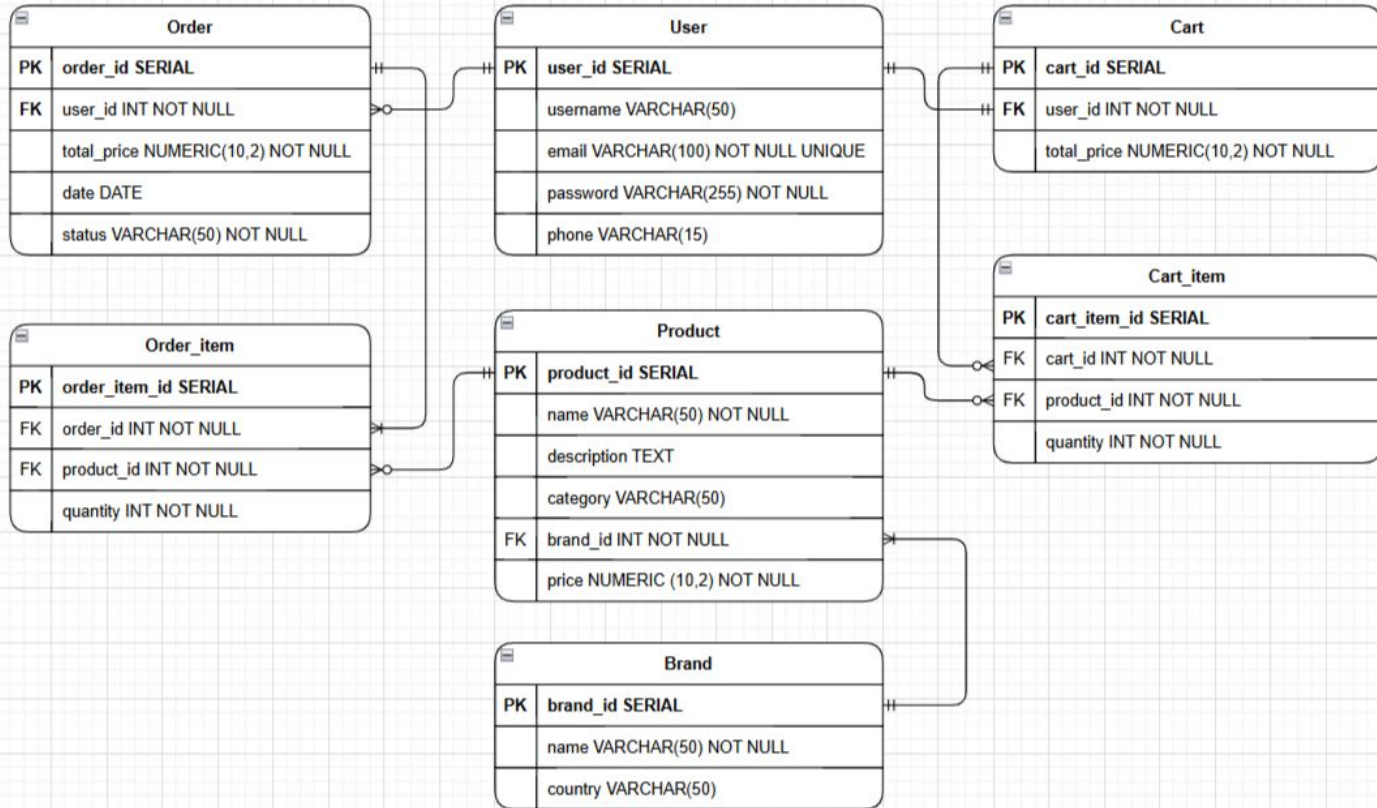


Интернет магазин косметики “Silver Pear”

Выполнили Жданова Валерия и
Русяева Софья 6301-010302D

Схема БД



Настроены маршруты для взаимодействия с API

Например для Product:

