
TP N°3.2:

**Monitoring des microservices :
Spring Cloud Actuator**

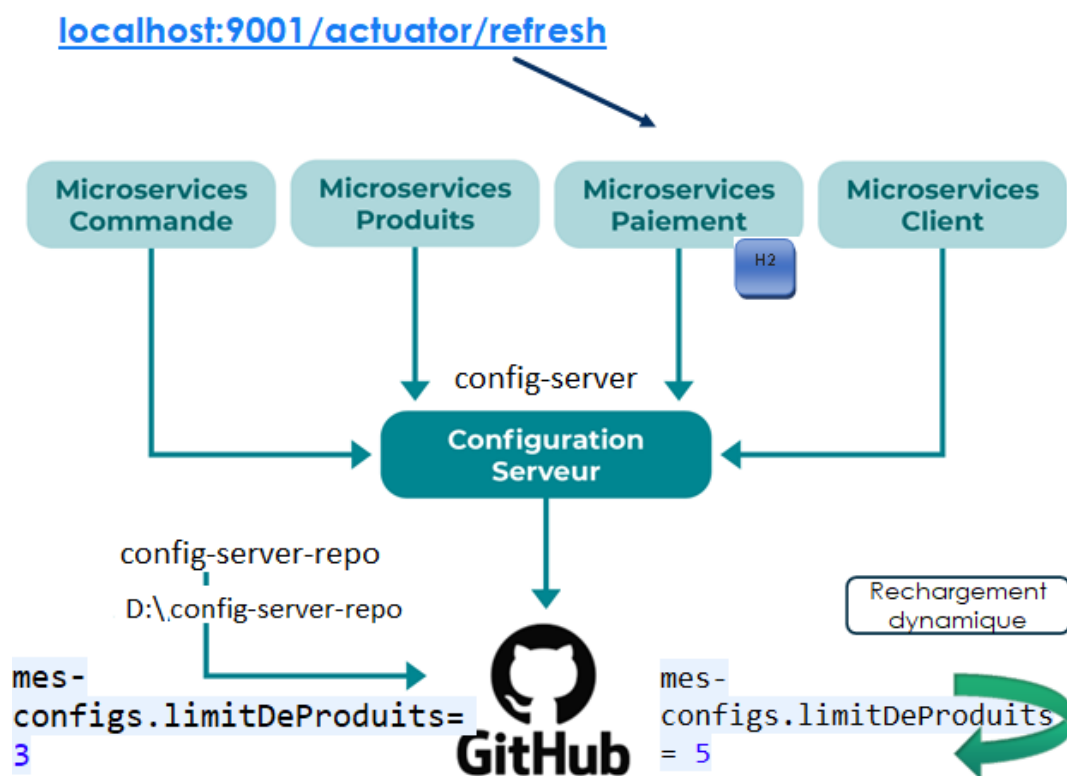
1. Prérequis

- TP3.1 Spring Cloud Config : ce n'est pas nécessaire d'avoir le Spring Cloud Config Server
- POSTMAN ou un autre outil pour tester les méthodes POST, PUT et DELETE.

2. Objectifs

1. Développer un microservice « Monitorable » grâce à Spring **Actuator**
2. Afficher les services offerts par Spring Actuator
3. Mise en œuvre du service Spring Actuator
4. Rechargement dynamique de la configuration d'un microservice : **@RefreshScope**
5. Etat de santé du microservice avec le service « **health** »
6. Personnalisation du service **health** avec l'interface « **HealthIndicator** »

