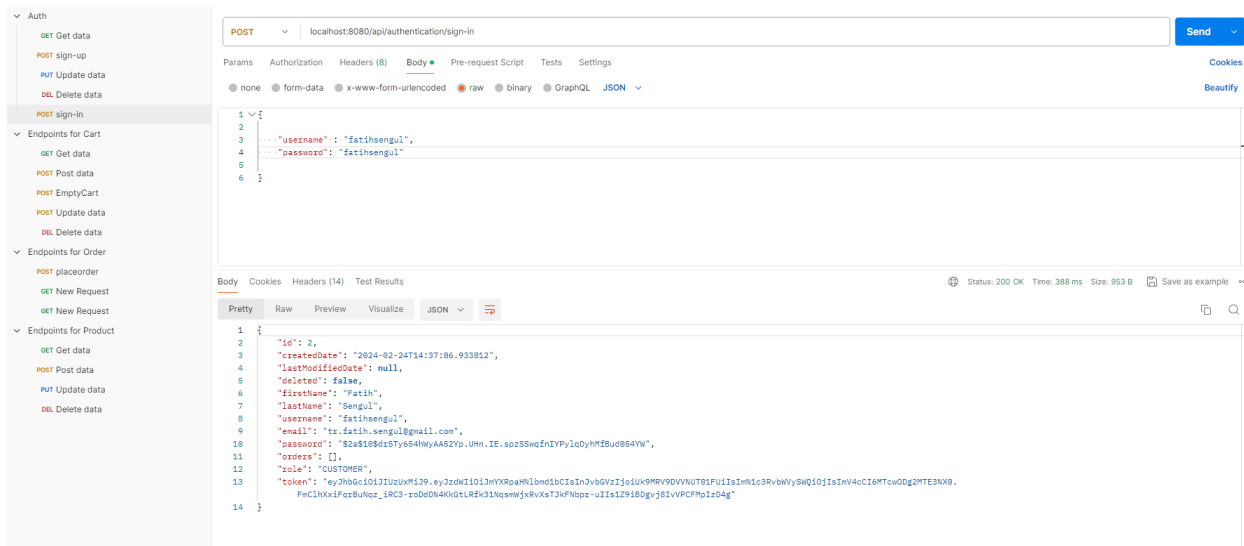
1. Sign Up (POST /sign-up)

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}`

The screenshot shows a REST client interface with a GET request to `localhost:8080/api/products/getproduct/1`. The request is authenticated with a Bearer Token. The response is a JSON object with the following structure:

```
1 {
2   "id": 1,
3   "name": "Macbook Pro Laptop ",
4   "description": "Laptop 19",
5   "price": 9999.99,
6   "stock": 32
7 }
```

4. CreateProduct (POST /products)

`localhost:8080/api/products`

The screenshot shows a REST client interface with a POST request to `localhost:8080/api/products`. The request body is a JSON object with the following structure:

```
1 {
2   "name": "Lenovo ThinkPad Laptop ",
3   "description": "Laptop 15",
4   "price": 8888,
5   "stock": 158
6 }
```

The response is a JSON object with the following structure:

```
1 {
2   "id": 2,
3   "name": "Lenovo ThinkPad Laptop ",
4   "description": "Laptop 15",
5   "price": 8888,
6   "stock": 158
7 }
```

5. UpdateProduct(PUT/products)

localhost:8080/api/products

The screenshot shows a REST client interface with a PUT request to `localhost:8080/api/products/2`. The request body is a JSON object: `{ "name": "Lenovo ThinkPad Laptop ", "description": "Laptop i5", "price": 600, "stock": 18 }`. The response is shown in the bottom pane, displaying the same JSON object with an additional `"id": 2` field. The status is 200 OK.

```
1 {
2   "name": "Lenovo ThinkPad Laptop ",
3   "description": "Laptop i5",
4   "price": 600,
5   "stock": 18
6 }
```

```
1 {
2   "id": 2,
3   "name": "Lenovo ThinkPad Laptop ",
4   "description": "Laptop i5",
5   "price": 600,
6   "stock": 18
7 }
```

