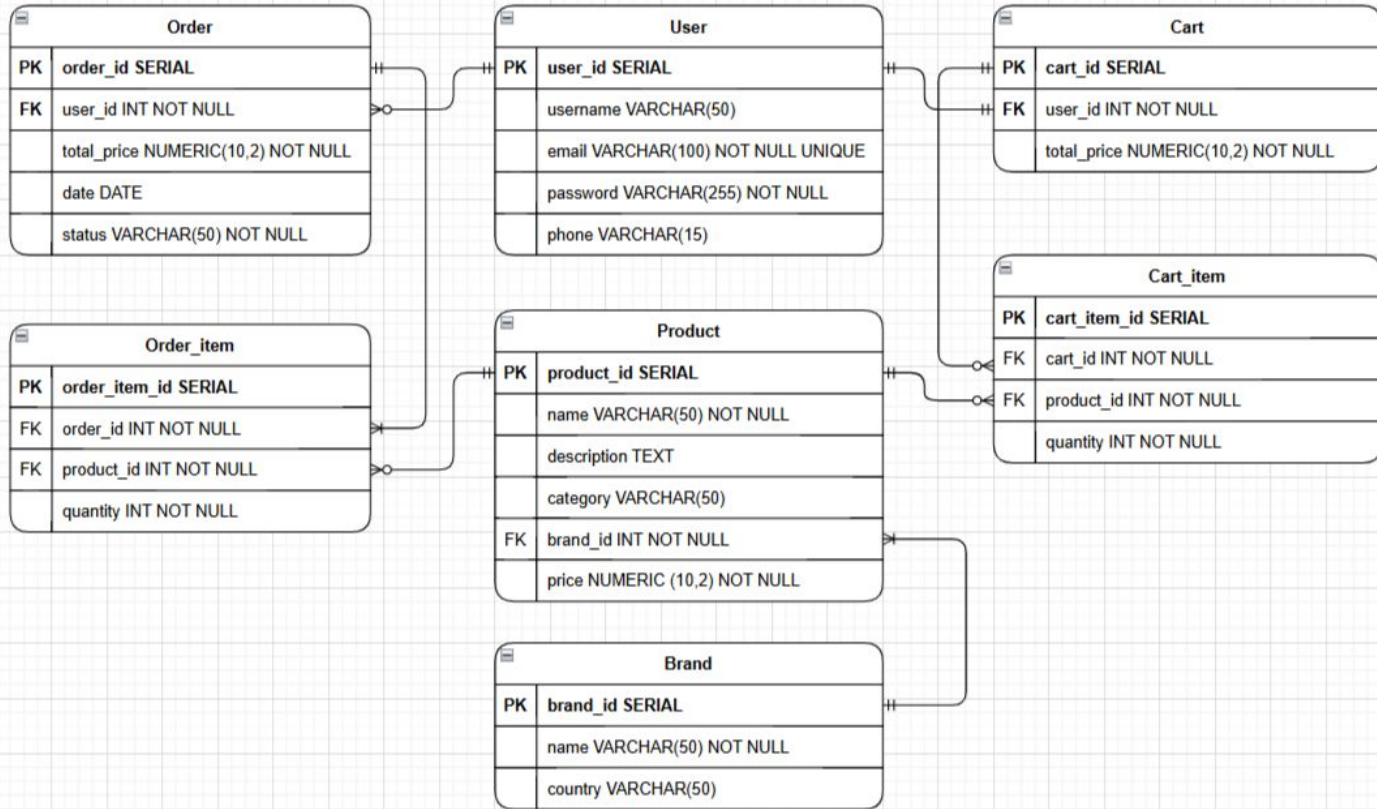


Интернет магазин косметики “Silver Pear”

Выполнили Жданова Валерия и
Русяева Софья 6301-010302D

Схема БД



Классы, соответствующие сущностям БД

The screenshot shows an IDE with a project structure on the left and a Java class file on the right. The project structure is as follows:

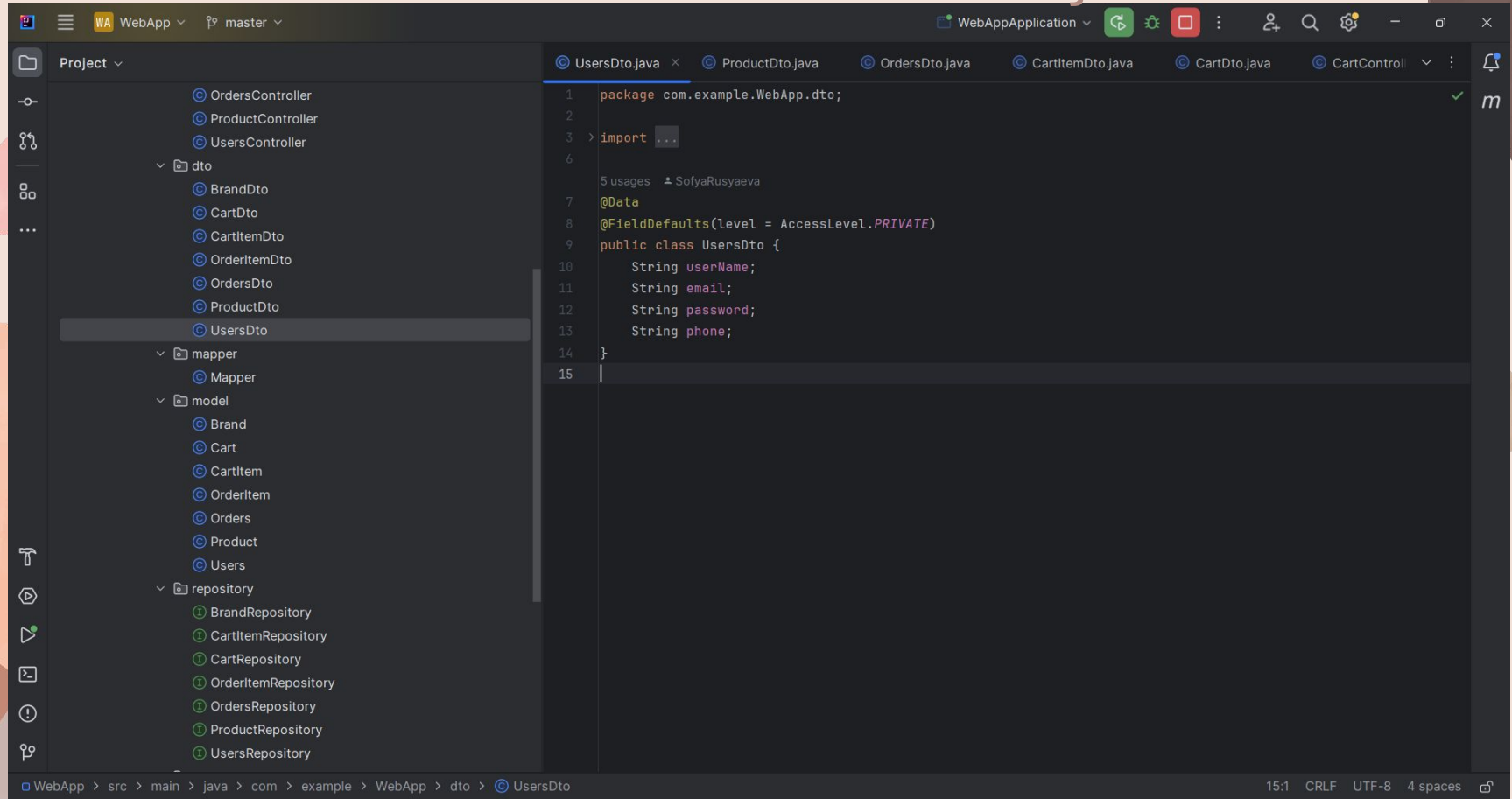
- Project
 - CartDto
 - CartItemDto
 - OrderItemDto
 - OrdersDto
 - ProductDto
 - UsersDto
 - mapper
 - Mapper
 - model
 - Brand
 - Cart
 - CartItem
 - OrderItem
 - Orders
 - Product
 - Users
 - repository
 - BrandRepository
 - CartItemRepository
 - CartRepository
 - OrderItemRepository
 - OrdersRepository
 - ProductRepository
 - UsersRepository
 - service
 - WebAppApplication
 - resources
 - db.migration
 - application.properties