3. Architecture de mise en œuvre

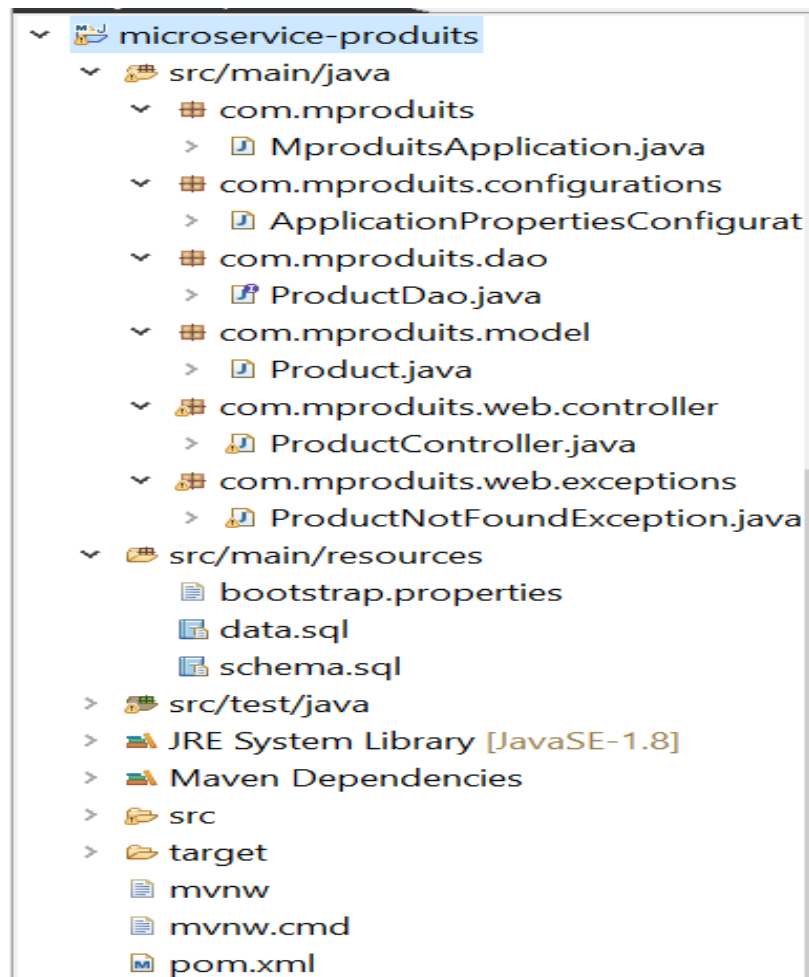


4. Démarche de développement de l'application :

1. Use case : Le microservice « Produits » permet de réaliser les opération CRUD et permet en outre d'afficher la liste des produits avec une taille limite configurée au niveau du fichier de configuration : « `mes-configs.limitDeProduits= 4` »
2. On va modifier cette propriété à « 5 » puis appliquer le service Actuator « `refresh` » du microservice-produit.
3. Vérifier que la configuration a été prise en charge de manière dynamique (à chaud).

5. Développement du microservice « Produit »

- a. Se baser sur le projet microservice-produit dans sa version de configuration locale ou bien avec Spring cloud Config Server.
- b. Dans ce TP on va se baser sur la version avec Spring Cloud Config Server



- c. « `bootstrap.properties` » du projet du microservice-produits:

```
spring.application.name=microservice-produits
#URL de Spring Cloud Config Server
spring.cloud.config.uri=http://localhost:9101
```

d. « microservice-produits.properties »

```
#Configurations H2
spring.jpa.show-sql=true
spring.h2.console.enabled=true
spring.datasource.sql-script-encoding=UTF-8
#Les configurations exetrenalisés
mes-configs.limitDeProduits= 4
#Configuration Actuator
management.endpoints.web.exposure.include=*
#management.endpoints.web.exposure.include=refresh
```

- La configuration d'Actuator se fait via la ligne :
`management.endpoints.web.exposure.include=*` et permet d'exposer l'ensemble des services Actuator disponibles (~24).
- «=refresh» Indique aux beans de se rafraichir à chaque fois qu'un événement Refresh est lancé
- Actuator expose des endpoints/API qui fournissent des données sur des aspects du microservice
- Si on utilise Spring Cloud Config : Ne pas oublier de synchroniser le repository local avec Github.