- **GET** /api/shop/product - получение списка всех продуктов
- **GET** /api/shop/product/{productId} - получение информации о конкретном продукте
- **POST** /api/shop/product - добавление продукта
- **PUT** /api/shop/product/{productId} - изменение продукта
- **DELETE** /api/shop/product/{productId} - удаление продукта

CRUD-методы

The screenshot shows an IDE with a project named 'WebApp' and a file explorer on the left. The project structure is as follows:

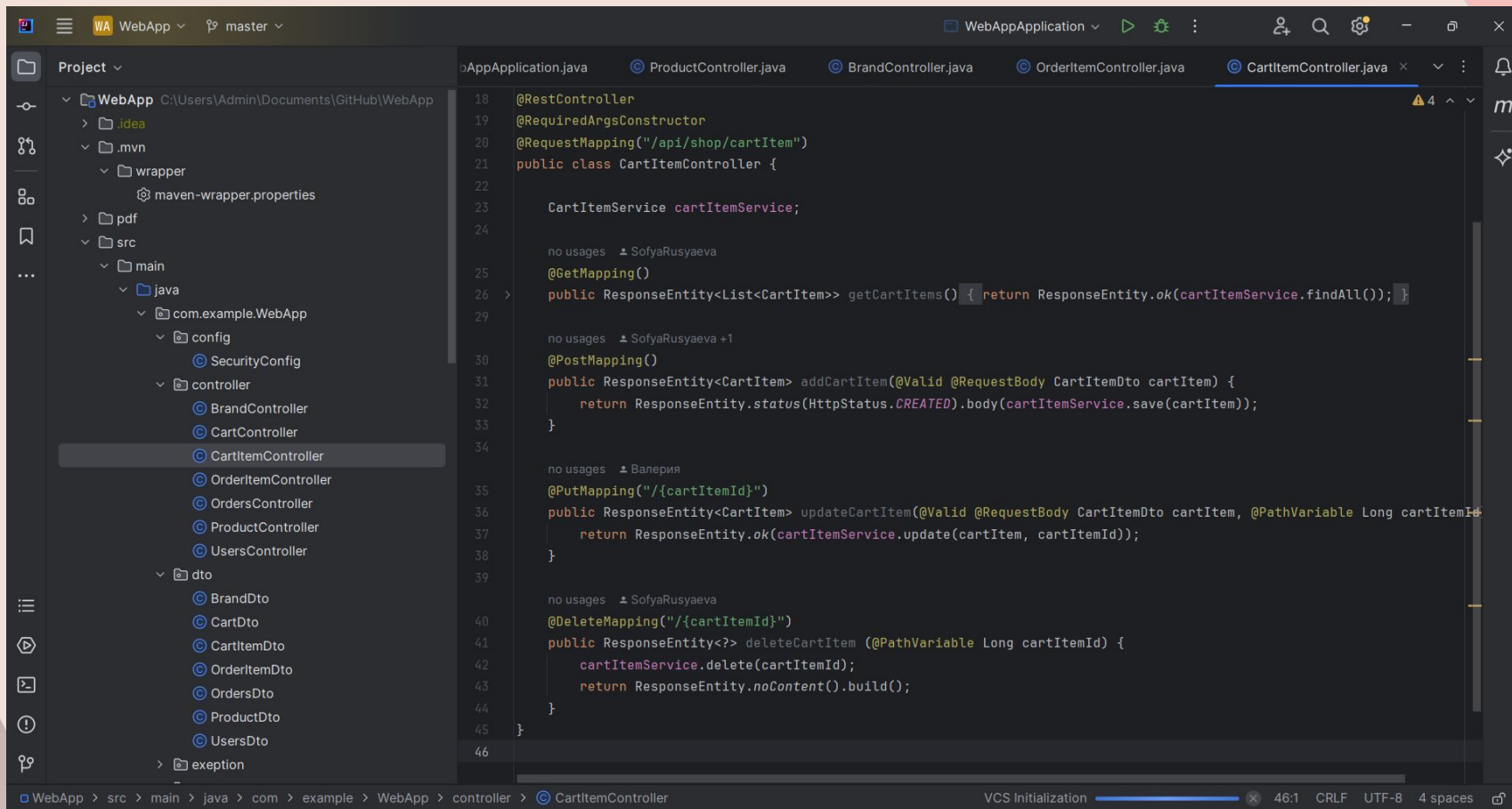
- WebApp
 - .idea
 - .mvn
 - wrapper
 - maven-wrapper.properties
 - src
 - main
 - java
 - com.example.WebApp
 - config
 - SecurityConfig
 - controller
 - BrandController
 - CartController
 - CartItemController
 - OrderItemController
 - OrdersController
 - ProductController
 - UsersController
 - dto
 - BrandDto
 - CartDto
 - CartItemDto
 - OrderItemDto
 - OrdersDto
 - ProductDto
 - UsersDto
 - exception

