Exercise 02 - Decomposing the ProductSrv

Required Services

The following services are involved and have to be started before the final exercise validation:

- CustomerSrv (`http://localhost:8000`)
- NotificationSrv (`http://localhost:8010`)
- OrderSrv (`http://localhost:8030`)
- ProductSrv (`http://localhost:8050`)
- CategorySrv (`http://localhost:8060`)
- WarehouseSrv (`http://localhost:8070`)
- WebUI (`http://localhost:5000`)

Description

The `ProductSrv` has grown over time and is now fairly large compared to the other services. It is responsible for several different entities, namely products, product categories, and the available amount of product copies in the warehouse. The following resources are currently provided:

```
# Product category resources
GET     /categories (experiment.webshop.products.resources.ProductResource)
POST    /categories (experiment.webshop.products.resources.ProductResource)
DELETE  /categories/{id} (experiment.webshop.products.resources.ProductResource)
GET     /categories/{id} (experiment.webshop.products.resources.ProductResource)
PUT     /categories/{id} (experiment.webshop.products.resources.ProductResource)

# Product resources
GET     /products (experiment.webshop.products.resources.ProductResource)
POST    /products (experiment.webshop.products.resources.ProductResource)
DELETE  /products/{id} (experiment.webshop.products.resources.ProductResource)
GET     /products/{id} (experiment.webshop.products.resources.ProductResource)
PUT     /products/{id} (experiment.webshop.products.resources.ProductResource)

# Warehouse resources for product availability
GET     /products/{id}/availability
(experiment.webshop.products.resources.ProductResource)
PUT     /products/{id}/availability
(experiment.webshop.products.resources.ProductResource)
```

The lead developer has decided to split up the `ProductSrv` to increase maintainability and scaling efficiency. Two new services will be created: A `CategorySrv` handling product categories and a `WarehouseSrv` responsible for product availability. The CRUD operations related to products will remain in the `ProductSrv`. Runnable skeleton projects for the new services have already been created, they just provide no resources yet.

Tasks

1. **Move the product category related functionality.** Move all functionality related to product categories (see above) from the `ProductSrv` to the new `CategorySrv`. It already has a resource class (`experiment.webshop.categories.resources.ProductCategoryResource`) and a repository class (`experiment.webshop.categories.db.ProductCategoryRepository`) that have to be extended. All necessary model classes are already present in `experiment.webshop.categories.api` (you simply have to adjust the import statements for the copied lines).
2. **Move the product availability related functionality.** Move all functionality related to the warehouse product availability (see above) from the `ProductSrv` to the new `WarehouseSrv`. It already has a resource class (`experiment.webshop.warehouse.resources.WarehouseResource`) and a repository class (`experiment.webshop.warehouse.db.WarehouseRepository`) that have to be extended. All necessary model classes are already present in `experiment.webshop.warehouse.api` (you simply have to adjust the import statements for the copied lines).
3. **Fix the service consumers of the moved functionality.** The old `ProductSrv` had two consumers that used its resources, namely the `OrderSrv` and the `WebUI`. These two consumers have to be adjusted with the new base URLs to reflect the changes of the service decomposition. The `OrderSrv` change has to be performed in the `experiment.webshop.orders.resources.OrderResource` class (adjust `PRODUCT_AVAILABILITY_CHECK_ENDPOINT`). The `WebUI` changes have to be performed in `app/main.js`: The retrieving of all categories has to be fixed in the `created()` method while checking product availability has to be fixed in `checkProductAvailability()`. The variables `categorySrvEndpoint` and `warehouseSrvEndpoint` have already been configured and can be used.

Validation

When you are finished with all tasks, make sure all required services (see Required Services) and the exercise validation UI is up and running (if not, execute `exercise-validation/build-and-run-validation-ui.sh`) and then navigate to `http://localhost:5001` (**it is important to start from this page, because it will determine which version you are working on**). Click on `Exercise 02` and then on `Start Validation`. If every check is successful (`status: true`), pause your stopwatch and notify an experiment admin for the manual validation part and to write down your time.