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

The screenshot shows a REST client interface with a DELETE request to `localhost:8080/api/products/2`. The response status is 204 No Content. Below the REST client, a database table is shown with the following data:

	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

GET localhost:8080/api/carts/1

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (14) Test Results (1/1) Status: 200 OK Time: 62 ms Size: 1015 B Save as example

```
11 {
12   "email": "isengui@gmail.com",
13   "password": "$2a$10$4fPRmjWnz.ho8NnJV5KY0.WdKK6vZ3EidY9pWdY7oF9QwYNUwhg70",
14   "orders": [],
15   "role": "ADMIN",
16   "token": null
17 },
18 "totalPrice": 25000.00,
19 "cartItems": [
20   {
21     "id": 1,
22     "product": {
23       "id": 1,
24       "createdAt": "2024-02-24T12:54:51.697911",
25       "lastModifiedDate": null,
26       "deleted": false,
27       "name": "Macbook Pro Laptop ",
28       "description": "Laptop 19",
29       "price": 6000.00,
30       "stock": 32
31     },
32     "quantity": 5
33   }
34 ]
```

8. UpdateCart(POST/update)

localhost:8080/api/carts/update

POST localhost:8080/api/carts/update

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "customerId": 1,
3   "productId": 1,
4   "quantity": 1
5 }
```

Body Cookies Headers (14) Test Results (1/1) Status: 200 OK Time: 198 ms Size: 1014 B Save as example

```
11 {
12   "email": "isengui@gmail.com",
13   "password": "$2a$10$4fPRmjWnz.ho8NnJV5KY0.WdKK6vZ3EidY9pWdY7oF9QwYNUwhg70",
14   "orders": [],
15   "role": "ADMIN",
16   "token": null
17 },
18 "totalPrice": 6000.00,
19 "cartItems": [
20   {
21     "id": 1,
22     "product": {
23       "id": 1,
24       "createdAt": "2024-02-24T12:54:51.697911",
25       "lastModifiedDate": null,
26       "deleted": false,
27       "name": "Macbook Pro Laptop ",
28       "description": "Laptop 19",
29       "price": 6000.00,
30       "stock": 32
31     },
32     "quantity": 1
33   }
34 ]
```

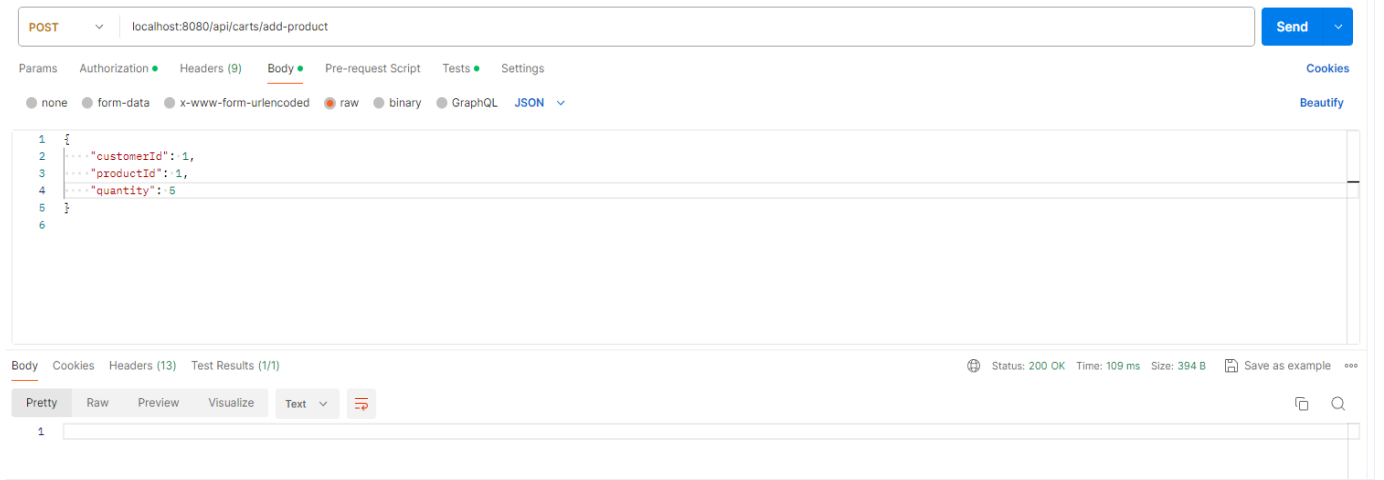
Her cart'a ekleme, çıkarma, miktar artırma yapıldığında Total Price alanında güncellendiği görülmektedir.

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 artırılmıştır. Product ile CartItem arasında OneToOne ilişkisi kurulmuştur.

