

MEMORIA P1 RESTFUL

*SISTEMAS ORIENTADOS A SERVICIOS
CURSO 2019-2020*

DEL RINCÓN GARCÍA, MARIO Z170031

DE LA TORRE JUÁREZ, ALBERTO Z170107

Contenido

1.	RESUMEN DEL DISEÑO DEL SERVICIO	3
1.1.	TABLAS POR CADA RECURSO DEFINIDO Y MÉTODO SOPORTADO PARA DICHO RECURSO	3
1.2.	DISEÑO DE LA PERSISTENCIA DEL SERVICIO.....	9
2.	CAPTURAS DE LA EJECUCIÓN DE OPERACIONES DESDE UN CLIENTE REST.....	10
3.	CAPTURAS DE LA EJECUCIÓN DEL CLIENTE DE PRUEBA.....	29

1. RESUMEN DEL DISEÑO DEL SERVICIO

1.1. TABLAS POR CADA RECURSO DEFINIDO Y MÉTODO SOPORTADO PARA DICHO RECURSO

PARA CLIENTES:

Listas de clientes JSON/XML generada con listas en JAXB

URI	http://localhost:8080/miBancoUPM/api /clientes
MÉTODO	GET
CADENA DE CONSULTA	Para obtener listados como el de todas las transferencias emitidas de una cuenta personal o el listado con todos los usuarios y su saldo actual. QUERYPARAMS: Param1 = desde -> para filtrar por fecha Param2 = hasta -> para filtrar por fecha Param3 = limite1 -> limita cantidad de info. obtenida Param4 = limite2 -> limita cantidad de info. obtenida
DEVUELVE	200 OK y POX (miBancoUPM/clientes + xml/json) 400 BAD REQUEST 500 INTERNAL SERVER ERROR

Obtener datos de un cliente

URI	http://localhost:8080/miBancoUPM/api/clientes/{cliente_id}
MÉTODO	GET
CADENA DE CONSULTA	Para obtener los datos de un cliente en concreto. PATHPARAMS: Param1 = cliente_id
DEVUELVE	200 OK y POX(clientes/cliente_id + xml/json) 400 BAD REQUEST 404 NOT FOUND 500 INTERNAL SERVER ERROR

Añadir un cliente al banco o crear perfil de usuario

URI	http://localhost:8080/miBancoUPM/api/clientes
MÉTODO	POST
CADENA DE LA PETICIÓN	POX (miBancoUPM/clientes/cliente_id + xml/json)
DEVUELVE	201 CREATED 500 INTERNAL SERVER ERROR

Modificar datos de un cliente

URI	http://localhost:8080/miBancoUPM/api/clientes/{cliente_id}
MÉTODO	PUT
CADENA DE LA PETICIÓN	POX (clientes/cliente_id + json/xml)
DEVUELVE	200 OK y POX (clientes/cliente_id + json/xml) 404 NOT FOUND 500 INTERNAL SERVER ERROR

Eliminar un cliente (precondición: sin cuentas abiertas)

URI	http://localhost:8080/miBancoUPM/api/clientes/{cliente_id}
MÉTODO	DELETE
DEVUELVE	204 NO CONTENT 404 NOT FOUND 412 PRECONDITION FAILED 500 INTERNAL SERVER ERROR

PARA CUENTAS BANCARIAS:

Listas de cuentas de clientes JSON/XML generada con listas en JAXB

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias
MÉTODO	GET
CADENA DE CONSULTA	Para obtener listados de cuentas asociadas a un cliente. PATHPARAMS: Param1 = id_cliente QUERYPARAMS: Param2 = desde -> para filtrar por fecha Param3 = hasta -> para filtrar por fecha Param4 = limite1 -> limita cantidad de info. obtenida Param5 = limite2 -> limita cantidad de info. obtenida
DEVUELVE	200 OK y POX (miBancoUPM/api/clientes/id_cliente/cuentas_bancarias + xml/json) 400 BAD REQUEST 500 INTERNAL SERVER ERROR

Obtener datos de una cuenta bancaria

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}
MÉTODO	GET
CADENA DE CONSULTA	Para obtener los datos de una cuenta bancaria en concreto. PATHPARAMS: Param1 = cuenta_bancaria_id
DEVUELVE	200 OK y POX(clientes/id_cliente/cuentas_bancarias/cuenta_bancaria_id + xml/json) 400 BAD REQUEST 404 NOT FOUND 500 INTERNAL SERVER ERROR

Crear una cuenta bancaria

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias
MÉTODO	POST
CADENA DE LA PETICIÓN	POX (clientes/id_cliente/cuentas_bancarias/cuenta_bancaria_id + json/xml)
DEVUELVE	201 CREATED 500 INTERNAL SERVER ERROR

Eliminar una cuenta bancaria (precondición: saldo = 0)

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}
MÉTODO	DELETE
DEVUELVE	204 NO CONTENT 404 NOT FOUND 412 PRECONDITION FAILED 500 INTERNAL SERVER ERROR

PARA MOVIMIENTOS ENTRE CUENTAS:

Listas de movimientos con o sin filtro JSON/XML generada con listas en JAXB

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/movimientos
MÉTODO	GET
CADENA DE CONSULTA	<p>Para obtener listados de movimientos asociadas a un cliente.</p> <p>PATHPARAMS: Param1 = id_cliente</p> <p>QUERYPARAMS: Param2 = desde -> para filtrar por fecha Param3 = hasta -> para filtrar por fecha Param4 = limite1 -> limita cantidad de info. obtenida Param5 = limite2 -> limita cantidad de info. obtenida</p>
DEVUELVE	<p>200 OK y POX (clientes/id_cliente/cuentas_bancarias/movimientos + json/xml)</p> <p>400 BAD REQUEST</p> <p>500 INTERNAL SERVER ERROR</p>

PARA RETIRADAS DE EFECTIVO:

Listas de retiradas de efectivo con o sin filtro JSON/XML generada con listas en JAXB

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/retiradas_efectivo
MÉTODO	GET
CADENA DE CONSULTA	<p>Para obtener listados de cuentas asociadas a un cliente.</p> <p>PATHPARAMS: Param1 = cuenta_bancaria_id</p> <p>QUERYPARAMS: Param2 = desde -> para filtrar por fecha Param3 = hasta -> para filtrar por fecha Param4 = limite1 -> limita cantidad de info. obtenida Param5 = limite2 -> limita cantidad de info. obtenida</p>
DEVUELVE	<p>200 OK y POX (cuentas_bancarias/cuenta_bancaria_id/movimientos/retiradas_efectivo + json/xml)</p> <p>400 BAD REQUEST</p> <p>500 INTERNAL SERVER ERROR</p>

Obtener datos de una retirada de efectivo

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/retiradas_efectivo/{retirada_efectivo_id}
MÉTODO	GET
CADENA DE CONSULTA	Para obtener los datos de una retirada de efectivo en concreto. PATHPARAMS: Param1 = retirada_efectivo_id
DEVUELVE	200 OK y POX(cuentas_bancarias/cuenta_bancaria_id/movimientos/retiradas_efectivo/retirada_efectivo_id + json/xml) 400 BAD REQUEST 404 NOT FOUND 500 INTERNAL SERVER ERROR

Crear una retirada de efectivo

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/movimientos/{cuenta_bancaria_id}/retiradas_efectivo
MÉTODO	POST
CADENA DE LA PETICIÓN	POX(cuentas_bancarias/cuenta_bancaria_id/movimientos/retiradas_efectivo/retirada_efectivo_id + json/xml)
DEVUELVE	201 CREATED 406 NOT ACCEPTABLE 500 INTERNAL SERVER ERROR

PARA TRANSFERENCIAS:

Listas de transferencias filtrada por fecha y limite JSON/XML generada con listas en JAXB

URI	http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/transferencias
MÉTODO	GET
CADENA DE CONSULTA	Para obtener listados de cuentas asociadas a un cliente. PATHPARAMS: Param1 = cuenta_bancaria_id QUERYPARAMS: Param2 = desde -> para filtrar por fecha Param3 = hasta -> para filtrar por fecha Param4 = limite1 -> limita cantidad de info. obtenida Param5 = limite2 -> limita cantidad de info. obtenida
DEVUELVE	200 OK y POX (cuentas_bancarias/cuenta_bancaria_id/movimientos/transferencias + json/xml) 400 BAD REQUEST 500 INTERNAL SERVER ERROR

