МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ОТЧЕТ

по лабораторной работе №2
Дисциплина «Технологии сетевого программирования»

Выполнили студенты
группы 6301-010302D
Жданова В.И.
Русяева С.П.

САМАРА 2025

Для каждой модели были реализованы стандартные CRUD-операции. Например ProductService:

```
@Service
@RequiredArgsConstructor
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class ProductService {

   ProductRepository productRepository;
   BrandRepository brandRepository;
   CartRepository cartRepository;
   CartItemRepository cartItemRepository;
   Mapper mapper;

   public Product findById(Long productId) {
       return productRepository.findById(productId)
               .orElseThrow(() -> new
ObjectNotFoundException(String.format("Product %s not found", productId)));
   }

   public Product save(ProductDto productDto) {
       Brand brand = brandRepository.findById(productDto.getBrandId())
               .orElseThrow(() -> new
ObjectNotFoundException(String.format("Product %s not found",
productDto.getBrandId())));
       Product product = mapper.toProduct(productDto, brand);
       try {
           return productRepository.save(product);
       } catch (Exception e) {
           throw new ObjectSaveException("Error saving brand");
       }
   }

   public void delete(Long productId) {
       productRepository.deleteById(productId);
   }

   public Product update(ProductDto newProduct, Long id) {
       Product oldProduct = productRepository.findById(id)
               .orElseThrow(() -> new
ObjectNotFoundException(String.format("Product %s not found", id)));

       Brand brand = brandRepository.findById(newProduct.getBrandId())
               .orElseThrow(() -> new RuntimeException("Brand not found with id: "
+ newProduct.getBrandId()));

       oldProduct.setName(newProduct.getName());
       if (!Objects.equals(oldProduct.getPrice(), newProduct.getPrice()))
           updateProductPrice(id, newProduct.getPrice(), oldProduct.getPrice());
       oldProduct.setPrice(newProduct.getPrice());
       oldProduct.setBrand(brand);
       oldProduct.setDescription(newProduct.getDescription());
       oldProduct.setCategory(newProduct.getCategory());

       return productRepository.save(oldProduct);
   }
   }
}
```

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

Реализована обработка ошибок для случаев:

- отсутствие ресурса (404)
- некорректные данные запроса (400)
- ошибки сервера (500)

Для сущностей Product и Order добавлена сортировка и фильтрация данных. Например сортировка по возрастанию цены, фильтры: дата заказа между 01-01-2023 и 01-01-2025, статус заказа in_progress, цена в диапазоне от 0 до 3000

GET
`http://localhost:8080/api/shop/orders?sortBy=totalPrice&sortDirection=ASC&startDate=2023-01-01&endDate=2025-01-01&statuses=in_progress&minPrice=0&maxPrice=3000`

Метод реализующий сортировку и фильтрацию:

```
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 && !brandNames.isEmpty())
           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();
   }
```