The right pane shows the `Users.java` file with the following code:

```
1 package com.example.WebApp.model;
2
3 import ...
4
5 20 usages 1 SofiaRusyaeva +1
6
7 @Data
8 @Entity
9 @Table(name = "users")
10 @FieldDefaults(level = AccessLevel.PRIVATE)
11 public class Users {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     Long userId;
15
16     @NotBlank(message = "Username can't be blank")
17     String userName;
18
19     @NotBlank(message = "Email can't be blank")
20     @Column(unique = true)
21     String email;
22
23     @NotBlank(message = "Password can't be blank")
24     String password;
25
26     String phone;
27
28     @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
29     @JsonIgnore
30     List<Orders> orders;
31
32     @OneToOne(mappedBy = "user", cascade = CascadeType.ALL)
33     @JsonIgnore
34 }
```

The status bar at the bottom shows the file path: `WebApp > src > main > java > com > example > WebApp > model > Users` and the encoding: `2:1 CRLF UTF-8 4 sp`.

Dto (Data transfer object)



Сервисы

The screenshot displays an IDE window for a project named 'WebApp'. The left sidebar shows the project structure with the following components:

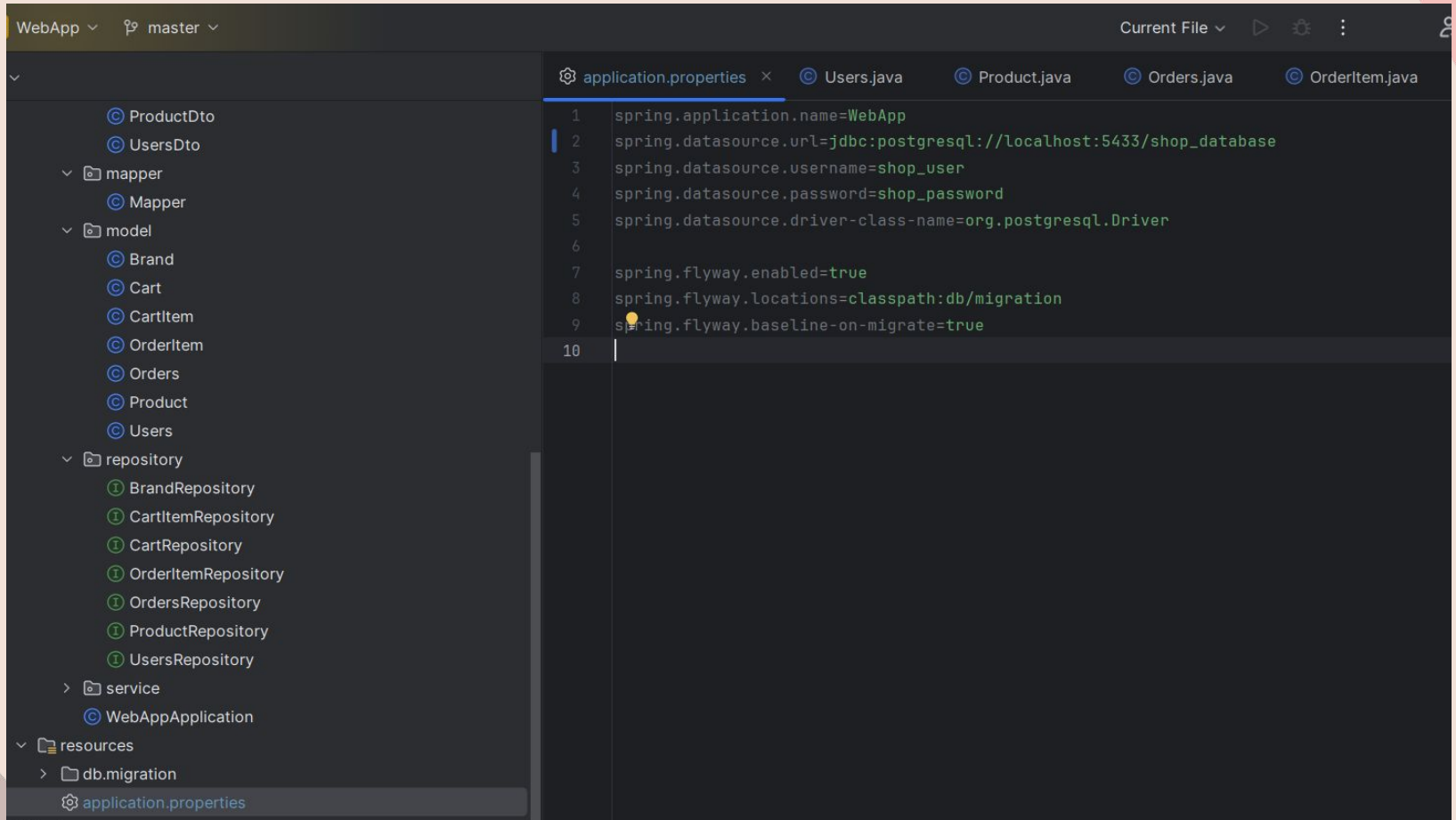
- BrandRepository
- CartItemRepository
- CartRepository
- OrderItemRepository
- OrdersRepository
- ProductRepository
- UsersRepository
- service
 - BrandService
 - CartItemService
 - CartService
 - OrderItemService
 - OrdersService
 - ProductService
 - UserService**
- WebAppApplication
 - resources
 - db.migration
 - application.properties
 - test
 - target**
 - .gitattributes
 - .gitignore
 - mvnw
 - mvnw.cmd
 - pom.xml
 - README.md
 - External Libraries
 - Scratches and Consoles

