Girón Guzmán Agustín
IBM Academia

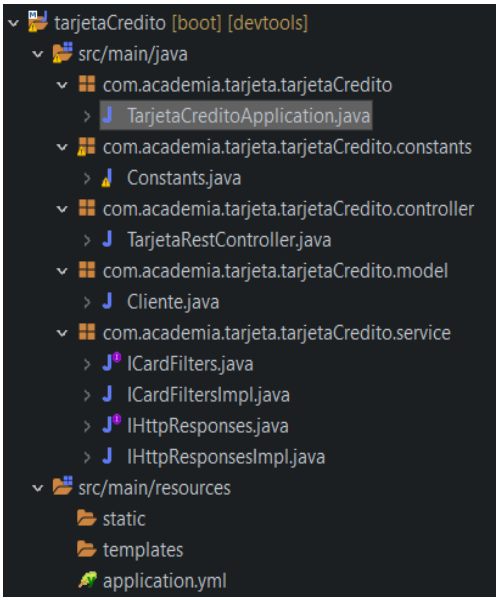
Link de Github:

https://github.com/tolon-tiolino/IBMAcademiaEj.git

tarjetaCredito

Microservicio que funge como servidor y que, a través de la recepción de un método POST proveniente del cliente o de manera directa a su endpoint, procesa la pasión (passion), el salario mensual (monthSalary) y la edad (age) del cliente, en un objeto, para posteriormente analizar sus atributos y regresar como respuesta el tipo de tarjetas que se le pudieran ofrecer al cliente o inclusive, si este no entrara en ningún apartado, regresa un mensaje de que no hay opción apta (no suitable option).

A grandes rasgos, la aplicación cuenta con los siguientes paquetes y clases en los cuales se divide toda la lógica de la aplicación:



Constans.java -> contiene constantes utilizadas a lo largo del programa.

TarjetaRestController -> cuenta con el endpoint que responde a la petición POST.

Cliente -> define el modelo del objeto que ayuda en el procesamiento de los datos recibidos.

ICardFilters -> define una interfaz que marca los métodos que aquellas implementaciones de esta deben de contemplar para cubrir la lógica del negocio.

ICardFiltersImpl -> define la implementación de la interfaz “ICardFilters” en la que se encuentran aquellos métodos que regresan el tipo de tarjeta que se le recomienda al cliente.

IHttpResponses -> define una interfaz que marca los métodos que aquellas implementaciones de esta deben de contemplar para cubrir la respuesta al método POST.

IHttpResponsesImpl -> define la implementación de la interfaz “IHttpResponses” en la que se encuentra aquel método que define la respuesta al método POST.

También se cuenta con el archivo “application.yml” que define las propiedades del microservicio.

Girón Guzmán Agustín
IBM Academia

Endpoints

POST: http://localhost:8860/api/cardType/get-cardtype

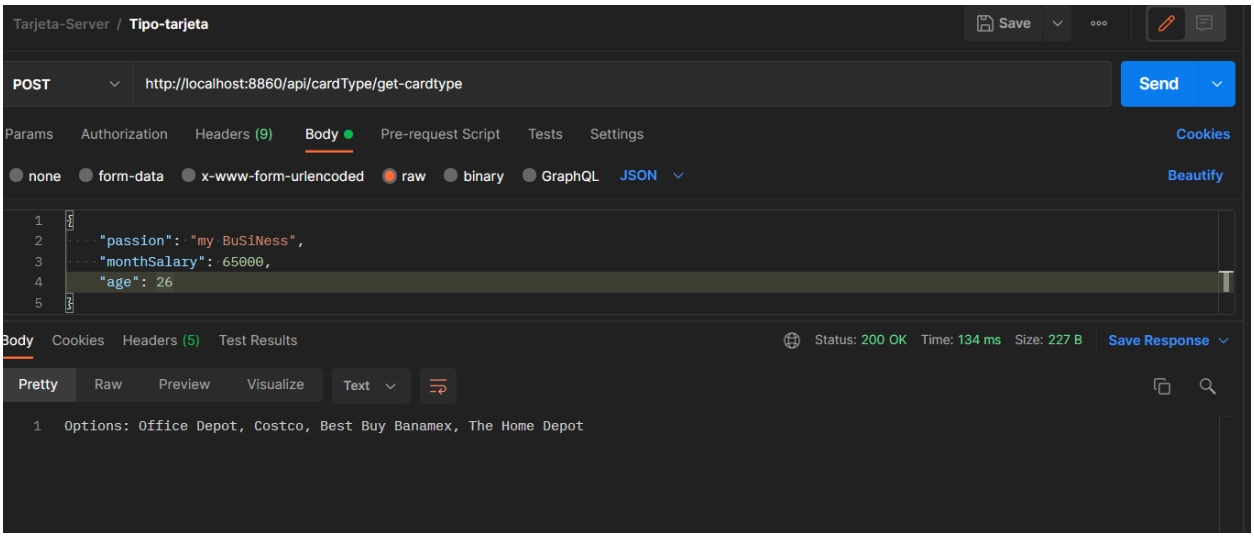
Recibe y procesa con un POST, el siguiente json:

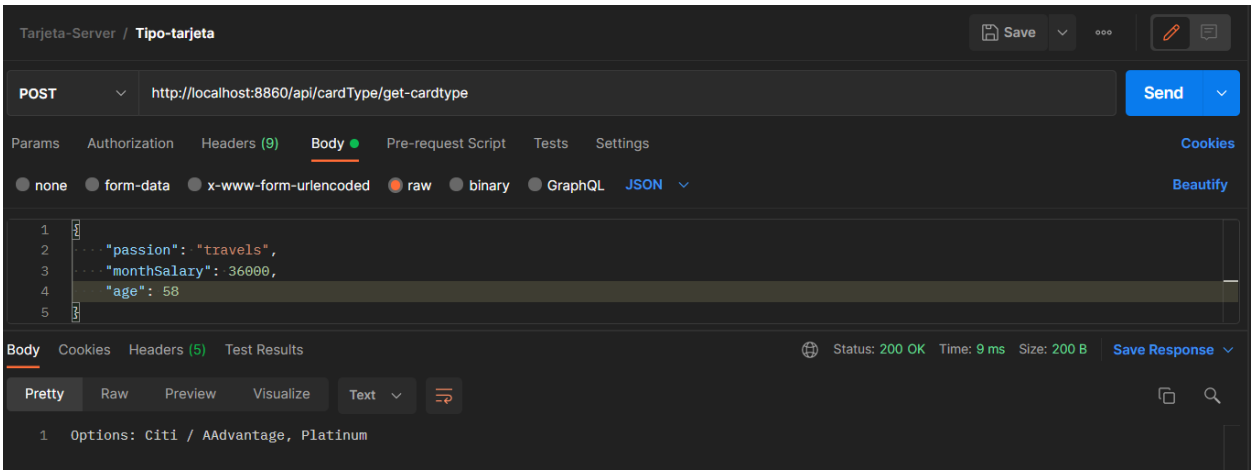
```
{
  "passion": "shopping",
  "monthSalary": 12000,
  "age": 26
}
```

El cual permite definir aquellas características del cliente para que, acorde a la lógica del negocio, regrese como respuesta aquellas tarjetas que se pudieran ofrecer al cliente.

