Práctica 3

@Richard Rivero Cornejo

Para realizar la práctica a continuación se utilizó:

- docker y docker-compose
  - Postgres
  - Kong y Konga
  - Ngingx
- Springboot con maven y Java 1.8
- Angular 15
- postman
- Bash

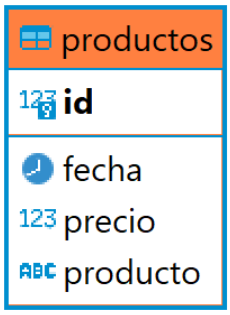
▼ Entregables

- Jar del servicio
- Compilado de angular
- Definición de contenedores
- Este documento

! Para ejecutar los entregables es probable que se necesite ajustar las IPs de algunos docker-compose.yaml

---

▼ Capa de acceso a Datos



Se eligió utilizar solamente una tabla que contenga toda la información de productos y sus precios. A nivel de programación los tipos de datos son:

id → Long

fecha → LocalDate

precio → Double

producto → String

▼ Servicios

Se han implementado tres endpoints en la api productos, uno para listar todos, otro para filtrar por nombre de producto y fecha, y uno para crear los registros.

▼ Get All

Path: /v2/productos/

Método: GET

Descripción: Retorna todos los registros en la persistencia

Ejemplo:

```
curl --location 'localhost:8081/v2/productos/'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2708    0  2708    0     0  31220      0 --:--:-- --:--:-- --:--:-- 31488
[
  {
    "id": 1,
    "fecha": "2012-01-09",
    "producto": "Coca Cola 2l",
    "precio": 5
  },
  {
    "id": 2,
    "fecha": "2013-01-09",
    "producto": "Coca Cola 2l",
    "precio": 5
  },
  {
    "id": 3,
    "fecha": "2014-01-09",
    "producto": "Coca Cola 2l",
```

```
    "precio": 6
  }
]
```

▼ Get by Producto y Fecha

Path: /v2/productos/{producto}/{fecha}

Método: GET

Descripción: Retorna el registro que coincida con fecha y producto

Ejemplo:

```
  curl --location 'localhost:8081/v2/productos/Coca%20Cola%203l/2017-01-09'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    70    0    70    0     0    119      0 --:--:-- --:--:-- --:--:--   119
{
  "id": 29,
  "fecha": "2017-01-09",
  "producto": "Coca Cola 3l",
  "precio": 13
}
```

▼ Create

Path: /v2/productos/

Método: POST

Descripción: Guarda un registro

Ejemplo:

```
curl --location 'localhost:8081/v2/productos' \
--header 'Content-Type: application/json' \
--data '
{
    "fecha": "2010-01-10",
    "producto": "Coca Cola 2l",
    "precio": 4.00
}'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   151    0    69  100    82    335    398 --:--:-- --:--:-- --:--:--   736
{
  "id": 39,
```

```
  "fecha": "2010-01-10",
  "producto": "Coca Cola 2l",
  "precio": 4
}
```

▼ Servicios en Kong

Para exponer el servicio en Kong, se puede tener el servicio corriendo localmente o en docker. El kong está en contenedores. A continuación la lista de puertos usados:

servicio-standalone → 8081

servicio-standalone → 8081, 8082, 8083

kong-configuración → 8001

kong-listener → 8000

konga → 1337

Si bien se puede configurar en konga, mostraré como hacerlo con peticiones al 8001

StandAlone

Creamos el servicio que apunta a la api, dockerizada o localmente

```
curl -i -X POST --url http://localhost:8001/services/ \
--data 'name=standalone' \
--data 'url=http://ip_servicio:8081/v2/productos'
```

A este servicio le creamos una ruta para que se mapee las peticiones al servicio

```
curl -i -X POST --url http://localhost:8001/services/standalone/routes \
--data 'name=standalone_route' \
--data 'paths[]=/standalone/productos'
```