The main editor area shows the code for 'UserService.java' with the following content:

```
1 package com.example.WebApp.service;
2
3 > import ...
15
16 @Service
17 @RequiredArgsConstructor
18 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
19 public class UserService {
20
21     UsersRepository usersRepository;
22     CartRepository cartRepository;
23     Mapper mapper;
24     PasswordEncoder passwordEncoder;
25
26     Валерия
27     public List<Users> findAll() { return usersRepository.findAll(); }
28
29     SofyaRusyaeva
30     public Users save(UsersDto userDto) {
31         Users user = mapper.toUsers(userDto);
32         user.setPassword(passwordEncoder.encode(user.getPassword()));
33         return usersRepository.save(user);
34     }
35
36     SofyaRusyaeva
37     public void delete(Long userId) { usersRepository.deleteById(userId); }
38 }
```

The status bar at the bottom indicates the file path: 'WebApp > src > main > java > com > example > WebApp > service > UserService' and the encoding: '38:1 CRLF UTF-8 4 spaces'.

application.properties




The screenshot shows an IDE interface with a project named 'WebApp' on the 'master' branch. The left sidebar displays a file tree with the following structure:

- ProductDto
- UsersDto
- mapper
 - Mapper
- model
 - Brand
 - Cart
 - CartItem
 - OrderItem
 - Orders
 - Product
 - Users
- repository
 - BrandRepository
 - CartItemRepository
 - CartRepository
 - OrderItemRepository
 - OrdersRepository
 - ProductRepository
 - UsersRepository
- service
 - WebAppApplication
- resources
 - db.migration
 - application.properties (selected)

The right pane shows the content of the selected 'application.properties' file:

```
1 spring.application.name=WebApp
2 spring.datasource.url=jdbc:postgresql://localhost:5433/shop_database
3 spring.datasource.username=shop_user
4 spring.datasource.password=shop_password
5 spring.datasource.driver-class-name=org.postgresql.Driver
6
7 spring.flyway.enabled=true
8 spring.flyway.locations=classpath:db/migration
9 spring.flyway.baseline-on-migrate=true
10
```

Docker контейнер

 **docker:desktop** PERSONAL

Ctrl+K ? 🔔³ 📦 ⚙️ ☰ V — □ ×

✦ ✦ More Docker. Easy Access. New Streamlined Plans. [Learn more](#)

📦 Containers

🔄 Images

💾 Volumes

🔑 Builds

🌐 Docker Hub

🔍 Docker Scout

⚙️ Extensions

Containers


[Give feedback](#)




View all your running containers and applications. [Learn more](#)

Container CPU usage ⓘ
0.02% / 800% (8 CPUs available)

Container memory usage ⓘ
24.75MB / 3.63GB

[Show charts](#)


 ☒ Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last star	Actions
<input type="checkbox"/>	● shop_container	efa818398101	postgres	5433:5432	0.02%	20 secon	  

Showing 1 item

Walkthroughs

×


 Engine running

||

:

RAM 1.29 GB CPU 1.12% Disk: 2.60 GB used (limit 1006.85 GB)

Terminal

 New version available

Созданная БД, открытая в pgAdmin

pgAdmin 4 interface showing a query execution result.