Consumir la API

Postman

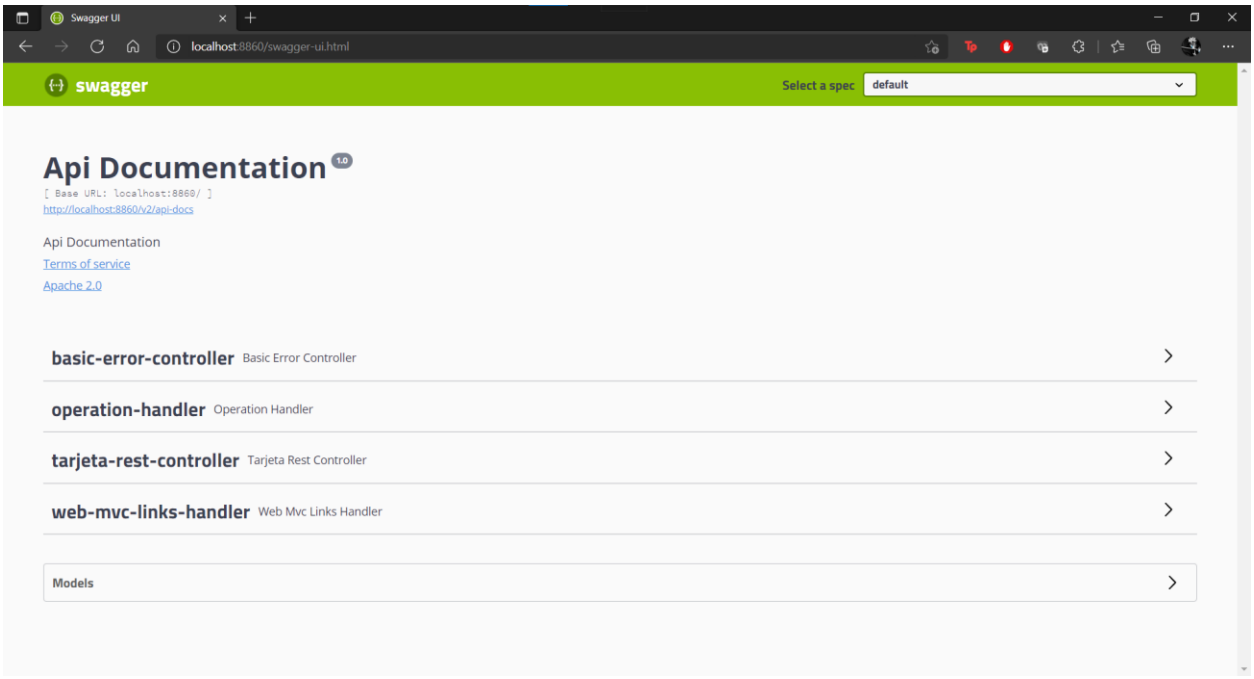


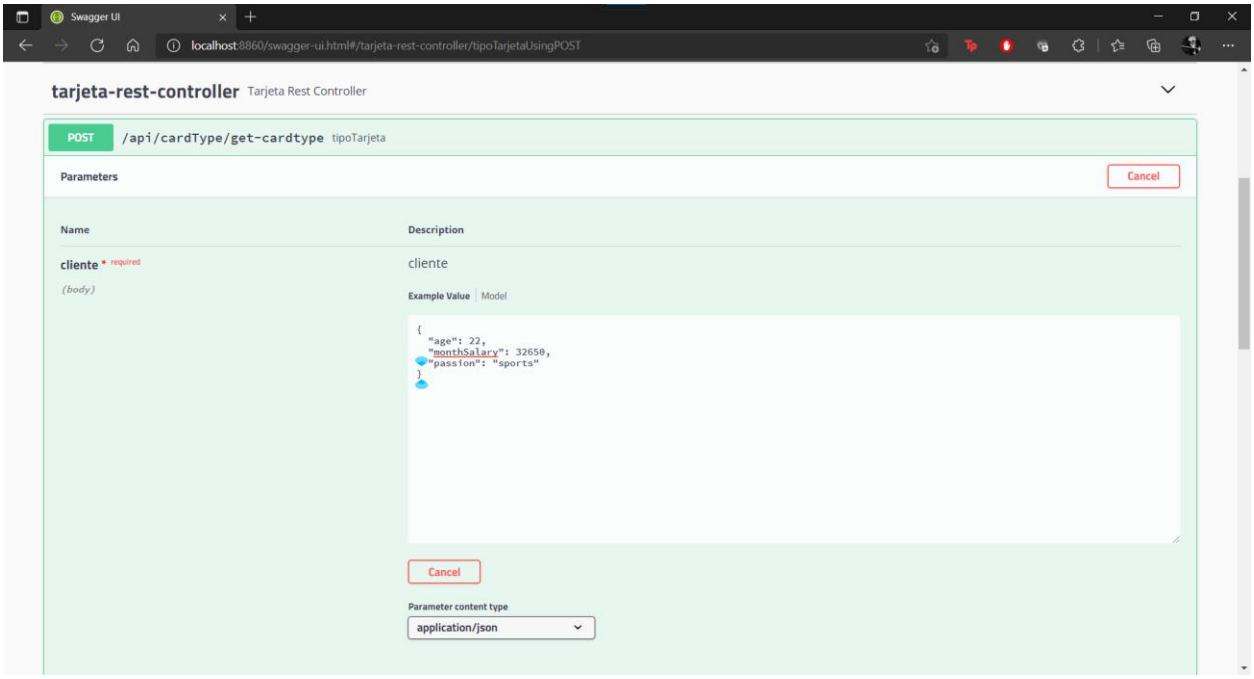


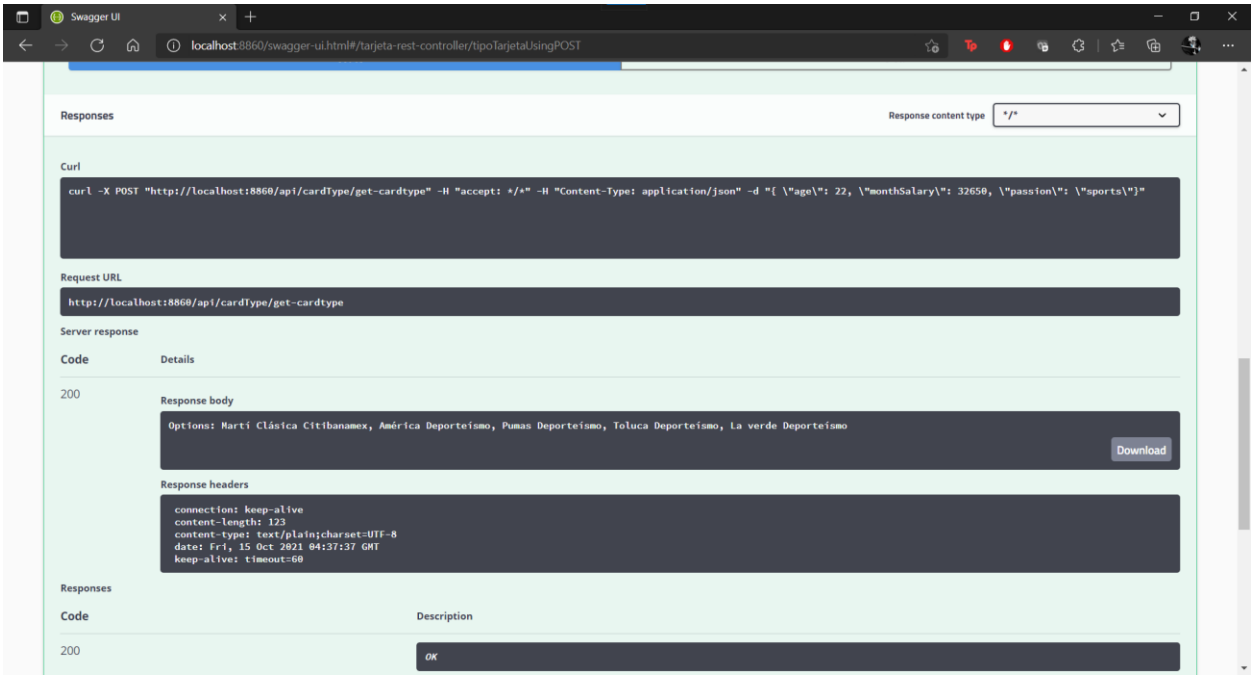
Si no se cuenta con Postman, se implementan unas dependencias de Swagger para que, a través de la siguiente URL:

http://localhost:8860/swagger-ui.html

Se pueda probar el método POST del microservicio en el navegador:





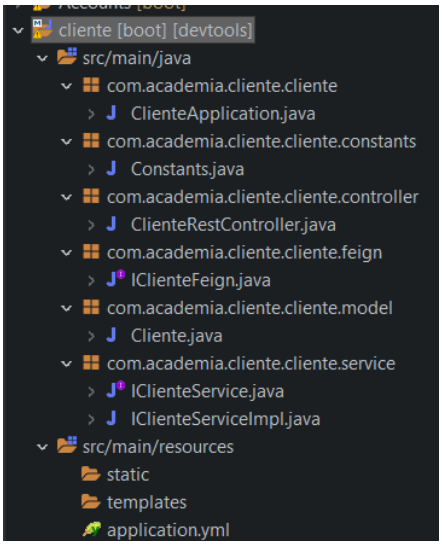

Swagger UI
localhost:8860/swagger-ui.html#/tarjeta-rest-controller/tipoTarjetaUsingPOST
Responses
Response content type
/
Curl
curl -X POST "http://localhost:8860/api/cardType/get-cardtype" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"age\": 22, \"monthSalary\": 32650, \"passion\": \"sports\"}"
Request URL
http://localhost:8860/api/cardType/get-cardtype
Server response
Code
Details
200
Response body
Options: Martí Clásica Citibanamex, América Deporteísmo, Pumas Deporteísmo, Toluca Deporteísmo, La verde Deporteísmo
Download
Response headers
connection: keep-alive
content-length: 123
content-type: text/plain;charset=UTF-8
date: Fri, 15 Oct 2021 04:37:37 GMT
keep-alive: timeout=60
Responses
Code
Description
200
OK

Girón Guzmán Agustín
IBM Academia

cliente

Microservicio que funge como cliente y que, a través de la recepción de un método POST, el procesamiento de la pasión (passion), el salario mensual (monthSalary) y la edad (age) del cliente, en un objeto y posteriormente de su envío al servidor gracias a Feign y a EurekaServer, recibe la respuesta del servidor indicando el tipo de tarjetas que se le pudieran ofrecer al cliente o inclusive, si no entrara en ningún apartado, regresa un mensaje de que no hay opción apta (no suitable option).

A grandes rasgos, la aplicación cuenta con los siguientes paquetes y clases en los cuales se divide toda la lógica de la aplicación:



Constans.java -> contiene constantes utilizadas a lo largo del programa.

ClienteRestController -> cuenta con el endpoint que responde a la petición POST.