Obtener datos de una transferencia

URI	<code>http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/transferencias/{transferencia_id}</code>
MÉTODO	GET
CADENA DE CONSULTA	<p>Para obtener los datos de una transferencia en concreto.</p> <p>PATHPARAMS: Param1 = transferencia_id</p> <p>QUERYPARAMS: Param2 = desde -> para filtrar por fecha Param3 = hasta -> para filtrar por fecha Param4 = limite1 -> limita cantidad de info. obtenida Param5 = limite2 -> limita cantidad de info. obtenida</p>
DEVUELVE	<p>200 OK y POX(cuentas_bancarias/cuenta_bancaria_id/movimientos/transferencias/transferencia_id + json/xml)</p> <p>400 BAD REQUEST</p> <p>404 NOT FOUND</p> <p>500 INTERNAL SERVER ERROR</p>

Crear una transferencia

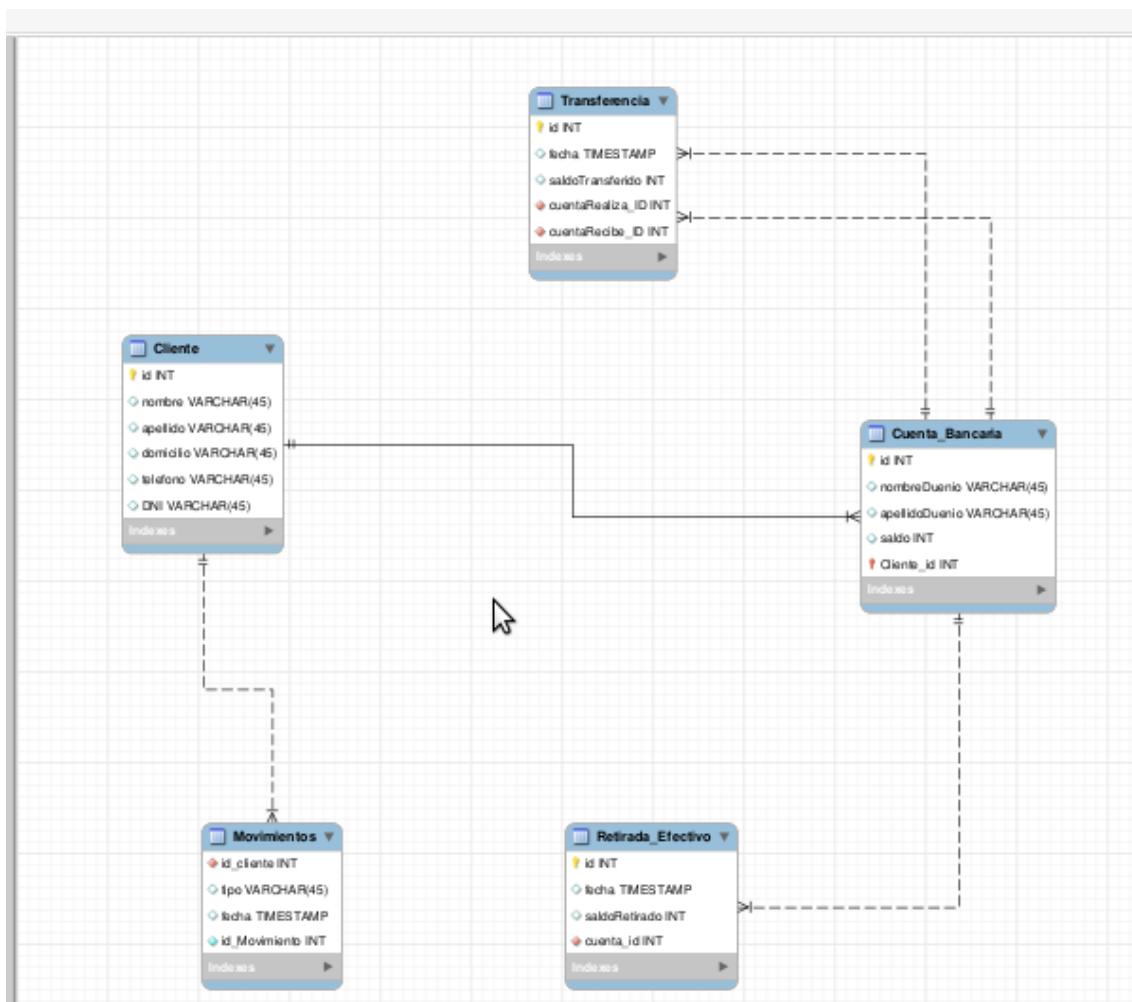
URI	<code>http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/<u>retiradas_efectivo</u></code>
MÉTODO	POST
CADENA DE LA PETICIÓN	POX(cuentas_bancarias/cuenta_bancaria_id/movimientos/transferencias/transferencia_id + json/xml)
DEVUELVE	<p>201 CREATED</p> <p>406 NOT ACCEPTABLE</p> <p>500 INTERNAL SERVER ERROR</p>

Eliminar una transferencia

URI	<code>http://localhost:8080/miBancoUPM/api/clientes/{id_cliente}/cuentas_bancarias/{cuenta_bancaria_id}/movimientos/transferencias/{transferencia_id}</code>
MÉTODO	DELETE
DEVUELVE	<p>204 NO CONTENT</p> <p>404 NOT FOUND</p> <p>500 INTERNAL SERVER ERROR</p>

1.2. DISEÑO DE LA PERSISTENCIA DEL SERVICIO

DIAGRAMA EER DE LA BASE DE DATOS:



2. CAPTURAS DE LA EJECUCIÓN DE OPERACIONES DESDE UN CLIENTE REST

- Añadir un cliente del banco o crear perfil de usuario de banco.

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://localhost:8080/miBancoUPM/api/clientes
- Body:** JSON (application/json)
- Body Content:**

```
1 - {
2   "DNI": "2873872832",
3   "apellido": "Broncano",
4   "domicilio": "Jaén,88",
5   "nombre": "David",
6   "telefono": "82398293"
7 }
```

- Response Status:** 201 Created
- Response Time:** 133 ms
- Response Size:** 353 B

- Ver los datos básicos de un cliente.

The screenshot shows the Postman interface with the following details:

- Request Method:** GET
- URL:** http://localhost:8080/miBancoUPM/api/clientes/11
- Authorization:** Inherit auth from parent
- Body:** JSON (application/json)
- Body Content:**

```
1 - {
2   "DNI": "2873872832",
3   "apellido": "Broncano",
4   "cuentas_Bancarias": [],
5   "domicilio": "Jaén,88",
6   "id": 11,
7   "telefono": [],
8   "nombre": "David",
9   "telefono": "82398293"
10 }
```

- Response Status:** 200 OK
- Response Time:** 42 ms
- Response Size:** 260 B

- Cambiar datos básicos del cliente.

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- URL:** <http://localhost:8080/miBancoUPM/api/clientes/11>
- Body (JSON):**

```

1 < {
2   "DNI": "66666666",
3   "apellido": "Ya no es Broncano",
4   "cuentas_Bancarias": [],
5   "domicilio": "Carabanchel,008",
6   "movimientos": [],
7   "nombre": "David Arturo",
8   "telefono": "35454593"
9 }

```
- Response Status:** 200 OK
- Time:** 440 ms
- Size:** 351 B

The screenshot shows the Postman interface with the following details:

- Request Method:** GET
- URL:** <http://localhost:8080/miBancoUPM/api/clientes/11>
- Headers:**

Key	Value	Description
Accept	application/json	
Content-Type	application/json	
- Body (JSON):**

```

1 < [
2   {
3     "DNI": "66666666",
4     "apellido": "Ya no es Broncano",
5     "cuentas_Bancarias": [],
6     "domicilio": "Carabanchel,008",
7     "id": 11,
8     "movimientos": [],
9     "nombre": "David Arturo",
10    "telefono": "35454593"
11  }
12 ]

```
- Response Status:** 200 OK
- Time:** 40 ms
- Size:** 284 B

- Borrar todos los datos de un cliente, siempre y cuando ya no tuviera cuentas abiertas.