Ahora podemos probar llamando a cualquiera de los servicios en la ruta http://ip_kong:8000/standalone/productos

```
curl --location 'localhost:8000/standalone/productos/Coca%20Cola%203l/2017-01-09'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    70    0    70    0     0    311      0 --:--:-- --:--:-- --:--:--   311
{
  "id": 29,
  "fecha": "2017-01-09",
  "producto": "Coca Cola 3l",
  "precio": 13
}
```

Balance de carga

Si es que tenemos múltiples instancias del servicio y queremos que kong distribuya la carga podemos hacer lo siguiente:

Creamos el upstream para productos

```
curl -i -X POST --url http://localhost:8001/upstreams \
--data 'name=productos_upstream'
```

Añadimos los targets al upstream, estos son las direcciones IPs y puertos de los nodos con el servicio

```
curl -i -X POST --url http://localhost:8001/upstreams/productos_upstream/targets \
 --data target='192.168.100.133:8081' # nodo 1
curl -i -X POST --url http://localhost:8001/upstreams/productos_upstream/targets \
 --data target='192.168.100.133:8082' # nodo 2
curl -i -X POST --url http://localhost:8001/upstreams/productos_upstream/targets \
 --data target='192.168.100.133:8083' # nodo 3
```

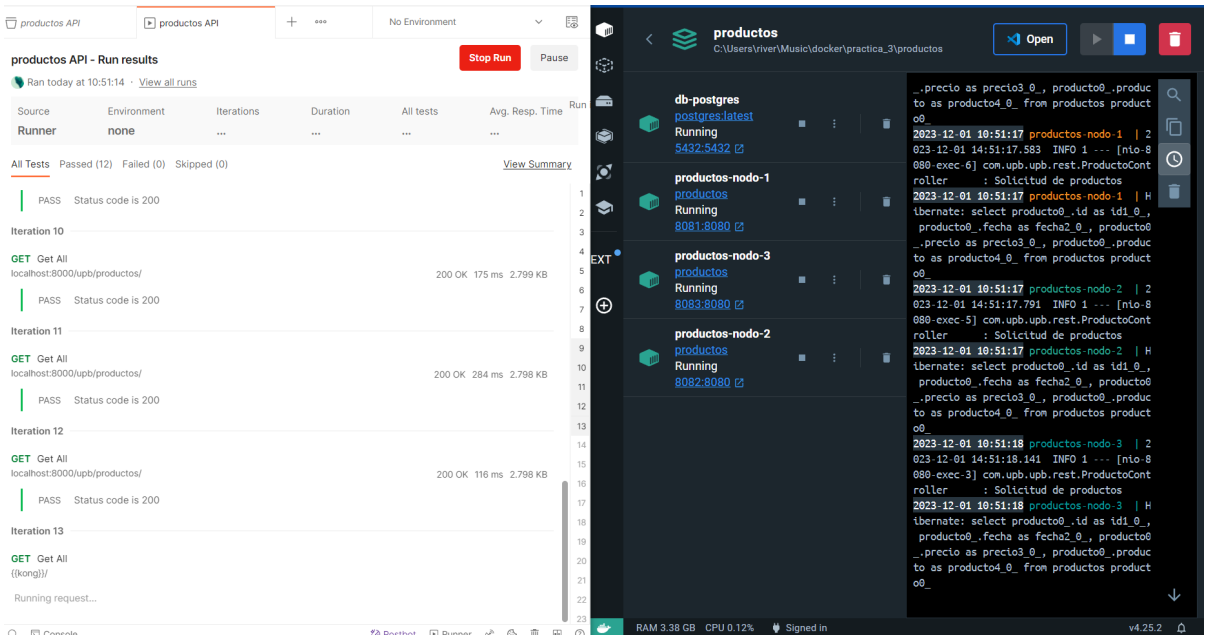
Ahora hacemos lo mismo que en el standalone, pero en vez de la ip del servicio ponemos el nombre del upstream

```
# creación de servicio
curl -i -X POST --url http://localhost:8001/services/ \
--data 'name=cluster' \
--data 'url=http://productos_upstream/v2/productos'
# nota que el host es el nombre del upstream

# creación de la ruta
```

```
curl -i -X POST --url http://localhost:8001/services/cluster/routes \
--data 'name=cluster_route' \
--data 'paths[]=/upb/productos'
```