IClienteFeign -> define una interfaz que marca los métodos que Feign va a referenciar cuando se mande la petición POST de allí a Eureka Server y posteriormente al Servidor.

Cliente -> define el modelo del objeto que ayuda en el procesamiento de los datos recibidos.

IClienteService -> define una interfaz que marca el método que aquellas implementaciones de esta deben de contemplar para la llamada a Feign.

IClienteServiceImpl -> define la implementación de la interfaz “IClienteService” en la que se encuentra el método que llama a Feign para enviar la petición a Eureka y con esto, al servidor.

También se cuenta con el archivo “application.yml” que define las propiedades del microservicio.

Girón Guzmán Agustín
IBM Academia

Endpoints

POST: http://localhost:8040/tipo-tarjeta

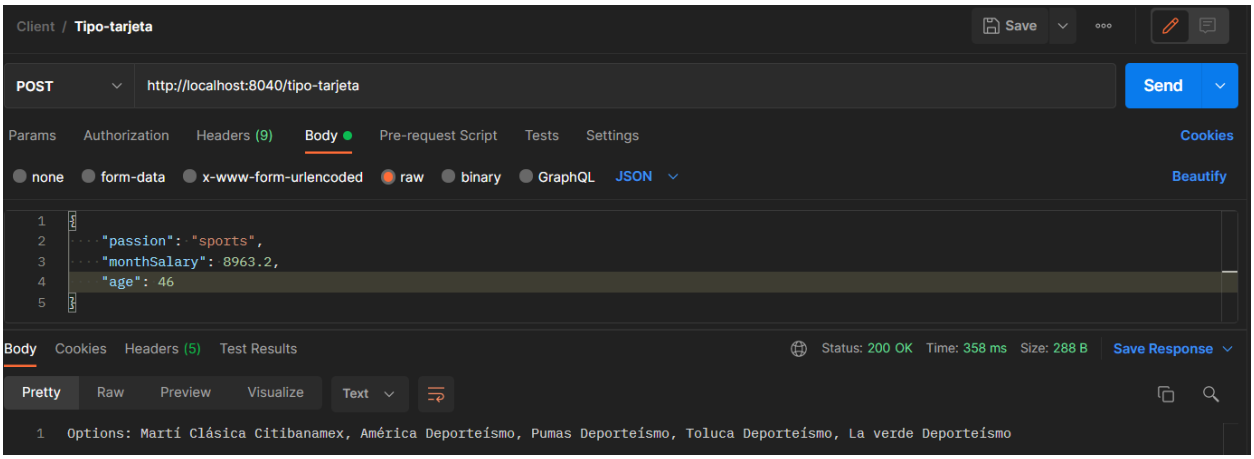
Recibe y procesa con un POST, el siguiente json:

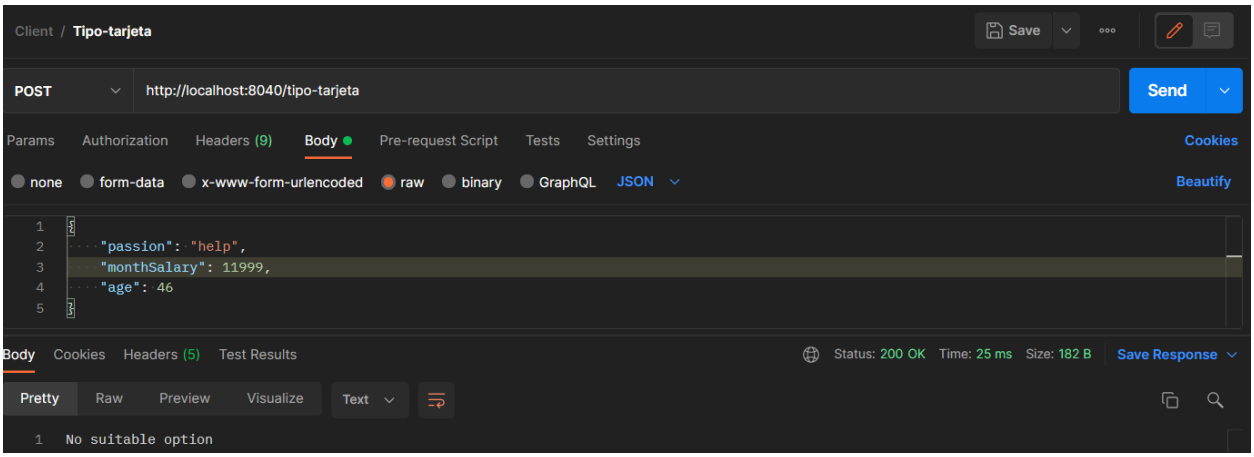
```
{
  "passion": "shopping",
  "monthSalary": 12000,
  "age": 26
}
```

El cual permite definir aquellas características del cliente para que, acorde a la lógica del negocio que proporciona la respuesta del servidor, muestra aquellas tarjetas que se pudieran ofrecer al cliente.