Explorer:

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (8)
 - brand
 - cart
 - cart_item
 - flyway_schema_history
 - order_item
 - orders
 - product
 - Columns
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
 - users
- Trigger Functions
- Types

Query:

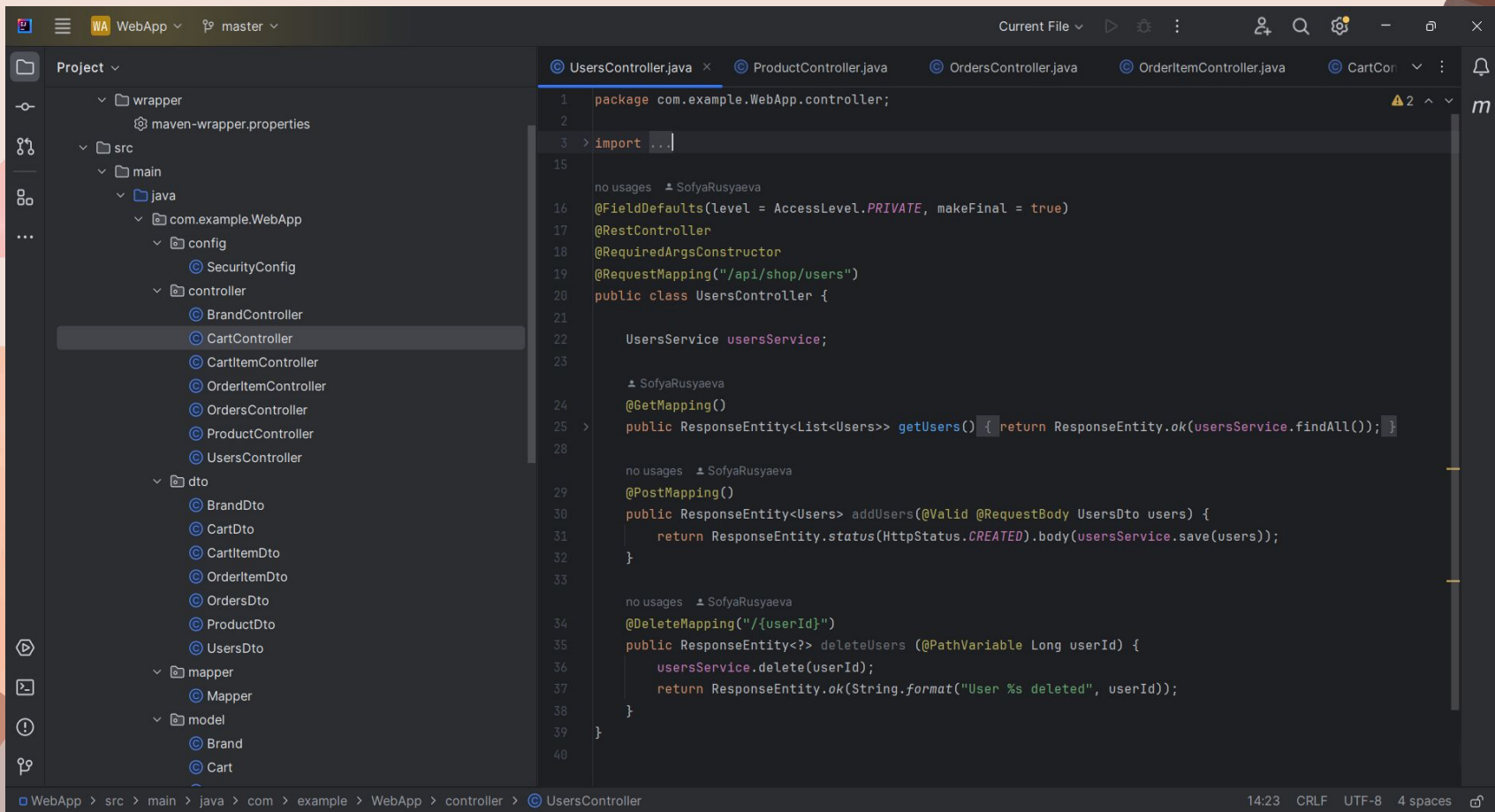
```
1 SELECT * FROM public.users
2 ORDER BY user_id ASC
```