```
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

The screenshot displays a REST client interface for a DELETE request to `localhost:8080/api/carts/remove-product`. The request is configured with a Bearer Token authorization header. The response is shown in the bottom pane, indicating a 200 OK status.

Request Configuration:

- Method: DELETE
- URL: localhost:8080/api/carts/remove-product
- Authorization: Bearer Token
- Token: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJmc2Vh...

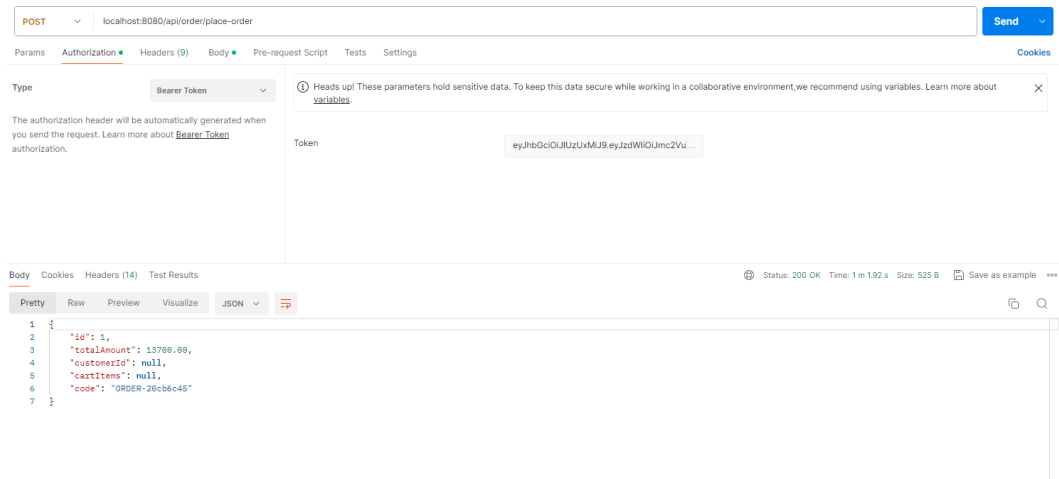
Response:

```
1 {
2   "customerId": 1,
3   "productId": 2
4 }
5
```

Status: 200 OK Time: 40 ms Size: 394 B

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 sipariş 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										
	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		

GET localhost:8080/api/order/ORDER-cc409910 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token eyJhbGciOiJIUzUxMU9.eyJzdWIiOiUpYnJha...

Body Cookies Headers (14) Test Results Status: 200 OK Time: 24 ms Size: 525 B Save as example

```
1 {
2   "id": 3,
3   "totalAmount": 13700.00,
4   "customerId": null,
5   "cartItems": null,
6   "code": "ORDER-cc409910"
7 }
```

14. GetAllOrdersForCustomer

localhost:8080/api/order/customer/3

GET localhost:8080/api/order/customer/3 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token eyJhbGciOiJIUzUxMU9.eyJzdWIiOiUpYnJha...

Body Cookies Headers (14) Test Results Status: 200 OK Time: 57 ms Size: 589 B Save as example

```
1 [
2   {
3     "id": 9,
4     "price": 1000.00,
5     "quantity": 4
6   },
7   {
8     "id": 10,
9     "price": 500.00,
10    "quantity": 7
11  },
12  {
13    "id": 11,
14    "price": 600.00,
15    "quantity": 2
16  },
17  {
18    "id": 12,
19    "price": 5000.00,
20    "quantity": 1
21  }
22 ]
```