e. Vérifier que le starter « spring-boot-starter-actuator » existe au niveau du fichier pom.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mproduits</groupId>
  <artifactId>mproduits</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>mproduits</name>
  <description>Microservice de gestion des produits</description>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.16</version>
    <relativePath/>
  </parent>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
```

```

        <spring-cloud.version>2021.0.8</spring-cloud.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-
processor</artifactId>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>

<!-- Add the following dependency to avoid : No spring.config.import property
has been defined -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-bootstrap</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
            </dependency>
        </dependencies>
    </dependencyManagement>

```

```

        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>

<build> <plugins> <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
</plugin></plugins></build>
<repositories>
    <repository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/milestone</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
</repositories>
</project>

```

```

package com.mproduits.web.controller;

import com.mproduits.configurations.ApplicationPropertiesConfiguration;
import com.mproduits.dao.ProductDao;
import com.mproduits.model.Product;
import com.mproduits.web.exceptions.ProductNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.actuate.health.Health;
import org.springframework.boot.actuate.health.HealthIndicator;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
import java.util.Optional;

@RestController
public class ProductController implements HealthIndicator {

    @Autowired
    ProductDao productDao;

    @Autowired
    ApplicationPropertiesConfiguration appProperties;

    // Affiche la liste de tous les produits disponibles
    @GetMapping(value = "/Produits")
    public List<Product> listeDesProduits() {

        System.out.println(" ***** ProductController listeDesProduits() ");
        List<Product> products = productDao.findAll();

        if (products.isEmpty())
            throw new ProductNotFoundException("Aucun produit n'est disponible à la vente");

        List<Product> listeLimitee = products.subList(0,
appProperties.getLimitDeProduits());

        return listeLimitee;
    }
}

```

```

    }

    // Récupérer un produit par son id
    @GetMapping(value = "/Produits/{id}")
    public Optional<Product> recupererUnProduit(@PathVariable int id) {
        System.out.println(" ***** ProductController
recupererUnProduit(@PathVariable int id) ");

        Optional<Product> product = productDao.findById(id);

        if (!product.isPresent())
            throw new ProductNotFoundException("Le produit correspondant à
l'id " + id + " n'existe pas");

        return product;
    }

    @Override
    public Health health() {

        System.out.println("***** Actuator : ProductController health() ");

        List<Product> products = productDao.findAll();
        if (products.isEmpty()) {
            return Health.down().build();
        }
        return Health.up().build();
    }
}

```

f. <http://localhost:9001/actuator>

Dans le cas où : `management.endpoints.web.exposure.include=*`

```

← ↻ ⓘ localhost:9001/actuator
1 {
2   "_links": {
3     "self": {
4       "href": "http://localhost:9001/actuator",
5       "templated": false
6     }
7   },
8   "beans": {
9     "href": "http://localhost:9001/actuator/beans",
10    "templated": false
11  },
12  "caches-cache": {
13    "href": "http://localhost:9001/actuator/caches/{cache}",
14    "templated": true
15  },
16  "caches": {
17    "href": "http://localhost:9001/actuator/caches",
18    "templated": false
19  },
20  "health": {
21    "href": "http://localhost:9001/actuator/health",
22    "templated": false
23  },
24  "health-path": {
25    "href": "http://localhost:9001/actuator/health/{*path}",
26    "templated": true
27  },
28  "info": {
29    "href": "http://localhost:9001/actuator/info",
30    "templated": false
31  },
32  "conditions": {
33    "href": "http://localhost:9001/actuator/conditions",
34    "templated": false
35  },
36  "configprops": {
37    "href": "http://localhost:9001/actuator/configprops",
38    "templated": false
39  },
40  "configprops-prefix": {
41    "href": "http://localhost:9001/actuator/configprops/{prefix}",
42    "templated": true
43  },
44  "env": {
45    "href": "http://localhost:9001/actuator/env",
46    "templated": false
47  },
48  "env-toMatch": {
49    "href": "http://localhost:9001/actuator/env/{toMatch}",
50    "templated": true
51  },
52  },
53  },
54  },
55  },
56  },
57  },
58  },
59  },
60  },
61  },
62  },
63  },
64  },
65  },
66  },
67  },
68  },
69  },
70  },
71  },
72  },
73  },
74  },
75  },
76  },
77  },
78  },
79  },
80  },
81  },
82  },
83  },
84  },
85  },
86  },
87  },
88  },
89  },
90  },
91  },
92  },
93  },
94  },
95  },
96  },

```