Data Output:

	user_id [PK] bigint	email character varying (255)	password character varying (255)	phone character varying (255)	user_name character varying (255)
1	1	dkhfc@gmail.com	\$2a\$10\$UqrJCIOXviG04TxkwC7qun9yEjW0rzAPeicmfBAUF15cNWmw...	+79274696832	Valeria
2	2	ann@gmail.com	123	1234567890	Ann
3	3	milana@gmail.com	456	0987654321	Milana
4	4	dkhfc@gmail.com	\$2a\$10\$Kajvp1iINrLPi2ejALRSkeGvIaitM9d45cS8OGCLLDNOH008avK1...	+79274696832	Valeria
5	575	35435	56656	77777	663u4

Successfully run. Total query runtime: 222 msec. 5 rows affected.

Контроллеры



Работа запроса GET

The screenshot displays the Postman API client interface. At the top, the navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar for 'Search Postman' is present. The main interface shows a GET request to the URL `http://localhost:8080/api/shop/brand`. The 'Send' button is visible on the right. Below the URL bar, the 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response. The response status is '200 OK' with a response time of '1.81 s' and a size of '359 B'. The JSON body contains an array of two brand objects.

Query Params

Key	Value	Description
Key	Value	Description

Body | Cookies | Headers (5) | Test Results

200 OK • 1.81 s • 359 B

```
{
  "brandId": 54,
  "name": "Name1",
  "country": "Russia"
},
{
  "brandId": 1,
  "name": "name2",
  "country": "USA"
}
```

Работа запроса POST

The screenshot displays the Postman web 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 navigation icons for 'Collections', 'Environments', 'Flows', and 'History'. The main workspace is titled 'POST http://localhost:8080/api/shop/brand'. Below the URL bar, the 'Body' tab is selected, showing a JSON payload:

```
{  "name": "Art Visage",  "country": "Russia"}
```

. The 'Send' button is visible on the right. Below the request editor, the 'Test Results' tab shows the response:

```
{  "brandId": 4,  "name": "Art Visage",  "country": "Russia"}
```

. The status bar at the bottom indicates '201 Created' with a response time of 196 ms and a body size of 221 B. The footer contains status indicators (Online), search, console, and various tool links (Postbot, Runner, Start Proxy, Cookies, Vault, Trash).

Home Workspaces API Network Search Postman

Overview POST http://localhost:8080/api/shop/brand

http://localhost:8080/api/shop/brand

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

Params Authorization Headers (8) Body Scripts Settings

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

```
1 {
2   "name": "Art Visage",
3   "country": "Russia"
4 }
```

Body Cookies Headers (5) Test Results

JSON Preview Visualize

```
1 {
2   "brandId": 4,
3   "name": "Art Visage",
4   "country": "Russia"
5 }
```

201 Created • 196 ms • 221 B

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

Работа запроса DELETE

The screenshot displays the Postman interface for a DELETE request. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The main area shows the request details for 'http://localhost:8080/api/shop/brand/3'. The method is set to 'DELETE'. The request body is a JSON object:

```
{  "name": "Art Visage",  "country": "Russia"}
```

. The response status is '200 OK' with a response time of '173 ms' and a response size of '179 B'. The response body is 'Brand 3 deleted'.

Overview **DEL** http://localhost:8080/ap... +

HTTP http://localhost:8080/api/shop/brand/3 Save Share

DELETE http://localhost:8080/api/shop/brand/3 Send

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

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

```
1 {
2   "name": "Art Visage",
3   "country": "Russia"
4 }
```

Body Cookies Headers (5) Test Results

Raw Preview Visualize

```
1 Brand 3 deleted
```

200 OK • 173 ms • 179 B

Postbot Runner Start Proxy Cookies Vault Trash