The main editor displays the `ProductController.java` file, which implements the following methods:

```
// ...  
42 //  
43  
44 no usages SofiaRusyaeva  
45 @GetMapping("/{productId}")  
46 public ResponseEntity<Product> getProductById(@PathVariable Long productId) {  
47     return ResponseEntity.ok(productService.findById(productId));  
48 }  
49  
50 no usages SofiaRusyaeva +1  
51 @PostMapping()  
52 public ResponseEntity<Product> addProduct(@Valid @RequestBody ProductDto product) {  
53     return ResponseEntity.status(HttpStatus.CREATED).body(productService.save(product));  
54 }  
55  
56 no usages Валерия  
57 @PutMapping("/{productId}")  
58 public ResponseEntity<Product> updateProduct(@Valid @RequestBody ProductDto product, @PathVariable Long productId) {  
59     return ResponseEntity.ok(productService.update(product, productId));  
60 }  
61  
62 no usages SofiaRusyaeva +1  
63 @DeleteMapping("/{productId}")  
64 public ResponseEntity<?> deleteProduct (@PathVariable Long productId) {  
65     productService.delete(productId);  
66     return ResponseEntity.noContent().build();  
67 }  
68 }
```

The status bar at the bottom indicates the current file is `ProductController` in the path `WebApp > src > main > java > com > example > WebApp > controller`. It also shows 'VCS Initialization' progress, a 66:1 line count, and encoding settings of CRLF and UTF-8.