Para probar que se está accediendo a los tres nodos, podemos usar postman y hacer 100 solicitudes (Click the collection > run collection > select a service > 100 iterations > run). Y utilizar un encabezado que nos diga de donde viene la respuesta o ver los logs de docker (clik en el grupo productos).



De esta manera podemos ver que usa los tres nodos para responder las peticiones

▼ Cliente

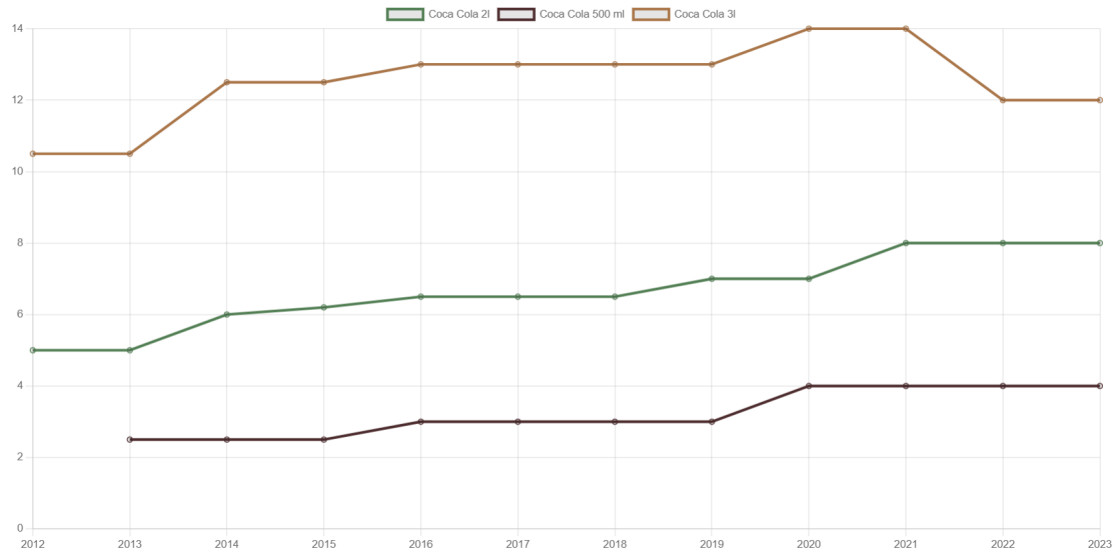
Se hizo un cliente an angular consumiendo el servicio con HttpClient y haciendo gráficos con la librería Chart.js

```
import { Injectable } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Enviroment } from '../enviroments/enviroment';
import { Observable } from 'rxjs';


@Injectable({
  providedIn: 'root'
})
export class ProductosService {
```

```
  private api_url: string = '';
  constructor(
    private http:HttpClient
  ) {
    this.api_url = Enviroment.api_host+Enviroment.api_path;
  }

  findAll(): Observable<any> {
    return this.http.get(this.api_url);
  }
}
```



```
                                                                        app.component.ts:24
▼ (35) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
  {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}] i
  ▶ 0: {id: 1, fecha: '2012-01-09', producto: 'Coca Cola 2l', precio: 5}
  ▶ 1: {id: 2, fecha: '2013-01-09', producto: 'Coca Cola 2l', precio: 5}
  ▶ 2: {id: 3, fecha: '2014-01-09', producto: 'Coca Cola 2l', precio: 6}
  ▶ 3: {id: 4, fecha: '2015-01-09', producto: 'Coca Cola 2l', precio: 6.2}
  ▶ 4: {id: 5, fecha: '2016-01-09', producto: 'Coca Cola 2l', precio: 6.5}
  ▶ 5: {id: 6, fecha: '2017-01-09', producto: 'Coca Cola 2l', precio: 6.5}
  ▶ 6: {id: 7, fecha: '2018-01-09', producto: 'Coca Cola 2l', precio: 6.5}
  ▶ 7: {id: 8, fecha: '2019-01-09', producto: 'Coca Cola 2l', precio: 7}
  ▶ 8: {id: 9, fecha: '2020-01-09', producto: 'Coca Cola 2l', precio: 7}
  ▶ 9: {id: 10, fecha: '2021-01-09', producto: 'Coca Cola 2l', precio: 8}