Consumir la API

Postman

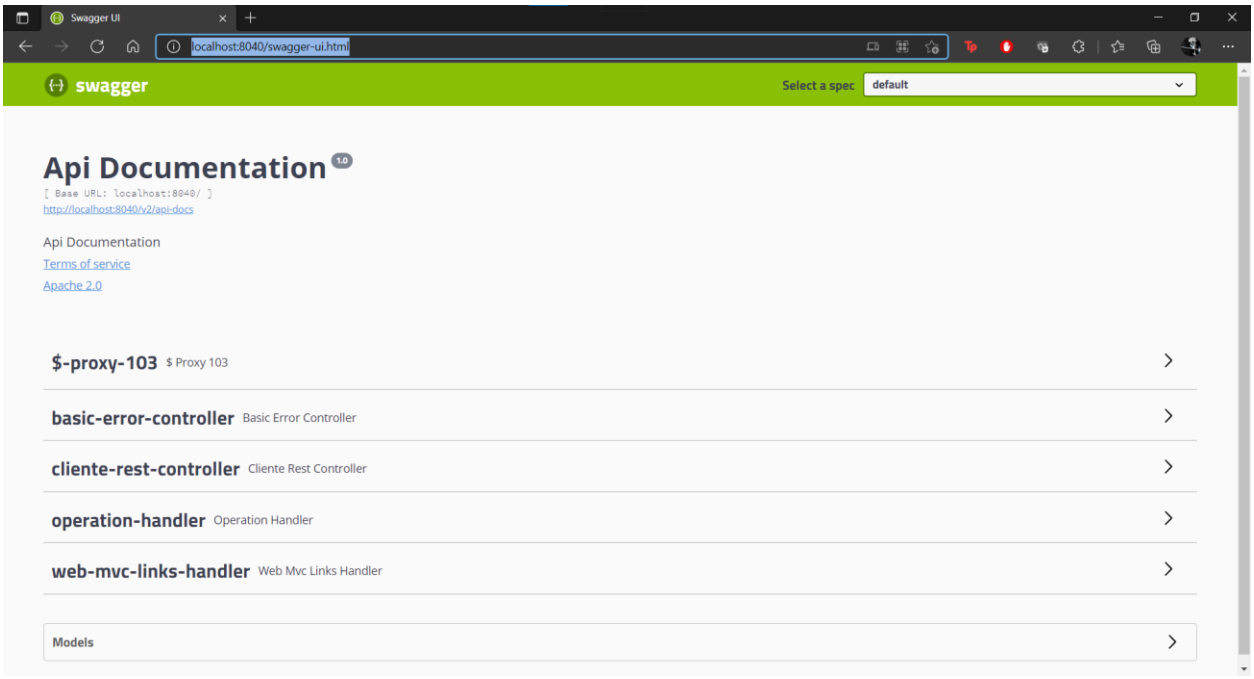


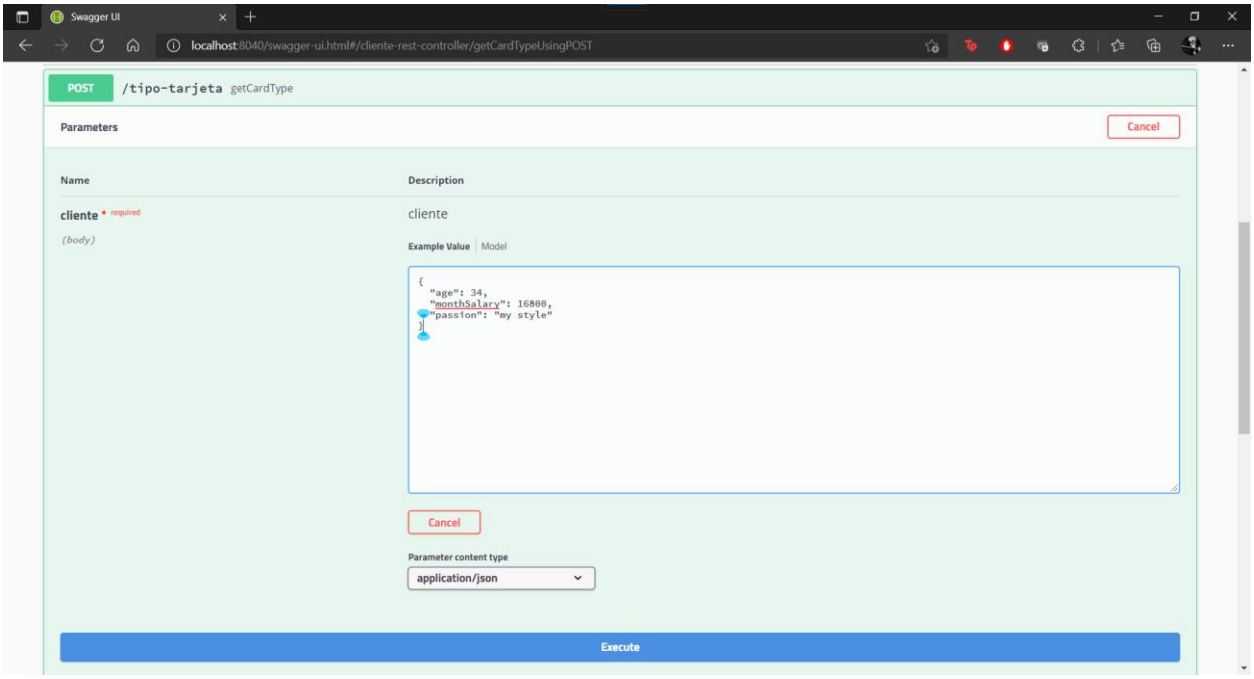


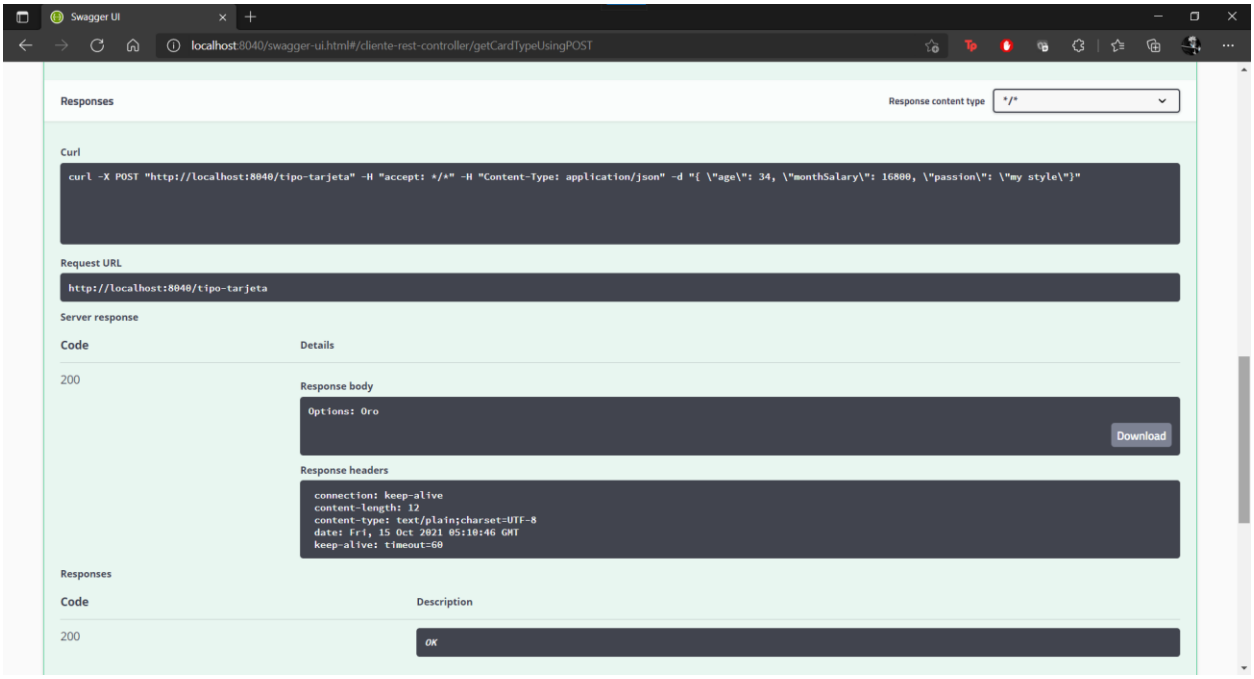
Si no se cuenta con Postman, se implementan unas dependencias de Swagger para que, a través de la siguiente URL:

http://localhost:8040/swagger-ui.html

Se pueda probar el método POST del microservicio en el navegador:





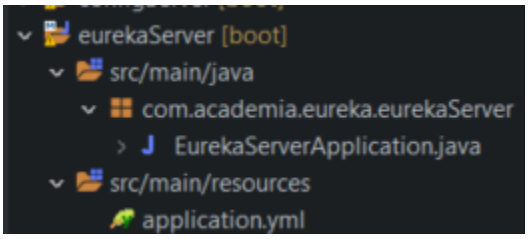

Swagger UI
localhost:8040/swagger-ui.html#/cliente-rest-controller/getCardTypeUsingPOST
Responses
Response content type
/
Curl
curl -X POST "http://localhost:8040/tipo-tarjeta" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"age\": 34, \"monthSalary\": 16800, \"passion\": \"my style\"}"
Request URL
http://localhost:8040/tipo-tarjeta
Server response
Code
Details
200
Response body
Options: Oro
Download
Response headers
connection: keep-alive
content-length: 12
content-type: text/plain;charset=UTF-8
date: Fri, 15 Oct 2021 05:10:46 GMT
keep-alive: timeout=60
Responses
Code
Description
200
OK

Girón Guzmán Agustín
IBM Academia

eurekaServer

Microservicio que funge como un servidor DNS en el cual se pueden registrar a los clientes y que permite que estos puedan tener comunicación sin necesidad de saber la URL y puerto exactos de cada uno.

A grandes rasgos, la aplicación cuenta con los siguientes paquetes y clases en los cuales se divide toda la lógica de la aplicación:

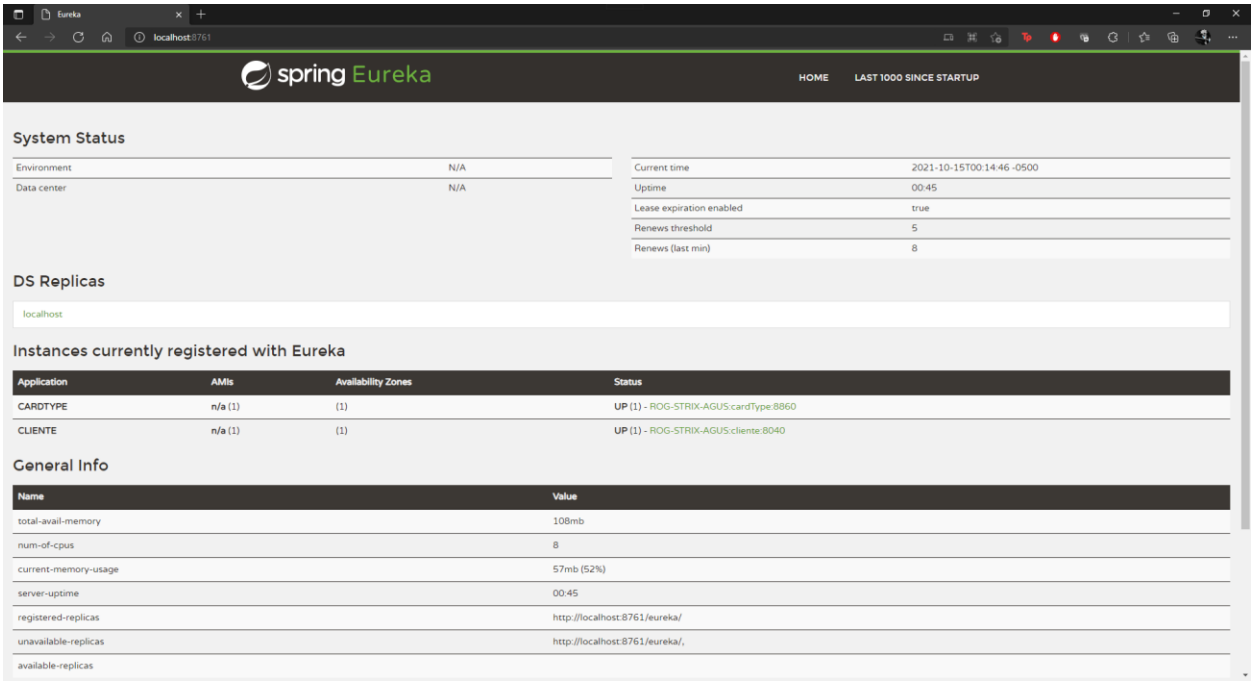


También se cuenta con el archivo “application.yml” que define las propiedades del microservicio.

Consumir la API

Conociendo la URL y puerto de este, se puede acceder a su dashboard que permite visualizar aquellos microservicios registrados y su estatus.

En este caso: http://localhost:8761/



Dicho servidor trabaja de la mano con Feign en el cliente para poder dirigir las peticiones y respuestas entre los primeros dos microservicios sin que estos conozcan directamente la ubicación exacta del otro.