Intentamos borrar un cliente que aún tiene cuentas abiertas, y como es de esperar, no nos lo permite:

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/miBancoUPM/api/clientes/6`. The Headers tab shows `Accept: application/json` and `Content-Type: application/json`. The Body tab is empty. The response status is `412 Precondition Failed` with the message `No se puede eliminar cliente ya que tiene cuentas abiertas`.

Repetimos la operación con un cliente sin ninguna cuenta abierta:

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/miBancoUPM/api/clientes/11`. The Headers tab shows `Accept: application/json` and `Content-Type: application/json`. The Body tab is empty. The response status is `204 No Content`.

The screenshot shows the Postman interface with a failed GET request to `http://localhost:8080/miBancoUPM/api/clientes/11`. The response status is 404 Not Found. The Headers tab shows `Accept: application/json` and `Content-Type: application/json`.

- Crear una cuenta bancaria para un cliente.

Creamos cuenta con saldo:

The screenshot shows a successful POST request to `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias`. The response status is 201 Created. The JSON body sent was:

```

1 {
2   "nombreDueño": "Cristiano",
3   "apellidoDueño": "Ronaldo",
4   "saldo": 500000
5 }
    
```

De nuevo creamos cuenta, pero con saldo 0:

The screenshot shows a successful POST request to `http://localhost:8080/miBancoUPM/api/clientes/9/cuentas_bancarias`. The response status is 201 Created. The JSON body sent was:

```

1 {
2   "nombreDueño": "Mariano",
3   "apellidoDueño": "Rajoy",
4   "saldo": 0
5 }
    
```

- Eliminar la cuenta bancaria (si esta tiene saldo 0).

Intentamos eliminar cuenta sin saldo 0 y, como debe ser, no nos lo permite:

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20
- Headers:**
 - Accept: application/json
 - Content-Type: application/json
- Body:** (Empty)
- Response Status:** 412 Precondition Failed
- Response Body:** "No se puede eliminar cuenta: saldo mayor de 0."

Eliminamos cuenta con saldo 0:

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://localhost:8080/miBancoUPM/api/clientes/9/cuentas_bancarias/21
- Headers:**
 - Content-Type: application/json
- Body:**

```
{
  "nombreDueño": "Mariano",
  "apellidoDueño": "Rajoy",
  "saldo": 0
}
```
- Response Status:** 204 No Content
- Response Body:** (Empty)

- Realizar una transferencia entre cuentas (pueden ser suyas o de otros usuarios), estas transferencias tendrán que actualizar el saldo en ambas cuentas. Una transferencia tendrá se realizará correctamente si hay saldo suficiente para realizar la transferencia.

Situación inicial del cliente que envía dinero en la transferencia:

My Workspace

GET http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9

Headers (2)

Key	Value	Description
Accept	application/json	
Content-Type	application/json	

Body

```
{
  "apellidoDueño": "Gómez",
  "id": 9,
  "idCliente": 6,
  "nombreDueño": "Paco",
  "saldo": 570
}
```

Situación inicial del cliente que recibe dinero en la transferencia:

My Workspace

GET http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7

Headers (2)

Key	Value	Description
Accept	application/json	
Content-Type	application/json	

Body

```
{
  "apellidoDueño": "Ramiro",
  "id": 7,
  "idCliente": 3,
  "nombreDueño": "Emilio",
  "saldo": 20
}
```

Transferencia:

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/miBancoUP/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias`. The request body is JSON-formatted with the following data:

```

1: {
2:   "fecha": "2020-04-14 17:43:00",
3:   "saldoTransferido": 30,
4:   "cuentaRepite": 7
5: }

```

The response status is 201 Created, and the response body contains the transferred account's updated information:

```

1: {
2:   "cuentaRealiza": 9,
3:   "cuentaRepite": 7,
4:   "fecha": "2020-04-14 17:43:00",
5:   "id": 21,
6:   "saldoTransferido": 30
7: }

```

Tras la transferencia, situación del cliente que enviaba el dinero:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUP/api/clientes/6/cuentas_bancarias/9`. The request includes headers for Accept and Content-Type set to application/json. The response status is 200 OK, and the response body is:

```

1: {
2:   "apellidoDuenio": "Gomez",
3:   "id": 9,
4:   "idCliente": 6,
5:   "nombreDuenio": "Paco",
6:   "saldo": 540
7: }

```

Tras la transferencia, situación del cliente que recibía el dinero:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUP/api/clientes/3/cuentas_bancarias/7`. The request includes headers for Accept and Content-Type set to application/json. The response status is 200 OK, and the response body is:

```

1: {
2:   "apellidoDuenio": "Bamiro",
3:   "id": 7,
4:   "idCliente": 3,
5:   "nombreDuenio": "Emilio",
6:   "saldo": 50
7: }

```

- En casos excepcionales, el administrador podría eliminar una transferencia y por tanto actualizar los saldos de cuentas.

La transferencia mostrada anteriormente tenía un id = 21, que es lo que usaremos para realizar este apartado. Además, fijándonos en el ejemplo anterior también podemos ver la situación en la que se encuentran los clientes (receptor y emisor) antes de la eliminación de la transferencia.

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/21
- Headers:**
 - Accept: application/json
 - Content-Type: application/json
- Body:** (Empty)
- Status:** 204 No Content
- Time:** 202 ms
- Size:** 64 B

Tras la eliminación de la transferencia, el cliente emisor queda así:

The screenshot shows the Postman interface with the following details:

- Request Method:** GET
- URL:** http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9
- Headers:**
 - Accept: application/json
 - Content-Type: application/json
- Body:** (Pretty JSON output)

```

1 {
2   "apellidoDueño": "Gómez",
3   "id": 9,
4   "idCliente": 6,
5   "nombreDueño": "Paco",
6   "saldo": 578
7 }
```
- Status:** 200 OK
- Time:** 79 ms
- Size:** 189 B

El cliente receptor queda de la siguiente manera:

My Workspace ▾

History Collections Clear all

GET http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7 Headers (2) Body Pre-request Script Tests

Key	Value	Description	...	Bulk Edit	Presets ▾
<input checked="" type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Content-Type	application/json				
New key	Value	Description			

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON ↗

```

1 [
2   {
3     "apellidoDuenio": "Ramiro",
4     "id": 7,
5     "idCliente": 3,
6     "nombreDuenio": "Emilio",
7     "saldo": 20
8   }
9 ]

```

Status: 200 OK Time: 203 ms Size: 191 B

- Realizar una retirada de efectivo, y por tanto actualizará el saldo disponible. Al igual que en el caso de la transferencia, para realizar correctamente una retirada, debe haber saldo suficiente en la cuenta.

Situación del cliente que empleamos para este ejemplo:

My Workspace ▾

History Collections Clear all

GET http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7 Headers (2) Body Pre-request Script Tests

Key	Value	Description	...	Bulk Edit	Presets ▾
<input checked="" type="checkbox"/> Accept	application/json				
<input checked="" type="checkbox"/> Content-Type	application/json				
New key	Value	Description			

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON ↗

```

1 [
2   {
3     "apellidoDuenio": "Ramiro",
4     "id": 7,
5     "idCliente": 3,
6     "nombreDuenio": "Emilio",
7     "saldo": 20
8   }
9 ]

```

Status: 200 OK Time: 203 ms Size: 191 B

Probamos en primer lugar a intentar retirarle más saldo del que dispone:

The screenshot shows the Postman interface with a failed API call. The URL is `http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7/movimientos/retiradas_efectivo`. The request method is POST. The body contains the following JSON:

```
1: {  
2:   "fecha": "2020-08-20 19:43:00",  
3:   "saldoRetirado": 30  
4: }
```

The response status is 412 Precondition Failed, with a message: "No dispone del saldo necesario".

Ahora hacemos correctamente la retirada de efectivo:

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7/movimientos/retiradas_efectivo`. The request method is POST. The body contains the following JSON:

```
1: {  
2:   "fecha": "2020-08-20 19:43:00",  
3:   "saldoRetirado": 10  
4: }
```

The response status is 201 Created, with a JSON response indicating the new withdrawal record:

```
1: {  
2:   "cuenta": 7,  
3:   "fecha": "2020-08-20 19:43:00",  
4:   "id": 25,  
5:   "saldoRetirado": 10  
6: }
```

El cliente queda así finalmente:

The screenshot shows the Postman interface with a successful response. The URL is `http://localhost:8080/miBancoUPM/api/clientes/3/cuentas_bancarias/7`. The response body is a JSON object:

```

1 - [
2   {
3     "id": 3,
4     "cliente": 3,
5     "nombreDueño": "Emilio",
6     "saldo": 10,
7     "apellidoDueño": "Ramiro"
8   }
9 ]

```

- Obtener un listado de todas las transferencias emitidas de una cuenta personal.