  ▶ 10: {id: 11, fecha: '2022-01-09', producto: 'Coca Cola 2l', precio: 8}
  ▶ 11: {id: 12, fecha: '2023-01-09', producto: 'Coca Cola 2l', precio: 8}
  ▶ 12: {id: 13, fecha: '2013-01-09', producto: 'Coca Cola 500 ml', precio: 2.5}
  ▶ 13: {id: 14, fecha: '2014-01-09', producto: 'Coca Cola 500 ml', precio: 2.5}
  ▶ 14: {id: 15, fecha: '2015-01-09', producto: 'Coca Cola 500 ml', precio: 2.5}
  ▶ 15: {id: 16, fecha: '2016-01-09', producto: 'Coca Cola 500 ml', precio: 3}
  ▶ 16: {id: 17, fecha: '2017-01-09', producto: 'Coca Cola 500 ml', precio: 3}
  ▶ 17: {id: 18, fecha: '2018-01-09', producto: 'Coca Cola 500 ml', precio: 3}
  ▶ 18: {id: 19, fecha: '2019-01-09', producto: 'Coca Cola 500 ml', precio: 3}
  ▶ 19: {id: 20, fecha: '2020-01-09', producto: 'Coca Cola 500 ml', precio: 4}
  ▶ 20: {id: 21, fecha: '2021-01-09', producto: 'Coca Cola 500 ml', precio: 4}
  ▶ 21: {id: 22, fecha: '2022-01-09', producto: 'Coca Cola 500 ml', precio: 4}
  ▶ 22: {id: 23, fecha: '2023-01-09', producto: 'Coca Cola 500 ml', precio: 4}
  ▶ 23: {id: 24, fecha: '2012-01-09', producto: 'Coca Cola 3l', precio: 10.5}
  ▶ 24: {id: 25, fecha: '2013-01-09', producto: 'Coca Cola 3l', precio: 10.5}
  ▶ 25: {id: 26, fecha: '2014-01-09', producto: 'Coca Cola 3l', precio: 12.5}
  ▶ 26: {id: 27, fecha: '2015-01-09', producto: 'Coca Cola 3l', precio: 12.5}
  ▶ 27: {id: 28, fecha: '2016-01-09', producto: 'Coca Cola 3l', precio: 13}
  ▶ 28: {id: 29, fecha: '2017-01-09', producto: 'Coca Cola 3l', precio: 13}
  ▶ 29: {id: 30, fecha: '2018-01-09', producto: 'Coca Cola 3l', precio: 13}
  ▶ 30: {id: 31, fecha: '2019-01-09', producto: 'Coca Cola 3l', precio: 13}
  ▶ 31: {id: 32, fecha: '2020-01-09', producto: 'Coca Cola 3l', precio: 14}
  ▶ 32: {id: 33, fecha: '2021-01-09', producto: 'Coca Cola 3l', precio: 14}
  ▶ 33: {id: 34, fecha: '2022-01-09', producto: 'Coca Cola 3l', precio: 12}
  ▶ 34: {id: 35, fecha: '2023-01-09', producto: 'Coca Cola 3l', precio: 12}
    length: 35
  ▶ [[Prototype]]: Array(0)
```