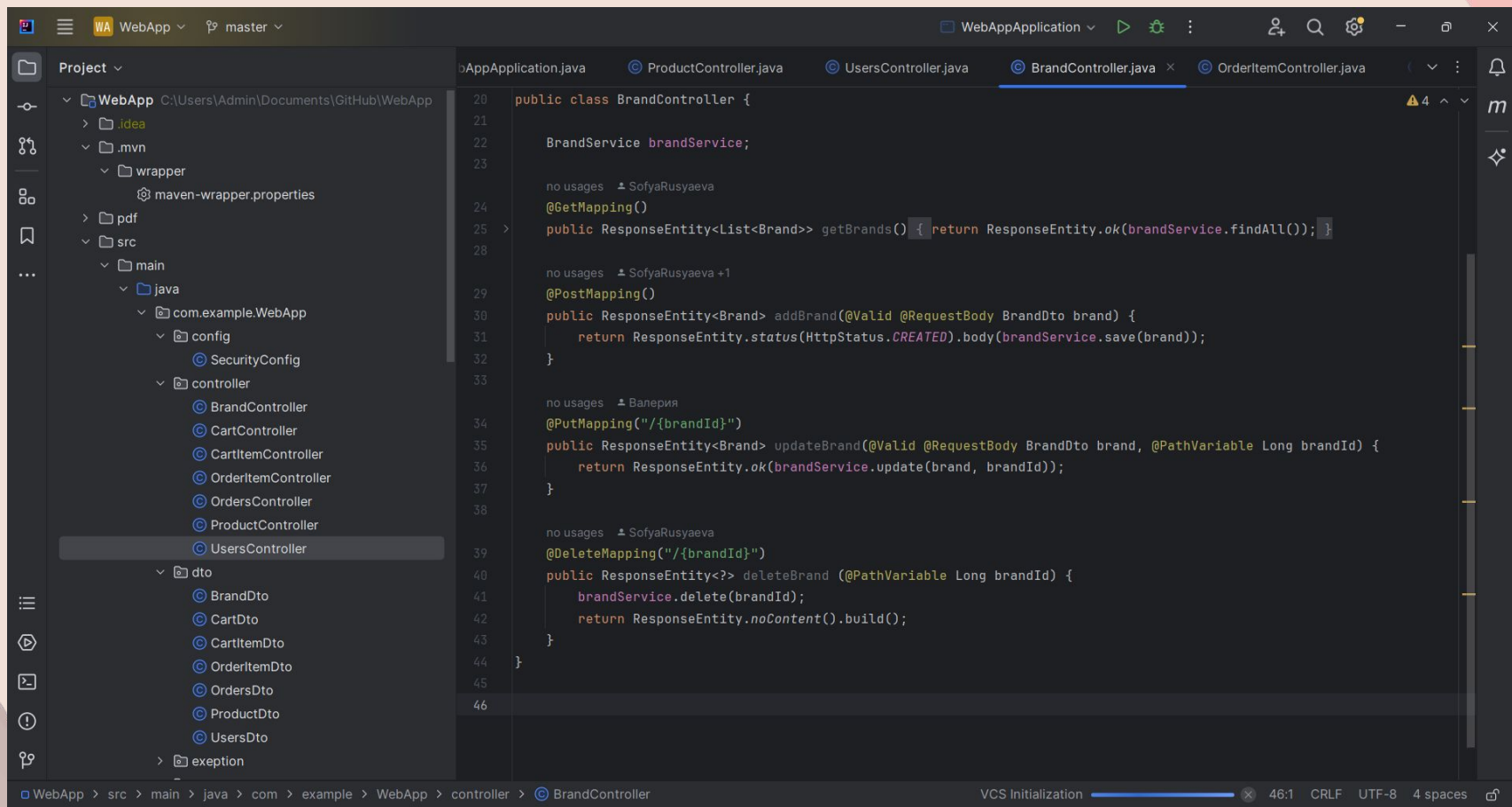
CRUD-методы



The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Displays the project structure. The path `src/main/java/com/example/WebApp/controller` is expanded, showing several controller classes. `CartItemController` is selected and highlighted.
- Code Editor (Center):** Displays the source code of `CartItemController.java`. The code includes:
 - Annotations: `@RestController`, `@RequiredArgsConstructor`, and `@RequestMapping("/api/shop/cartItem")`.
 - Field: `CartItemService cartItemService;`
 - Method `findAll()`: `public ResponseEntity<List<CartItem>> getCartItems() { return ResponseEntity.ok(cartItemService.findAll()); }`
 - Method `addCartItem()`: `public ResponseEntity<CartItem> addCartItem(@Valid @RequestBody CartItemDto cartItem) { return ResponseEntity.status(HttpStatus.CREATED).body(cartItemService.save(cartItem)); }`
 - Method `updateCartItem()`: `public ResponseEntity<CartItem> updateCartItem(@Valid @RequestBody CartItemDto cartItem, @PathVariable Long cartItemId) { return ResponseEntity.ok(cartItemService.update(cartItem, cartItemId)); }`
 - Method `deleteCartItem()`: `public ResponseEntity<?> deleteCartItem(@PathVariable Long cartItemId) { cartItemService.delete(cartItemId); return ResponseEntity.noContent().build(); }`
- Bottom Panel:** Shows the breadcrumb path `WebApp > src > main > java > com > example > WebApp > controller > CartItemController`. On the right, it indicates `VCS Initialization` and file encoding details: `46:1 CRLF UTF-8 4 spaces`.