Además, esta lista debe permitir la opción de ser filtrada por fecha o limitar la cantidad de información obtenida por número de movimientos. (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

Listado sin ningún filtro:

The screenshot shows the Postman interface with a successful response. The URL is `http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias`. The response body is a JSON object:

```

1 - {
2   "transferencias": [
3     {
4       "rel": "self",
5       "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/7"
6     },
7     {
8       "rel": "self",
9       "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/5"
10    },
11    {
12      "rel": "self",
13      "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/8"
14    }
15  ]
16 }

```

Listado con el filtro de fecha:

The screenshot shows the Postman interface with a 'History' tab selected. A recent request is displayed with the following details:

- Method: GET
- URL: `http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias?desde=2020-01-01 00:00:00&hasta=2023-02-02 00:00:00`
- Authorization header: `authorization: [REDACTED]`
- Status: 200 OK
- Time: 133 ms
- Size: 249 B

The response body is a JSON object:

```
1  {
2   "transferencias": [
3     {
4       "rel": "self",
5       "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/5"
6     }
7   ]
8 }
```

Listado por cantidad de información:

The screenshot shows the Postman interface with a 'History' tab selected. A recent request is displayed with the following details:

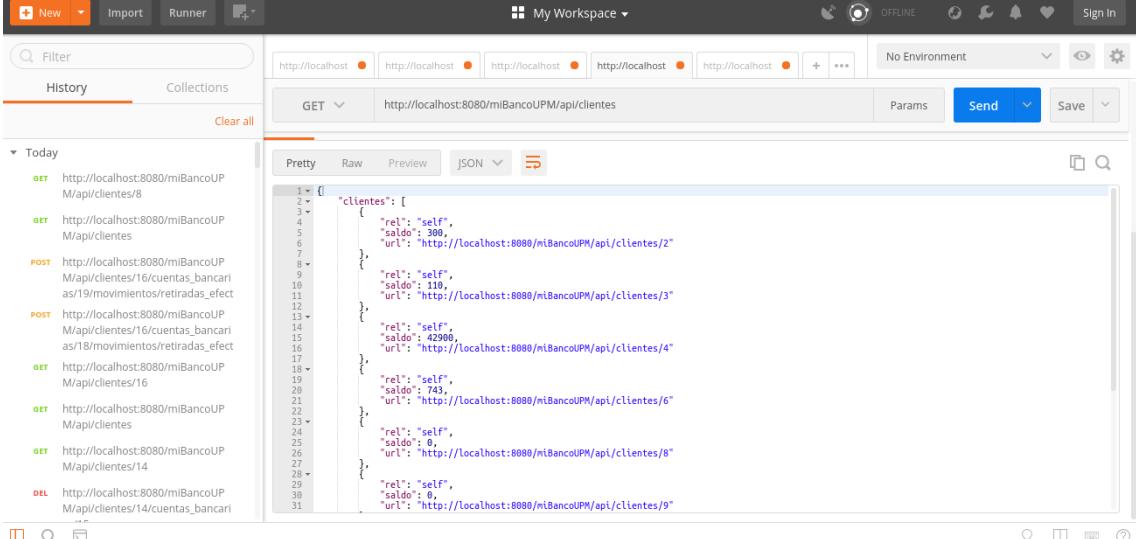
- Method: GET
- URL: `http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias?as?limite1=1&limite2=3`
- Authorization header: `authorization: [REDACTED]`
- Status: 200 OK
- Time: 130 ms
- Size: 369 B

The response body is a JSON object:

```
1  {
2   "transferencias": [
3     {
4       "rel": "self",
5       "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/5"
6     },
7     {
8       "rel": "self",
9       "url": "http://localhost:8080/miBancoUPM/api/clientes/6/cuentas_bancarias/9/movimientos/transferencias/7"
10    }
11  ]
12 }
```

- El banco debe permitir obtener un listado de todos sus usuarios y su saldo actual. Además, esta lista debe permitir la opción de ser filtrada por cantidad de saldo o limitar la cantidad de información obtenida.

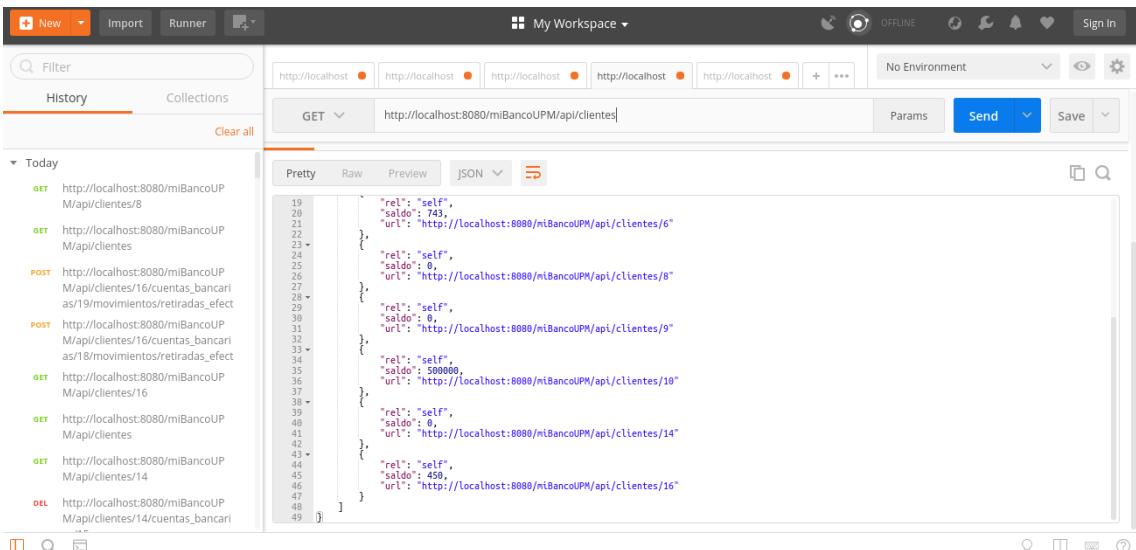
En primer lugar, obtenemos el listado sin filtros:



```

1 [ {
2   "clientes": [
3     {
4       "rel": "self",
5       "saldo": 300,
6       "url": "http://localhost:8080/miBancoUPM/api/clientes/2"
7     },
8     {
9       "rel": "self",
10      "saldo": 110,
11      "url": "http://localhost:8080/miBancoUPM/api/clientes/3"
12    },
13    {
14      "rel": "self",
15      "saldo": 42900,
16      "url": "http://localhost:8080/miBancoUPM/api/clientes/4"
17    },
18    {
19      "rel": "self",
20      "saldo": 743,
21      "url": "http://localhost:8080/miBancoUPM/api/clientes/6"
22    },
23    {
24      "rel": "self",
25      "saldo": 6,
26      "url": "http://localhost:8080/miBancoUPM/api/clientes/8"
27    },
28    {
29      "rel": "self",
30      "saldo": 6,
31      "url": "http://localhost:8080/miBancoUPM/api/clientes/9"
32    }
33  ],
34  "meta": {
35    "total": 10
36  }
37 }
38 ]
39 ]
40 ]
41 ]
42 ]
43 ]
44 ]
45 ]
46 ]
47 ]
48 ]
49 ]

