Implementation info

The project was implemented using:

- Java 10
- IntelliJ IDEA
- Spring 5, including:
  - Spring Boot
  - Spring Data JPA
  - Spring Security
  - Lombok
- Hibernate 5
- In-memory database: H2, access via http://localhost:8080/h2-console

REST API is secured with HTTP Basic Authentication.

The filter used to find products is built dynamically.

A product URL is available to the customer in the search result only if it is ordered.

Database H2 is initialized in the class SpaceAgencyDataHubApplication with CommandLineRunner. Sample 2 missions, 3 products and 2 orders are added.

Exceptions are handled globally in the class GlobalExceptionHandler.

Models accessible through the service.:

- Mission:
  - name
  - imageryType
  - startDate
  - finishDate
- Product:
  - missionName
  - acquisitionDate
  - footprint
  - price
  - url
- ProductFootprint:
  - startCoordinateLatitude
  - startCoordinateLongitude
  - endCoordinateLatitude
  - endCoordinateLongitude
- ProductOrder – POST /api/orders:
  - productIds (ordered products id list)
- ProductOrder – GET /api/orders:
  - placedOn
  - productIds (ordered products id list)

Authentication, authorization, API

Two users are authenticated:

- user: ContentManager, pass: contentmanager, role: CONTENT_MANAGER – can access:
    - /api/missions – possible operations (CRUD):
        - GET /api/missions
        - GET /api/missions/1
        - POST /api/missions
        - PUT /api/missions
        - DELETE /api/missions/1
    - /api/products – possible operations (CRD):
        - GET /api/products
        - GET /api/products/1
        - GET /api/products/most-ordered (the product list is returned, in order from most ordered to least ordered)
        - POST /api/products
        - DELETE /api/products
- user: Customer, pass: customer, role: CUSTOMER – can access:
    - /api/products/find
        - GET /api/products/find – returns the list of all products (no filtering)
        - GET /api/products/find?parametry – Dynamically built filter. It is possible every combination of the parameters with one exception: latitude and longitude must be specified together. If afterDate is earlier than beforeDate, there are returned products with acquisition date between these dates.
            - missionName
            - imageryType (PANCHROMATIC, MULTISPECTRAL or HYPERSPECTRAL)
            - beforeDate
            - afterDate
            - latitude
            - longitude
    - /api/orders – possible operations:
        - GET /api/orders – is returned the history of orders in order from newest to oldest
        - POST /api/orders – ordering products by specifying their ids

Tests

There are implemented:

- Integration tests of the controller classes (http query –> controller –> response returned by the controller):
    - MissionController
    - ProductController
    - ProductOrderController
- Unit tests of the service classes:
    - MissionService
    - ProductService

    - ProductOrderService
- Integration test – from http request up to test in-memory database:
    - Mission
    - Product
    - ProductOrder
    - Product – method findProduct of the class ProductController (testing dynamic filters)