gateway2

No pude llamarlo gateway porque antes cree un proyecto con ese nombre y tuve que borrarlo y por eso no me dejaba poner ese nombre porque me daba errores, por eso lo llamé gateway2. He creado dos resources, uno para los socios y otro para los productos. Cada uno de estos resources se conecta al web service correspondiente.

SociosResources

Los métodos de esta clase producen un MediaType.APPLICATION_JSON, por lo que los responses tendrán el correspondiente header application/json, su url es http://localhost:8888/socios

GET requestSocios()

Este método no consume nada y devuelve una lista en Json de todos los socios que estén guardados en la base de datos. Su url es http://localhost:8888/socios. Devuelve el código 200 OK.

Ejemplo:[{


```
    "codigo": "XB",
    "email": "ana@asdf-com",
    "nombre": "ANA",
    "saldo": 23.0
  },
  {
    "codigo": "XC",
    "email": "pepe@asdf-com",
    "nombre": "PEPE",
    "saldo": 8.0
  },
  {
    "codigo": "1",
    "email": "miguel@upsa.es",
    "nombre": "Miguel",
    "saldo": 32.0
  },
  {
    "codigo": "XA",
    "email": "enrqiue@asdf-com",
    "nombre": "ENRIQUE",
    "saldo": 320.0
  }
]
```

GET requestSocioByCodigo(PathParam(“codigo“))

Su url es http://localhost:8888/socios/{codigo}.Este método devuelve el socio que se ha introducido como PathParam y devuelve la información del socio con codigo 200 OK. Por ejemplo si introducimos http://localhost:8888/socios/XA saldrá el siguiente JSON:


```
{
  "codigo": "XA",
  "email": "enrqiue@asdf-com",
  "nombre": "ENRIQUE",
  "saldo": 320.0
}
```


POST addSocio(UnidentifiedSocio)

Este método consume un MediaType.APPLICATION_JSON con un UnindentifiedSocio, por ejemplo: {


```
  "nombre":"Roberto",
  "email":"roberto@upsa.es"
}
```


La url es http://localhost:8888/socios y devuelve un Response con código 201 Created y un entity con un JSON con la información del socio creado, con el código que se le asigne y el saldo por defecto es 0 al crear un nuevo socio, un ejemplo es: {


```
  "codigo": "26",
  "email": "roberto@upsa.es",
  "nombre": "Roberto",
  "saldo": 0.0
}
```


PUT putSocio(Socio)

Este método consume un MediaType.APPLICATION_JSON con Socio, por ejemplo:


```
{
  "codigo": "26",
  "email": "roberto@upsa.es",
  "nombre": "Roberto",
  "saldo": 100.0
}
```

La url es http://localhost:8888/socios y devuelve un Response con código 201 Created y un entity con un JSON con la información del socio actualizado(si no hay un socio con ese código se lanza el código 404 Not found), un ejemplo de respuesta es:


```
{
  "código": "26",
  "email": "roberto@upsa.es",
  "nombre": "Roberto",
  "saldo": 100.0
}
```


DELETE deleteSocio(PathParam(“codigo”))

La url es http://localhost:8888/socios/{codigo}, este método recibe el path param código para saber que socio tiene que ser borrado ,devuelva un Response con el código 200 OK si existe el socio y lo ha borrado, si no devuelve el código 404 Not Found.

ProductosResource

Este resource tiene de url http://localhost:8888/productos, todos sus métodos devuelven un Response que produce un MediaType.APPLICATION_JSON y su header application/json

GET requestProductos()

La url de este método es http://localhost:8888/productos y no consume nada, lo que hace es devolver una lista con todos los productos existentes, el código del response que produce es 200 OK y un ejemplo del Json Array que devuelve es:


```
[
  {
    "codigo": "YA",
    "descripcion": "esto es comida",
    "nombre": "Comida",
    "precio": 43.3,
    "stock": 455
  },
  {
    "codigo": "YB",
    "descripcion": "se trata de bebida",
    "nombre": "Bebida",
    "precio": 54.0,
    "stock": 234
  },
  {
```

```
    "codigo": "YC",
    "descripcion": "no se me ocurre",
    "nombre": "Otros",
    "precio": 21.0,
    "stock": 543
  }
]
```


GET getProductoByCodigo(PathParam(“codigo”))

El path de este método es localhost:8888/productos/{codigo}, lo que hace es buscar el producto con el código introducido como PathParam. Devuelve un response con el código 200 OK si existe el alumno y si no existe un 404 Not Found. Si existe además devolverá un entity con el producto en JSON, por ejemplo:


```
{
    "codigo": "YA",
    "descripcion": "esto es comida",
    "nombre": "Comida",
    "precio": 43.3,
    "stock": 455
  }
```


POST addProducto(UnidentifiedProducto)

Este método Consume un MediaType.APPLICATION_JSON, por ejemplo:


```
{
  "nombre":"ordenador",
  "descripcion":"ordenador portatil",
  "stock":50,
  "precio":500.67
}
```


El path de este método es localhost:8888/productos y si ha sido creado devuelve un response con el código 201 Created y la uri correspondiente al producto para ver su información. Un ejemplo de respuesta es:


```
{
  "codigo": "19",
  "descripcion": "ordenador portatil",
  "nombre": "ordenador",
  "precio": 500.67,
  "stock": 50
}
```

PUT replaceProducto(PathParam(“codigo”), UnindentifiedProducto)

Este método necesita consumir un MediaType.APPLICATION_JSON con un UnidentifiedProducto por ejemplo:


```
{
  "descripcion": "ordenador portatil",
  "nombre": "ordenador",
  "precio": 500.67,
  "stock": 30
}
```


Además recibe un PathParam con el código del producto que se quiere modificar, su url es localhost:8888/productos/{codigo}. El response devuelve el codigo 200 OK si existe el producto con ese código y si no 404 NotFound. Un ejemplo de Json que devuelve es:


```
{
  "codigo": "19",
  "descripcion": "ordenador portatil",
  "nombre": "ordenador",
  "precio": 500.67,
  "stock": 50
}
```


DELETE deleteProductos(PathParam(“codigo))

Su path es localhost:8888/productos/{codigo} y borra el producto que tenga el código introducido por el Path param. Devuelve un Response con el codigo 200 OK si todo ha salido bien, si no existe devuelve 404 NotFound.