CRUD-методы



The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Displays the project structure. The path `WebApp > src > main > java > com > example > WebApp > controller > BrandController` is selected.
- Code Editor (Center):** Shows the `BrandController.java` file with the following code:

```
20 public class BrandController {  
21  
22     BrandService brandService;  
23  
24     no usages 1 SofyaRusyaeva  
25     @GetMapping()  
26     public ResponseEntity<List<Brand>> getBrands() { return ResponseEntity.ok(brandService.findAll()); }  
27  
28     no usages 1 SofyaRusyaeva +1  
29     @PostMapping()  
30     public ResponseEntity<Brand> addBrand(@Valid @RequestBody BrandDto brand) {  
31         return ResponseEntity.status(HttpStatus.CREATED).body(brandService.save(brand));  
32     }  
33  
34     no usages 1 Валерия  
35     @PutMapping("/{brandId}")  
36     public ResponseEntity<Brand> updateBrand(@Valid @RequestBody BrandDto brand, @PathVariable Long brandId) {  
37         return ResponseEntity.ok(brandService.update(brand, brandId));  
38     }  
39  
40     no usages 1 SofyaRusyaeva  
41     @DeleteMapping("/{brandId}")  
42     public ResponseEntity<?> deleteBrand (@PathVariable Long brandId) {  
43         brandService.delete(brandId);  
44         return ResponseEntity.noContent().build();  
45     }  
46 }
```
- Bottom Panel:** Shows the breadcrumb path `WebApp > src > main > java > com > example > WebApp > controller > BrandController` and a progress bar for "VCS Initialization".

Обработка запросов

The screenshot shows an IDE with the following components:

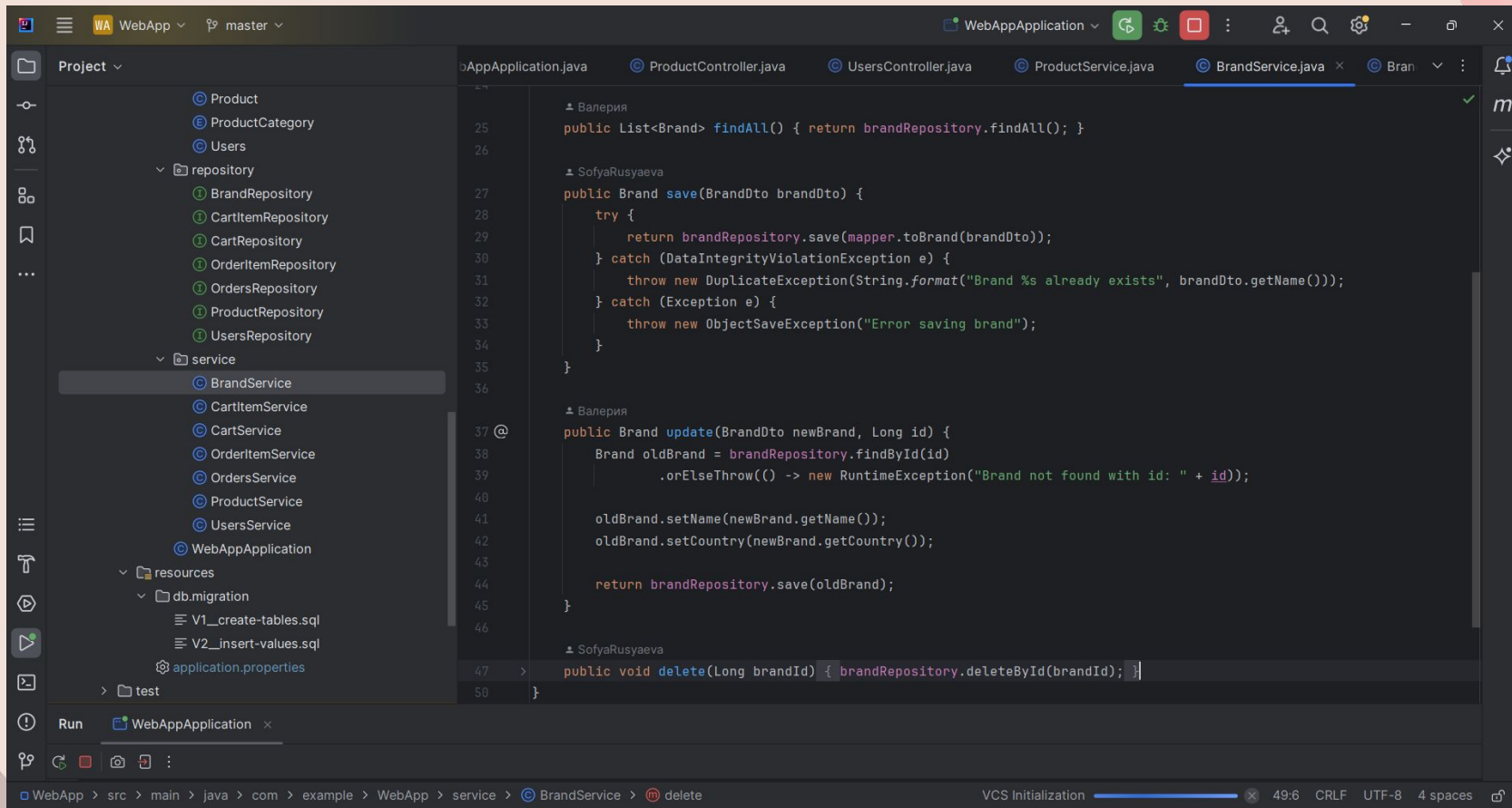
- Project Explorer (Left):** Displays a project structure with packages `Product`, `ProductCategory`, `Users`, `repository` (containing `BrandRepository`, `CartItemRepository`, `CartRepository`, `OrderItemRepository`, `OrdersRepository`, `ProductRepository`, `UsersRepository`), and `service` (containing `BrandService`, `CartItemService`, `CartService`, `OrderItemService`, `OrdersService`, `ProductService`, `UsersService`). The `ProductService` class is selected.
- Code Editor (Center):** Shows the `ProductService.java` file with the `sortAndFilter` method. The code is as follows:

```
1 usage SofiaRusyaeva
public List<Product> sortAndFilter (
    String sortBy, String sortDirection,
    List<ProductCategory> categories,
    BigDecimal minPrice, BigDecimal maxPrice,
    List<String> brandNames) {

    Sort sort = Sort.by(Sort.Direction.ASC, sortBy);
    if ("desc".equalsIgnoreCase(sortDirection))
        sort = Sort.by(Sort.Direction.DESC, sortBy);

    if(categories != null && !categories.isEmpty() && minPrice != null && maxPrice != null && brandNames != null &&
        return productRepository.findByCategoryInAndPriceBetweenAndBrand_NameIn(categories, minPrice, maxPrice, brandNames, sort);
    if(minPrice != null && maxPrice != null && brandNames != null && !brandNames.isEmpty())
        return productRepository.findByPriceBetweenAndBrand_NameIn(minPrice, maxPrice, brandNames, sort);
    if(categories != null && !categories.isEmpty() && brandNames != null && !brandNames.isEmpty())
        return productRepository.findByCategoryInAndBrand_NameIn(categories, brandNames, sort);
    if(categories != null && !categories.isEmpty() && minPrice != null && maxPrice != null)
        return productRepository.findByCategoryInAndPriceBetween(categories, minPrice, maxPrice, sort);
    if(brandNames != null && !brandNames.isEmpty())
        return productRepository.findByBrand_NameIn(brandNames, sort);
    if(categories != null && !categories.isEmpty())
        return productRepository.findByCategoryIn(categories, sort);
    if(minPrice != null && maxPrice != null)
        return productRepository.findByPriceBetween(minPrice, maxPrice, sort);
    return productRepository.findAll();
}
```
- Run Console (Bottom):** Shows the application running with the name `WebAppApplication`.
- Bottom Bar:** Displays the file path `WebApp > src > main > java > com > example > WebApp > service > ProductService`, VCS status `VCS Initialization`, and encoding settings `59:1 CRLF UTF-8 4 spaces`.