```

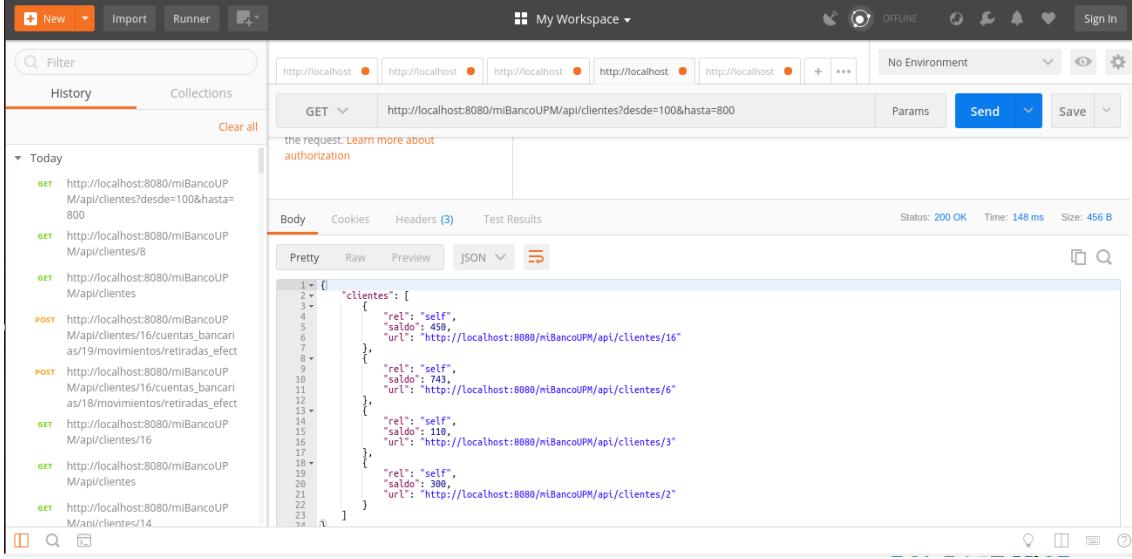



```

19 [ {
20   "clientes": [
21     {
22       "rel": "self",
23       "saldo": 42900,
24       "url": "http://localhost:8080/miBancoUPM/api/clientes/6"
25     },
26     {
27       "rel": "self",
28       "saldo": 6,
29       "url": "http://localhost:8080/miBancoUPM/api/clientes/8"
30     },
31     {
32       "rel": "self",
33       "saldo": 6,
34       "url": "http://localhost:8080/miBancoUPM/api/clientes/9"
35     },
36     {
37       "rel": "self",
38       "saldo": 500000,
39       "url": "http://localhost:8080/miBancoUPM/api/clientes/10"
40     },
41     {
42       "rel": "self",
43       "saldo": 6,
44       "url": "http://localhost:8080/miBancoUPM/api/clientes/14"
45     },
46     {
47       "rel": "self",
48       "saldo": 459,
49       "url": "http://localhost:8080/miBancoUPM/api/clientes/16"
49     }
50   ],
51   "meta": {
52     "total": 10
53   }
54 }
55 ]
56 ]
57 ]
58 ]
59 ]
60 ]
61 ]
62 ]
63 ]
64 ]
65 ]
66 ]
67 ]
68 ]
69 ]
69 ]

```

Aplicamos el filtro de saldo:



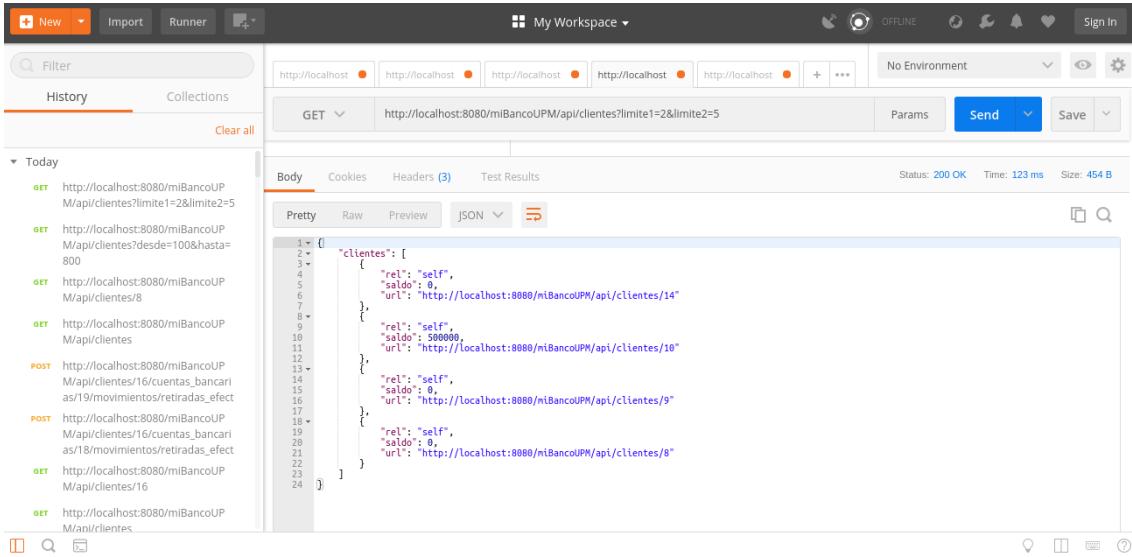
The screenshot shows the Postman interface with a history of requests on the left and a detailed view of a selected request on the right.

Request URL: `http://localhost:8080/miBancoUPM/api/clientes?desde=100&hasta=800`

Response Body (Pretty JSON):

```
[{"clientes": [{"rel": "self", "saldo": 450, "url": "http://localhost:8080/miBancoUPM/api/clientes/16"}, {"rel": "self", "saldo": 743, "url": "http://localhost:8080/miBancoUPM/api/clientes/6"}, {"rel": "self", "saldo": 110, "url": "http://localhost:8080/miBancoUPM/api/clientes/3"}, {"rel": "self", "saldo": 200, "url": "http://localhost:8080/miBancoUPM/api/clientes/2"}]}
```

Por último, el filtro de cantidad de información:



The screenshot shows the Postman interface with a history of requests on the left and a detailed view of a selected request on the right.

Request URL: `http://localhost:8080/miBancoUPM/api/clientes?limite1=2&limite2=5`

Response Body (Pretty JSON):

```
[{"clientes": [{"rel": "self", "saldo": 0, "url": "http://localhost:8080/miBancoUPM/api/clientes/14"}, {"rel": "self", "saldo": 500000, "url": "http://localhost:8080/miBancoUPM/api/clientes/10"}]}
```

- Un usuario puede consultar un listado con todas sus retiradas de efectivo realizadas. Esta lista debe permitir la opción de ser filtrada por cantidad de retirada y fecha de las retiradas.

Realizaremos este apartado en base al siguiente cliente:

```

1 {
2   "DNI": "77777777",
3   "apellido": "Ronaldo",
4   "cuentas_Bancarias": [
5     {
6       "apellidoDueocio": "Ronaldo",
7       "id": 20,
8       "idCliente": 10,
9       "nombreCuenta": "Cristiano",
10      "saldo": 432000
11    }
12  ],
13  "domicilio": "Madrid,7",
14  "id": 10,
15  "movimientos": [
16    {
17      "fecha": "2020-12-31 00:00:00.0",
18      "id": 0,
19      "idCliente": 10,
20      "idCuenta": 20,
21      "idMovimiento": 30,
22      "tipo": "Retirada_Efectivo"
23    },
24    {
25      "fecha": "2020-10-30 13:00:00.0",
26      "id": 0,
27      "idCliente": 10,
28      "idCuenta": 20,
29      "idMovimiento": 32,
30      "tipo": "Retirada_Efectivo"
31    }
32  ]
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }

```

```

13 {
14   "domicilio": "Madrid,7",
15   "movimientos": [
16    {
17      "fecha": "2020-12-31 00:00:00.0",
18      "id": 0,
19      "idCliente": 10,
20      "idCuenta": 20,
21      "idMovimiento": 30,
22      "tipo": "Retirada_Efectivo"
23    },
24    {
25      "fecha": "2020-10-30 13:00:00.0",
26      "id": 0,
27      "idCliente": 10,
28      "idCuenta": 20,
29      "idMovimiento": 32,
30      "tipo": "Retirada_Efectivo"
31    }
32  ]
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }

```

Listado sin filtros:

The screenshot shows the Postman interface with the following details:

- Header Bar:** My Workspace, OFFLINE, various icons.
- Left Sidebar:** History (selected), Collections, Clear all.
- Request Section:**
 - Method: GET
 - URL: `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo`
 - Description: The authorization header will be automatically generated when you send the request. Learn more about authorization.
 - Status: 200 OK, Time: 60 ms, Size: 514 B
- Body Section:**
 - Pretty (selected), Raw, Preview, JSON.
 - JSON Response (partial):

```

1  {
2   "retiradas_Efectivo": [
3    {
4     "rel": "self",
5     "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/30"
6    },
7    {
8     "rel": "self",
9     "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/31"
10   },
11   {
12    "rel": "self",
13    "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/32"
14  }
15 ]
16 }
```