```

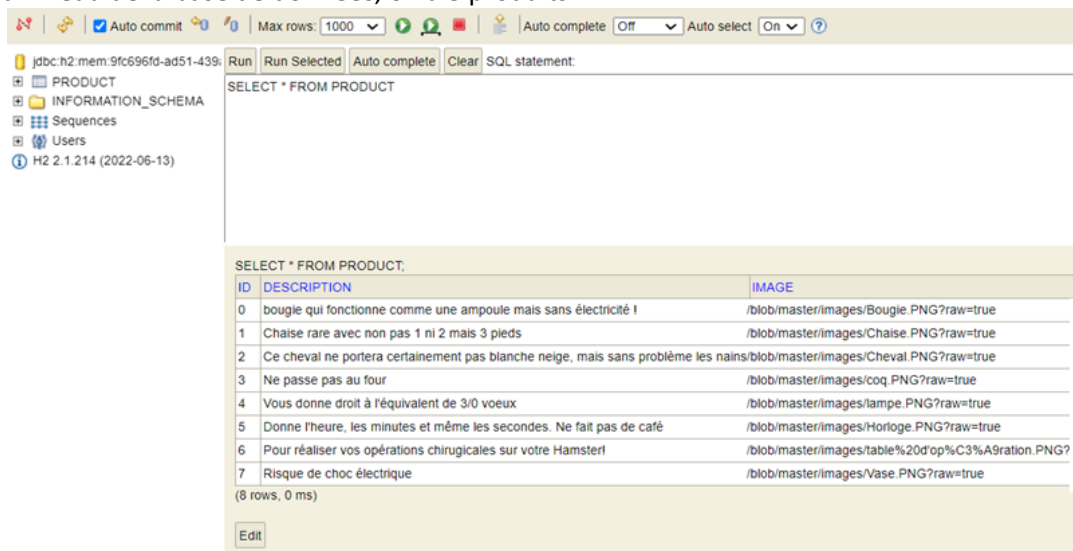
51   "loggers": {
52     "href": "http://localhost:9001/actuator/loggers",
53     "templated": false
54   },
55   "loggers-name": {
56     "href": "http://localhost:9001/actuator/loggers/{name}",
57     "templated": true
58   },
59   "heapdump": {
60     "href": "http://localhost:9001/actuator/heapdump",
61     "templated": false
62   },
63   "threaddump": {
64     "href": "http://localhost:9001/actuator/threaddump",
65     "templated": false
66   },
67   "metrics": {
68     "href": "http://localhost:9001/actuator/metrics",
69     "templated": false
70   },
71   "metrics-requiredMetricName": {
72     "href": "http://localhost:9001/actuator/metrics/{requiredMetricName}",
73     "templated": true
74   },
75   "scheduledtasks": {
76     "href": "http://localhost:9001/actuator/scheduledtasks",
77     "templated": false
78   },
79   "mappings": {
80     "href": "http://localhost:9001/actuator/mappings",
81     "templated": false
82   },
83   "refresh": {
84     "href": "http://localhost:9001/actuator/refresh",
85     "templated": false
86   },
87   "features": {
88     "href": "http://localhost:9001/actuator/features",
89     "templated": false
90   },
91   "serviceregistry": {
92     "href": "http://localhost:9001/actuator/serviceregistry",
93     "templated": false
94   }
95 }
96 }

```


➔ Dans le cas où : `management.endpoints.web.exposure.include=*`