Обработка запросов



The screenshot displays an IDE window for a project named 'WebApp'. The 'Project' view on the left shows a package structure with 'repository' and 'service' folders. The 'BrandService.java' file is open in the editor, showing the following code:

```
25 public List<Brand> findAll() { return brandRepository.findAll(); }
26
27
28
29
30
31
32
33
34
35
36
37 @
38 public Brand update(BrandDto newBrand, Long id) {
39     Brand oldBrand = brandRepository.findById(id)
40         .orElseThrow(() -> new RuntimeException("Brand not found with id: " + id));
41
42     oldBrand.setName(newBrand.getName());
43     oldBrand.setCountry(newBrand.getCountry());
44
45     return brandRepository.save(oldBrand);
46 }
47
48
49
50 public void delete(Long brandId) { brandRepository.deleteById(brandId); }
```

The bottom status bar shows the file path: 'WebApp > src > main > java > com > example > WebApp > service > BrandService > delete'.

Тестирование через Postman (GET)

The screenshot displays the Postman application interface. At the top, the navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar is present, and there are buttons for 'Invite', 'Upgrade', and window controls. The left sidebar contains icons for 'Collections', 'Environments', 'Flows', 'History', and a workspace icon.

The main area shows a GET request to `http://localhost:8080/api/shop/cart`. The 'Send' button is visible. Below the request, the 'Params' tab is active, showing 'Query Params' with a table:

Key	Value	Description
Key	Value	Description

Below the table, the 'Body' tab is active, showing a JSON response. The status bar indicates '200 OK' with a response time of 423 ms and a size of 300 B. The JSON response is:

```
[
  {
    "cartId": 1,
    "totalPrice": 2000.00,
    "userId": 1
  },
  {
    "cartId": 2,
    "totalPrice": 1500.00,
    "userId": 2
  },
  {
    "cartId": 3,
    "totalPrice": 1800.00,
    "userId": 3
  }
]
```

The bottom status bar includes 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and window controls.

Тестирование через Postman (GET) по id

The screenshot displays the Postman interface for testing an API endpoint. The top bar shows the 'API Network' workspace and a search bar. The left sidebar contains navigation options: Collections, Environments, Flows, and History.

The main area shows a GET request to `http://localhost:8080/api/shop/cart/2`. The 'Params' tab is active, showing a table for Query Params:

Key	Value	Description
Key	Value	Description

The 'Body' tab is also active, showing the response in JSON format:

```
{
  "cartId": 2,
  "totalPrice": 1500.00,
  "userId": 2
}
```

The response status is **200 OK** with a response time of 61 ms and a body size of 208 B. The bottom status bar includes links to Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and a help icon.

Тестирование через Postman (GET)

The screenshot displays the Postman application interface. At the top, the navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar is present with the text 'Search Postman'. On the right, there are buttons for 'Invite', 'Upgrade', and a dropdown menu.

The main workspace shows a GET request to the URL `http://localhost:8080/api/shop/product?sortBy=name&sortDirection=desc&brandNames=Clinique`. The request is saved and has a 'Send' button. Below the URL bar, the 'Params' tab is active, showing 'Query Params'.