Filtrado por fecha:

The screenshot shows the Postman interface with the following details:

- Header Bar:** My Workspace, OFFLINE, various icons.
- Left Sidebar:** History (selected), Collections, Clear all.
- Request Section:**
 - Method: GET
 - URL: `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo?desde=2020-10-01 00:00:00&hasta=2020-12-31 00:00:00`
 - Description: The authorization header will be automatically generated when you send the request. Learn more about authorization.
 - Status: 200 OK, Time: 119 ms, Size: 387 B
- Body Section:**
 - Pretty (selected), Raw, Preview, JSON.
 - JSON Response (partial):

```

1  {
2   "retiradas_Efectivo": [
3    {
4     "rel": "self",
5     "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/30"
6    },
7    {
8     "rel": "self",
9     "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/32"
10   }
11 ]
12 }
```

Filtrado por cantidad de información:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo?cantidad=3000`. The response status is 200 OK, time 51 ms, and size 260 B. The JSON response body is as follows:

```

1: {
2:   "retiradas_Efectivo": [
3:     {
4:       "rel": "self",
5:       "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/30"
6:     }
7:   ]
8: }

```

Ambos filtros simultáneamente:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo?desde=2020-10-01 00:00:00&hasta=2020-12-31 00:00:00&cantidad=7000`. The response status is 200 OK, time 172 ms, and size 260 B. The JSON response body is as follows:

```

1: {
2:   "retiradas_Efectivo": [
3:     {
4:       "rel": "self",
5:       "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/32"
6:     }
7:   ]
8: }

```

- Debe permitir consultar todos los movimientos (transferencias y retiradas) realizados en todas sus cuentas. Esta lista debe permitir la opción de ser filtrada por cantidad y fecha de los movimientos.

Para la resolución de este apartado continuaremos trabajando con el cliente (*id = 10*) del apartado anterior.

Lista sin filtrado:

```

{
  "movimientos": [
    {
      "rel": "self",
      "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/31"
    }
  ]
}

```

Filtrado de fecha:

```

{
  "movimientos": [
    {
      "rel": "self",
      "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/20/movimientos/retiradas_efectivo/32"
    }
  ]
}

```

Filtrado por cantidad de información:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/movimientos?limite=1`. The response is displayed in JSON format, showing a single object with a key `movimientos` containing an array of movement objects. The array has one element, which is itself an object with keys `rel`, `self`, and `url`.

```

1  {
2     "movimientos": [
3         {
4             "rel": "self",
5             "url": "http://localhost:8080/miBancoUPM/api/clientes/10/cuentas_bancarias/movimientos/retradas_efectivo/30"
6         }
7     ]
8 }

```

- Consultar fácilmente la descripción necesaria para una aplicación móvil de un banco que queremos realizar, que muestre los datos básicos de un cliente, sus cuentas y su saldo, los últimos 10 movimientos realizados en todas sus cuentas (identificando sus cuentas).

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUPM/api/clientes/3`. The response is displayed in JSON format, showing an array of client objects. The first object has a key `cuentas_Bancarias` which is an array of account objects. One account object is expanded to show its details, including the owner's name and ID, the client's ID, and the account's balance.

```

1  [
2     {
3         "DNI": "54742365",
4         "apellido": "Ramiro",
5         "cuentas_Bancarias": [
6             {
7                 "apellidoDueño": "Ramiro",
8                 "id": 3,
9                 "idCliente": 3,
10                "nombreDueño": "Emilio",
11                "saldo": 100
12            },
13            {
14                "apellidoDueño": "Ramiro",
15                "id": 7,
16                "idCliente": 3,
17                "nombreDueño": "Emilio",
18                "saldo": 10
19            },
20            {
21                "apellidoDueño": "Cadiz,45",
22                "id": 3,
23                "movimientos": [
24                    {
25                        "fecha": "2020-12-01 21:00:09.0",
26                        "id": 0,
27                        "idCliente": 3,
28                        "idCuenta": 3,
29                        "idMovimiento": 1,
30                        "tipo": "Transferencia"
31                    }
32                ]
33            }
34        ]
35    }
36 ]

```

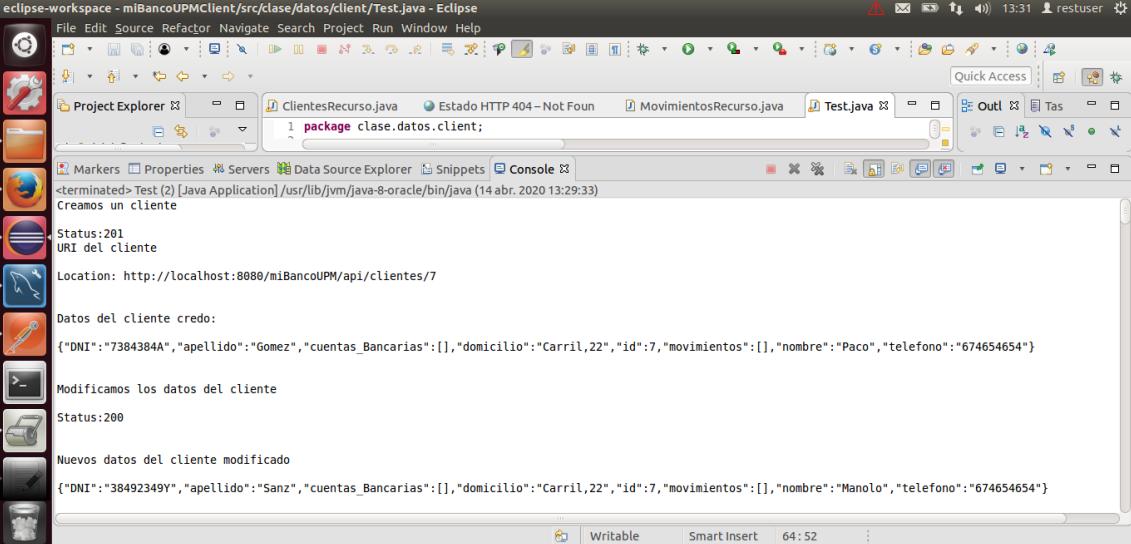
The screenshot shows the Postman interface with a GET request to `http://localhost:8080/miBancoUPM/api/clientes/3`. The response is displayed in JSON format, showing an array of client objects. The first object has a key `cuentas_Bancarias` which is an array of account objects. One account object is expanded to show its details, including the owner's name and ID, the client's ID, and the account's balance. The expanded account object also includes a key `movimientos` which is an array of movement objects. One movement object is expanded to show its details, including the date, ID, client ID, account ID, movement ID, and type.

```

28     "movimientos": [
29         {
30             "idMovimiento": 1,
31             "tipo": "Transferencia"
32         },
33         {
34             "fecha": "2020-12-01 21:00:09.0",
35             "id": 0,
36             "idCliente": 3,
37             "idCuenta": 3,
38             "idMovimiento": 2,
39             "tipo": "Retirada_Efectivo"
40         },
41         {
42             "fecha": "2020-12-01 21:00:09.0",
43             "id": 0,
44             "idCliente": 3,
45             "idCuenta": 3,
46             "idMovimiento": 3,
47             "tipo": "Transferencia"
48         },
49         {
50             "fecha": "2020-08-20 19:43:00.0",
51             "id": 0,
52             "idCliente": 3,
53             "idCuenta": 7,
54             "idMovimiento": 25,
55             "tipo": "Retirada_Efectivo"
56         }
57     ],
58     "nombre": "Emilio",
59     "telefono": "4564564566"
60   }
61 ]

```

3. CAPTURAS DE LA EJECUCIÓN DEL CLIENTE DE PRUEBA



```
eclipse-workspace - miBancoUPMClient/src/clase/datos/client/Test.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer ClientesRecurso.java Estado HTTP 404 – Not Found MovimientosRecurso.java Test.java
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
Creamos un cliente
Status:201
URI del cliente
Location: http://localhost:8080/miBancoUPM/api/clientes/7
Datos del cliente creado:
{"DNI":"7384384A","apellido":"Gomez","cuentas_Bancarias":[],"domicilio":"Carril,22","id":7,"movimientos":[],"nombre":"Paco","telefono":"674654654"}
Modificamos los datos del cliente
Status:200
Nuevos datos del cliente modificado
{"DNI":"38492349Y","apellido":"Sanz","cuentas_Bancarias":[],"domicilio":"Carril,22","id":7,"movimientos":[],"nombre":"Manolo","telefono":"674654654"}
```