```
localhost:9001/actuator
1 {
2   "_links": {
3     "self": {
4       "href": "http://localhost:9001/actuator",
5       "templated": false
6     },
7     "refresh": {
8       "href": "http://localhost:9001/actuator/refresh",
9       "templated": false
10    }
11  }
12 }
```

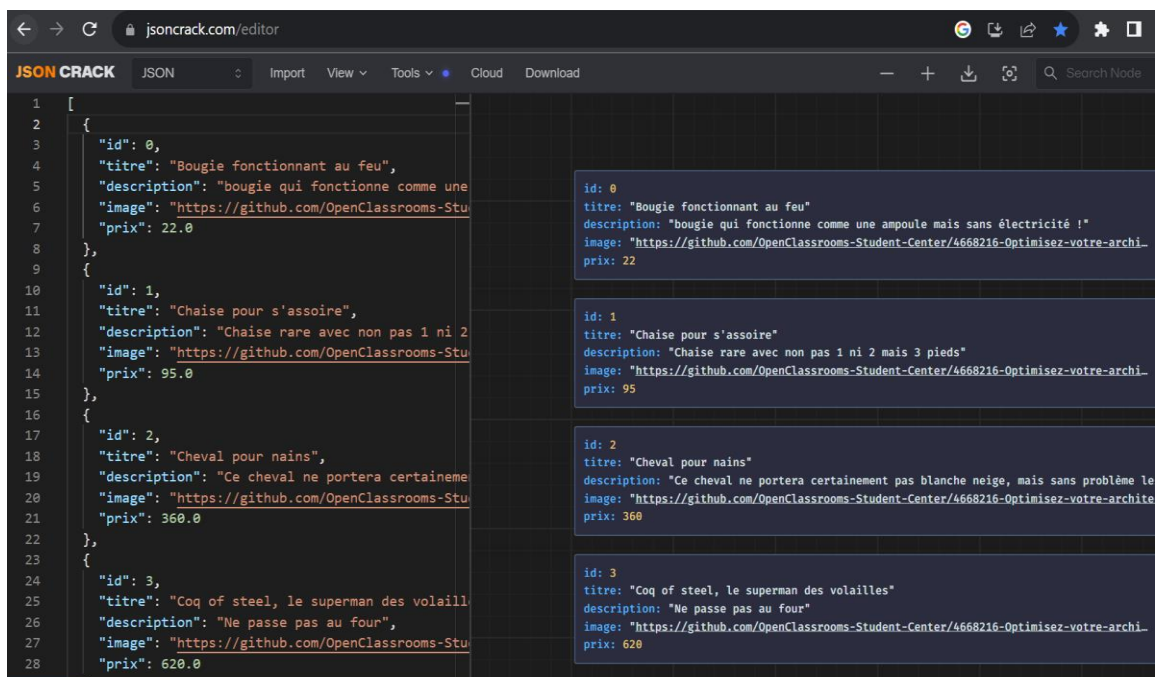
- g. Vérifier le bon fonctionnement du MS-Produits :
Au niveau de la base de données, on a 8 produits :



The screenshot shows a database client interface with a table named 'PRODUCT'. The table has three columns: ID, DESCRIPTION, and IMAGE. There are 8 rows of data. The interface includes a sidebar with a tree view showing the database structure, a top bar with various controls like 'Auto commit', 'Max rows', and 'SQL statement', and a bottom bar with 'Run', 'Run Selected', 'Auto complete', 'Clear', and 'SQL statement' buttons.

ID	DESCRIPTION	IMAGE
0	bougie qui fonctionne comme une ampoule mais sans électricité !	/blob/master/images/Bougie.PNG?raw=true
1	Chaise rare avec non pas 1 ni 2 mais 3 pieds	/blob/master/images/Chaise.PNG?raw=true
2	Ce cheval ne portera certainement pas blanche neige, mais sans problème les nains	/blob/master/images/Cheval.PNG?raw=true
3	Ne passe pas au four	/blob/master/images/coq.PNG?raw=true
4	Vous donne droit à l'équivalent de 3/0 voeux	/blob/master/images/lampe.PNG?raw=true