Key	Value	Description
sortBy	name	
sortDirection	desc	
brandNames	Clinique	

Below the query params, the 'Body' tab is active, showing the response in JSON format. The response is a 200 OK status with a response time of 88 ms and a body size of 376 B. The JSON data is as follows:

```
[
  {
    "productId": 3,
    "name": "mascara",
    "description": "waterproof",
    "category": "makeup",
    "price": 1700.00,
    "brandId": 1
  },
  {
    "productId": 1,
    "name": "lipstick",
    "description": "matt",
    "category": "makeup",
    "price": 2000.00,
    "brandId": 1
  }
]
```

At the bottom of the interface, there is a status bar with 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Тестирование через Postman (POST)

The screenshot displays the Postman application interface. At the top, the navigation bar includes 'Home', 'Workspaces', and 'API Network', along with a search bar and utility buttons like 'Invite', 'Upgrade', and window controls. The left sidebar contains icons for Collections, Environments, Flows, and History. The main workspace is titled 'POST http://localhost:8080/api/shop/users'. Below the URL bar, tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Scripts', and 'Settings' are visible. The 'Body' tab is active, showing a raw JSON payload. The 'Send' button is prominently displayed. On the right, 'Cookies' and 'Beautiful' links are present. The bottom section shows the response under the 'Body' tab, indicating a '201 Created' status with a response time of 357 ms and a size of 319 B. The response is a JSON object with user details. The footer contains a status bar with 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Overview **POST** http://localhost:8080/api/shop/users

Save Share

POST http://localhost:8080/api/shop/users

Send

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "userName": "Emma",
3   "email": "Emma@ssau.ru",
4   "password": "Igvjoeirjg",
5   "phone": "+79282348765"
6 }
7
```

Body Cookies Headers (5) Test Results

201 Created • 357 ms • 319 B

{ } JSON Preview Visualize

```
1 {
2   "userId": 4,
3   "userName": "Emma",
4   "email": "Emma@ssau.ru",
5   "password": "$2a$10$3Jv11InKFIZQcpNZwNBmtOV3DGIkVUEobRDlWN4E3SYUKgzHMTUW",
6   "phone": "+79282348765"
7 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash

Тестирование через Postman (PUT)

The screenshot displays the Postman application interface for testing a PUT request. The top navigation bar includes links for Home, Workspaces, and API Network, along with a search bar and utility buttons like 'Invite', 'Upgrade', and window controls. The left sidebar shows the 'Collections' and 'Environments' panels. The main workspace is titled 'Overview' and shows the active request: `PUT http://localhost:8080/api/shop/users/4`. Below the URL bar, the 'Body' tab is selected, showing a JSON payload:

```
{  "userName": "Emma",  "email": "Emma@mail.ru",  "password": "Igvjoerjg",  "phone": "+79282348765"}
```

. The 'Send' button is visible to the right. Below the request, the 'Test Results' tab shows a successful response: `200 OK` with a response time of 226 ms and a body size of 314 B. The response body is displayed in JSON format:

```
{  "userId": 4,  "userName": "Emma",  "email": "Emma@mail.ru",  "password": "$2a$10$.xfA.21/3VItmW5Y1Ze3l.p0u8aFS42sJw97bPFxtn3pE93tV8zHC",  "phone": "+79282348765"}
```

. The bottom status bar shows 'Online' and provides shortcuts for 'Find and replace' and 'Console'.

Тестирование через Postman (DELETE)

The screenshot displays the Postman interface for testing an API endpoint. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar is present with the text 'Search Postman'. The main workspace shows a DELETE request to `http://localhost:8080/api/shop/users/4`. The request body is a JSON object:

```
1 {
2   "username": "Emma",
3   "email": "Emma@mail.ru",
4   "password": "Igvjoerjg",
5   "phone": "+79282348765"
6 }
7
```

The response is `204 No Content` with a status of 300 ms and 112 B. The response body is empty. The bottom status bar shows 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.