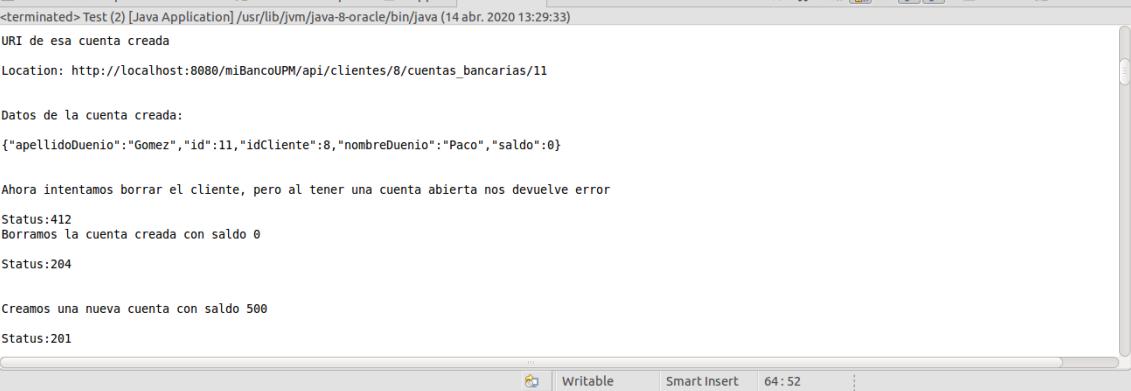


```
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
Nuevos datos del cliente modificado
{"DNI":"38492349Y","apellido":"Sanz","cuentas_Bancarias":[],"domicilio":"Carril,22","id":7,"movimientos":[],"nombre":"Manolo","telefono":"674654654"}

Eliminamos el cliente creado anteriormente
Status:204

Creamos de nuevo otro cliente
Creamos una cuenta bancaria para ese cliente con saldo 0
Status:201

URI de esa cuenta creada
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/11
```



```
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
URI de esa cuenta creada
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/11

Datos de la cuenta creada:
{"apellidoDuenio":"Gomez","id":11,"idCliente":8,"nombreDuenio":"Paco","saldo":0}

Ahora intentamos borrar el cliente, pero al tener una cuenta abierta nos devuelve error
Status:412
Borramos la cuenta creada con saldo 0
Status:204

Creamos una nueva cuenta con saldo 500
Status:201
```

```

Markers Properties Servers Data Source Explorer Snippets Console
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
Creamos una nueva cuenta con saldo 500
Status:201

Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12

{"apellidoDuenio":"Gomez","id":12,"idCliente":8,"nombreDuenio":"Paco","saldo":500}

Intentamos borrar esa cuenta pero al tener saldo distinto de 0 no es posible
Status:412

Creamos otra cuenta para el mismo cliente con saldo 300
Realizamos transferencias entre estas cuentas.

Primer caso: no se puede realizar la transferencia ya que no hay suficiente saldo para ella
Tratamos de realizar una transferencia de 500 la segunda cuenta creada(300 de saldo)
Status:412

Primer caso: no se puede realizar la transferencia ya que no hay suficiente saldo para ella
Tratamos de realizar una transferencia de 500 la segunda cuenta creada(300 de saldo)
Status:412

Como vemos, nos ha dado código de error y como podemos observar la cuenta sigue teniendo el mismo saldo:
{"apellidoDuenio":"Gomez","id":13,"idCliente":8,"nombreDuenio":"Paco","saldo":300}

Segundo caso: se realiza una transferencia de 50 con suficiente saldo
Status:201

URI de la transferencia
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9

URI de la transferencia
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9

Como observamos han cambiado el saldo de las dos cuentas
{"apellidoDuenio":"Gomez","id":12,"idCliente":8,"nombreDuenio":"Paco","saldo":550}

{"apellidoDuenio":"Gomez","id":13,"idCliente":8,"nombreDuenio":"Paco","saldo":250}

Vemos los datos de esa transferencia en concreto
{"cuentaRealiza":13,"cuentaRecibe":12,"fecha":"2020-01-01 10:00:00.0","id":9,"saldoTransferido":50}

Tercer caso: realizamos la misma transferencia (con distinta hora), vemos que ha cambiado, la borramos y vemos que las cuentas han vuelto a tener su antiguo saldo
Status:201

Status:201

Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/10

Observamos que los saldos de las dos cuentas han cambiado
{"apellidoDuenio":"Gomez","id":12,"idCliente":8,"nombreDuenio":"Paco","saldo":600}

{"apellidoDuenio":"Gomez","id":13,"idCliente":8,"nombreDuenio":"Paco","saldo":200}

Ahora borramos esa transferencia y las cuentas tienen el mismo saldo que antes de realizarla
Status:204

{"apellidoDuenio":"Gomez","id":12,"idCliente":8,"nombreDuenio":"Paco","saldo":550}

```

```

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
{"apellidoDuenio":"Gomez","id":12,"idCliente":8,"nombreDuenio":"Paco","saldo":550}

>{"apellidoDuenio":"Gomez","id":13,"idCliente":8,"nombreDuenio":"Paco","saldo":250}

Como hemos podido comprobar los saldos han vuelto a ser los mismos

Ahora probaremos las retiradas de efectivo. Como con las transferencias hay dos casos:
Primer caso: realizamos una retirada de efectivo de 1000 en una cuenta en la que no hay suficiente saldo
Status:412

Como podemos observar nos devuelve un codigo de status de error
Vamos ahora con el caso de una retirada de efectivo de 5 que si se puede realizar
Status:201

URI de la Retirada de Efectivo realizada
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
URI de la Retirada de Efectivo realizada
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14

Vemos los datos de esa retirada en concreto
{"cuenta":13,"fecha":"2021-01-01 10:28:00.0","id":14,"saldoRetirado":5}

Y ahora vemos que se ha actualizado el saldo de esa cuenta:
{"apellidoDuenio":"Gomez","id":13,"idCliente":8,"nombreDuenio":"Paco","saldo":245}

Obtenemos una lista de todas las transferencias emitidas por una cuenta. Para ello crearemos varias transferencias con distintos años, una del 2000 y otra en 2025
URI de la Transferencia del año 2000
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/11

URI de la Transferencia del año 2025
Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
URI de la Transferencia del año 2025
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/12

Obtenemos la lista de todas las transferencias, es decir, sin filtros
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/11
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/12

Como se puede observar, nos muestra las tres transferencias que hemos realizado. Ahora filtraremos por número de movimientos, le pediremos solamente un movimiento
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/12

Ahora pediremos desde el primer movimiento hasta el segundo, lo que nos debería mostrar todos menos el ultimo realizado
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/11

Ahora pediremos las transferencias hechas desde 2020 hasta 2025, lo que nos debería mostrar solo las dos hechas en ese periodo