5	Donne l'heure, les minutes et même les secondes. Ne fait pas de café	/blob/master/images/Horloge.PNG?raw=true
6	Pour réaliser vos opérations chirurgicales sur votre Hamster!	/blob/master/images/table%20d'op%C3%A9ration.PNG?
7	Risque de choc électrique	/blob/master/images/Vase.PNG?raw=true

➔ <http://localhost:9001/Produits> ➔ Affichage de 4 produits



The screenshot shows a JSON editor interface with a list of 4 products. The JSON is displayed on the left, and the corresponding data is shown in a table on the right. The table has columns for id, titre, description, image, and prix. The data is as follows:

id	titre	description	image	prix
0	"Bougie fonctionnant au feu"	"bougie qui fonctionne comme une ampoule mais sans électricité !"	"https://github.com/OpenClassrooms-Student-Center/4668216-Optimisez-votre-archi-	22
1	"Chaise pour s'asseoir"	"Chaise rare avec non pas 1 ni 2 mais 3 pieds"	"https://github.com/OpenClassrooms-Student-Center/4668216-Optimisez-votre-archi-	95
2	"Cheval pour nains"	"Ce cheval ne portera certainement pas blanche neige, mais sans problème le	"https://github.com/OpenClassrooms-Student-Center/4668216-Optimisez-votre-archi-	360
3	"Coq of steel, le superman des volailles"	"Ne passe pas au four"	"https://github.com/OpenClassrooms-Student-Center/4668216-Optimisez-votre-archi-	620

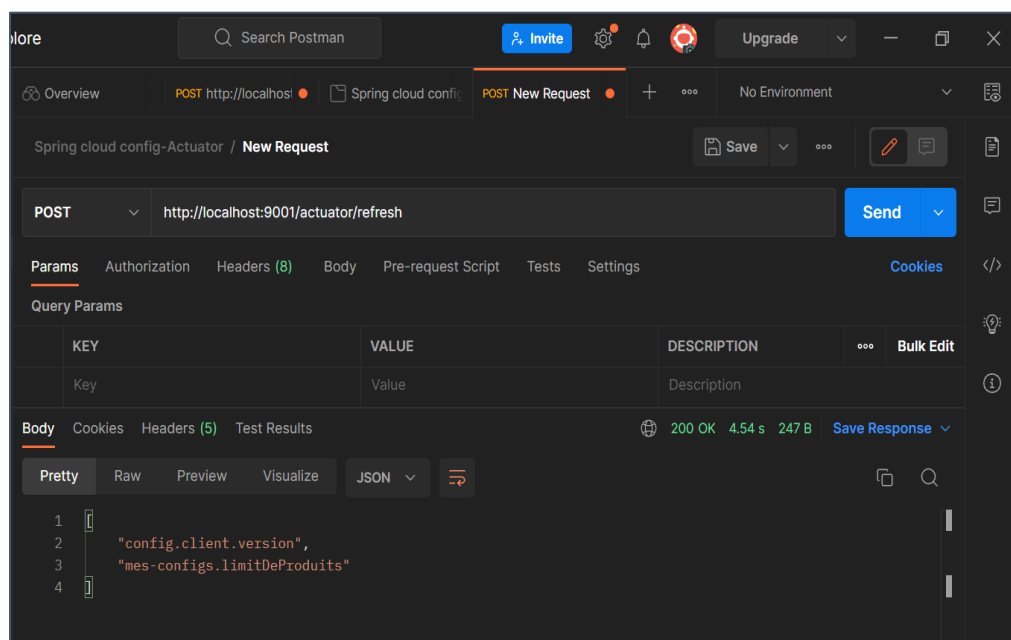
6. Rechargement de la configuration à chaud

- a. Modifier la propriété `mes-configs.limitDeProduits= 5`: augmenter le nombre à 5 (Commit puis push) → rien ne se passe

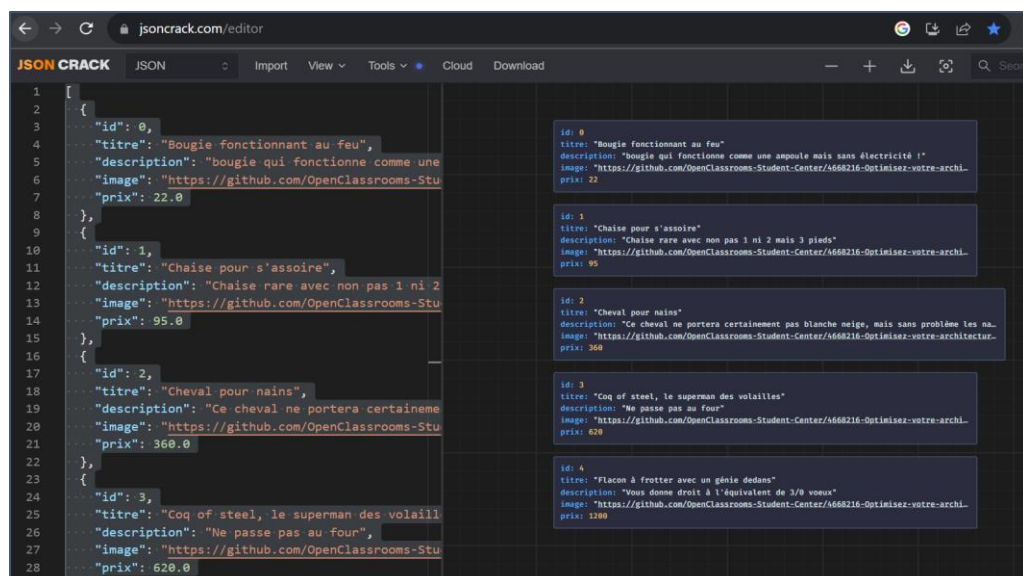
Déclenchez un événement Refresh en envoyant une requête POST

<http://localhost:9001/actuator/refresh> via Postman qui indique exactement les propriétés qui ont été modifiées depuis le dernier commit

- b. Remarque Request method 'GET' not supported] : il faut utiliser la méthode POST au lieu de GET



- c. <http://localhost:9001/Produits> → Affichage de 5 produits

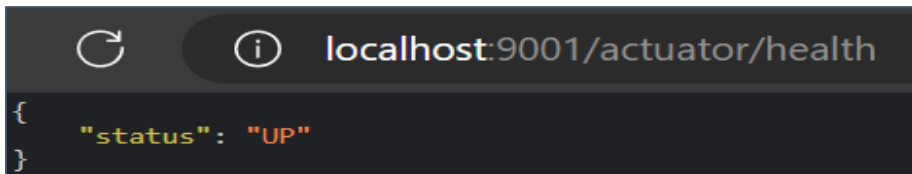


7. Etat de santé du microservice : service « health »

- a. Grâce à l'interface « `HealthIndicator` » :

```
@RestController
public class ProductController implements HealthIndicator
```

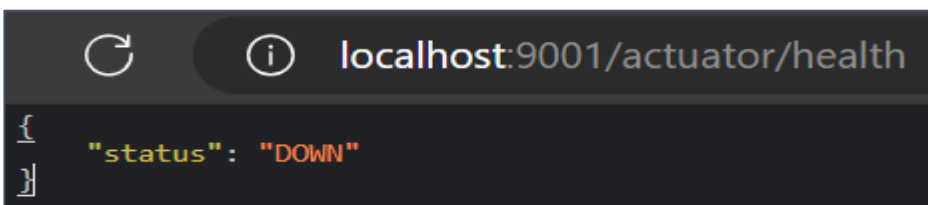
- b. On peut tester l'état de santé du MS : localhost:9001/actuator/health



```
{
  "status": "UP"
}
```

- c. Personnalisation du comportement de « `HealthIndicator` » :

```
@Override
public Health health()
{
    List<Product> products = productDao.findAll();
    if(products.isEmpty())
    {
        return Health.down().build();
    }
    return Health.up().build();
}
```



```
{
  "status": "DOWN"
}
```