```

```

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
Ahora pediremos las transferencias hechas desde 2020 hasta 2025, lo que nos deberia mostrar solo las dos hechas en ese periodo
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9

Ahora pediremos las transferencias hechas desde 2020 hasta 2025 con limite de 1, lo que nos deberia mostrar solo una de las dos hechas durante ese periodo
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9
Ahora haremos lo mismo pero con retiradas de efectivo. Para ello crearemos varias retiradas con distintos años, una del 2000 de 50 y otra en 2025 de 2

Status:201

URI de la Retirada de Efectivo realizada en 2000
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/15

Status:201

URI de la Retirada de Efectivo realizada en 2025
.

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
URI de la Retirada de Efectivo realizada en 2025
Location: http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16

Obtenemos la lista de todas las retiradas, es decir, sin filtros
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/15
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16

Como se puede observar, nos muestra las tres retiradas que hemos realizado. Ahora filtraremos por cantidad de dinero, le pediremos solamente las retiradas mayor
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/15

Ahora pediremos entre 1 y 10, lo que nos deberia mostrar todos menos el de 50
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14

Ahora pediremos las retiradas hechas desde 2020 hasta 2026, lo que nos deberia mostrar solo las dos hechas en ese periodo
.

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/11

Ahora pediremos las retiradas hechas desde 2020 hasta 2026, lo que nos deberia mostrar solo las dos hechas en ese periodo
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14

Ahora pediremos las retiradas hechas desde 2020 hasta 2026 que sea mayor de 4, lo que nos deberia mostrar solo una de las dos hechas durante ese periodo
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
Ahora consultaremos todos los movimientos

Obtenemos la lista de todos los movimientos, es decir, sin filtros
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/15
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12/movimientos/transferencias/11
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/11
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12/movimientos/transferencias/9
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12/movimientos/transferencias/12
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/12

Como se puede observar, nos muestra los tres movimientos que hemos realizado. Ahora filtraremos por numero de movimientos, le pediremos solamente un movimiento.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12/movimientos/transferencias/12

Ahora pediremos desde el primer movimiento hasta el segundo, lo que nos deberia mostrar todos menos el ultimo realizado
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/12
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/16

Ahora pediremos los movimientos hechos desde 2020 hasta 2025, lo que nos deberia mostrar solo las hechas en ese periodo
.
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/12/movimientos/transferencias/9
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/transferencias/9

Ahora pediremos los movimientos hechos desde 2020 hasta 2025 con limite de 1, lo que nos deberia mostrar solo una de las dos hechas durante ese periodo
.
```

```

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
Ahora pediremos los movimientos hechos desde 2020 hasta 2025 con limite de 1, lo que nos deberia mostrar solo una de las dos hechas durante ese periodo
http://localhost:8080/miBancoUPM/api/clientes/8/cuentas_bancarias/13/movimientos/retiradas_efectivo/14
Ahora miraremos todos los clientes con sus respectivos saldos. Primero sin filtros:
http://localhost:8080/miBancoUPM/api/clientes//2
saldo:300

http://localhost:8080/miBancoUPM/api/clientes//3
saldo:120

http://localhost:8080/miBancoUPM/api/clientes//4
saldo:42900

http://localhost:8080/miBancoUPM/api/clientes//6
saldo:743

http://localhost:8080/miBancoUPM/api/clientes//8
saldo:743

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
http://localhost:8080/miBancoUPM/api/clientes//8
saldo:743

Ahora miraremos todos los clientes con saldos mayor a 100:
http://localhost:8080/miBancoUPM/api/clientes//8
saldo:743

http://localhost:8080/miBancoUPM/api/clientes//6
saldo:743

http://localhost:8080/miBancoUPM/api/clientes//4
saldo:42900

http://localhost:8080/miBancoUPM/api/clientes//3
saldo:120

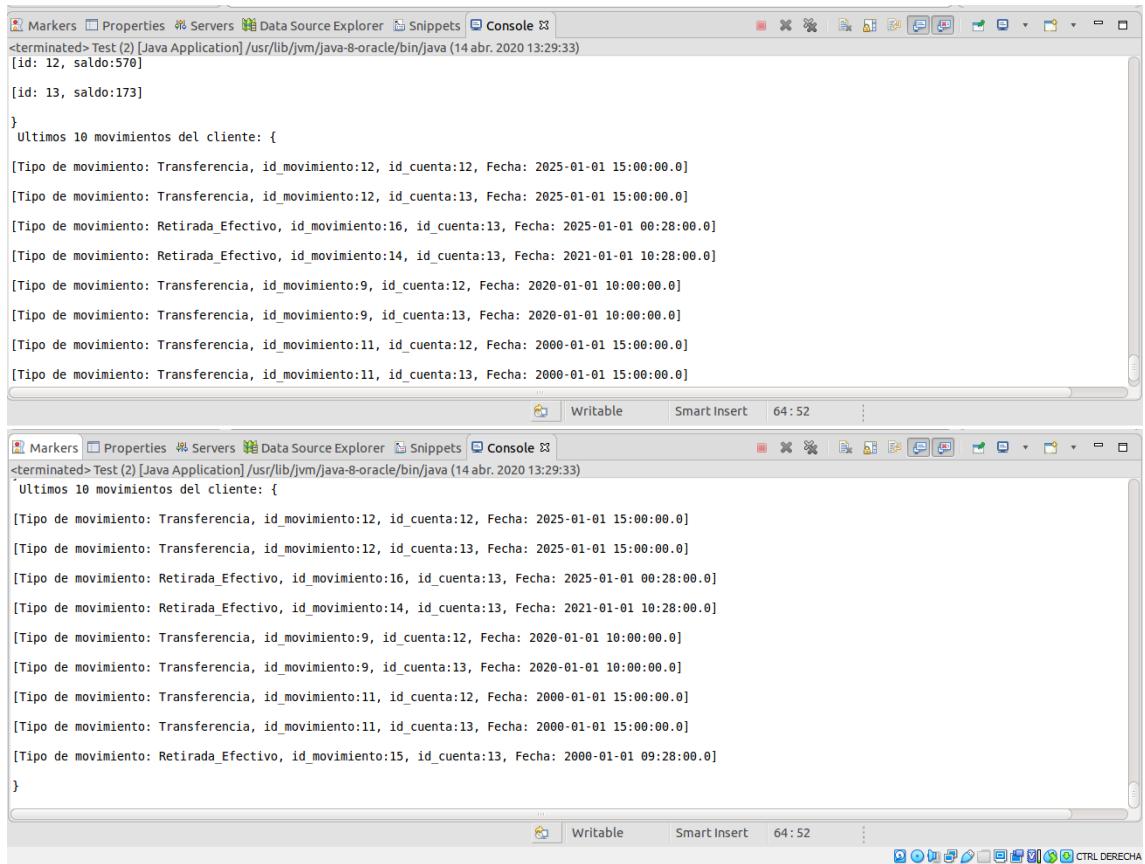
http://localhost:8080/miBancoUPM/api/clientes//2
saldo:300

Markers Properties Servers Data Source Explorer Snippets Console 
<terminated> Test (2) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)
http://localhost:8080/miBancoUPM/api/clientes//2
saldo:300

Ahora queremos solo el primer cliente con mayor saldo que 100
http://localhost:8080/miBancoUPM/api/clientes//8
saldo:743

Ahora veremos los datos basicos de un cliente, sus cuentas, sus saldos y los ultimos movimientos realizados en todas sus cuentas.
Id del cliente: 8
Nombre del cliente: Paco
Apellido del cliente: Gomez
DNI del cliente: 7384384A
Domicilio del cliente: Carril,22
Telefono del cliente: 674654654
Cuentas Bancarias del cliente: {
    [id: 12, saldo:570]
    [id: 13, saldo:173]
}
Ultimos 10 movimientos del cliente: [
]

```



The image shows two side-by-side Eclipse IDE consoles. Both consoles have the following header:

Markers Properties Servers Data Source Explorer Snippets Console

Console tab is selected.

<terminated> Test (2) [Java Application] /usr/lib/vm/java-8-oracle/bin/java (14 abr. 2020 13:29:33)

The output in both consoles is identical, displaying the following Java code execution:

```
[id: 12, saldo:570]
[id: 13, saldo:173]
}
Ultimos 10 movimientos del cliente: {
[Tipo de movimiento: Transferencia, id_movimiento:12, id_cuenta:12, Fecha: 2025-01-01 15:00:00.0]
[Tipo de movimiento: Transferencia, id_movimiento:12, id_cuenta:13, Fecha: 2025-01-01 15:00:00.0]
[Tipo de movimiento: Retirada_Efectivo, id_movimiento:16, id_cuenta:13, Fecha: 2025-01-01 00:28:00.0]
[Tipo de movimiento: Retirada_Efectivo, id_movimiento:14, id_cuenta:13, Fecha: 2021-01-01 10:28:00.0]
[Tipo de movimiento: Transferencia, id_movimiento:9, id_cuenta:12, Fecha: 2020-01-01 10:00:00.0]
[Tipo de movimiento: Transferencia, id_movimiento:9, id_cuenta:13, Fecha: 2020-01-01 10:00:00.0]
[Tipo de movimiento: Transferencia, id_movimiento:11, id_cuenta:12, Fecha: 2000-01-01 15:00:00.0]
[Tipo de movimiento: Transferencia, id_movimiento:11, id_cuenta:13, Fecha: 2000-01-01 15:00:00.0]
```