

## Apéndice B: Validación de los servicios de la API.

**Autora: Valentina Goyeneche Calderón**

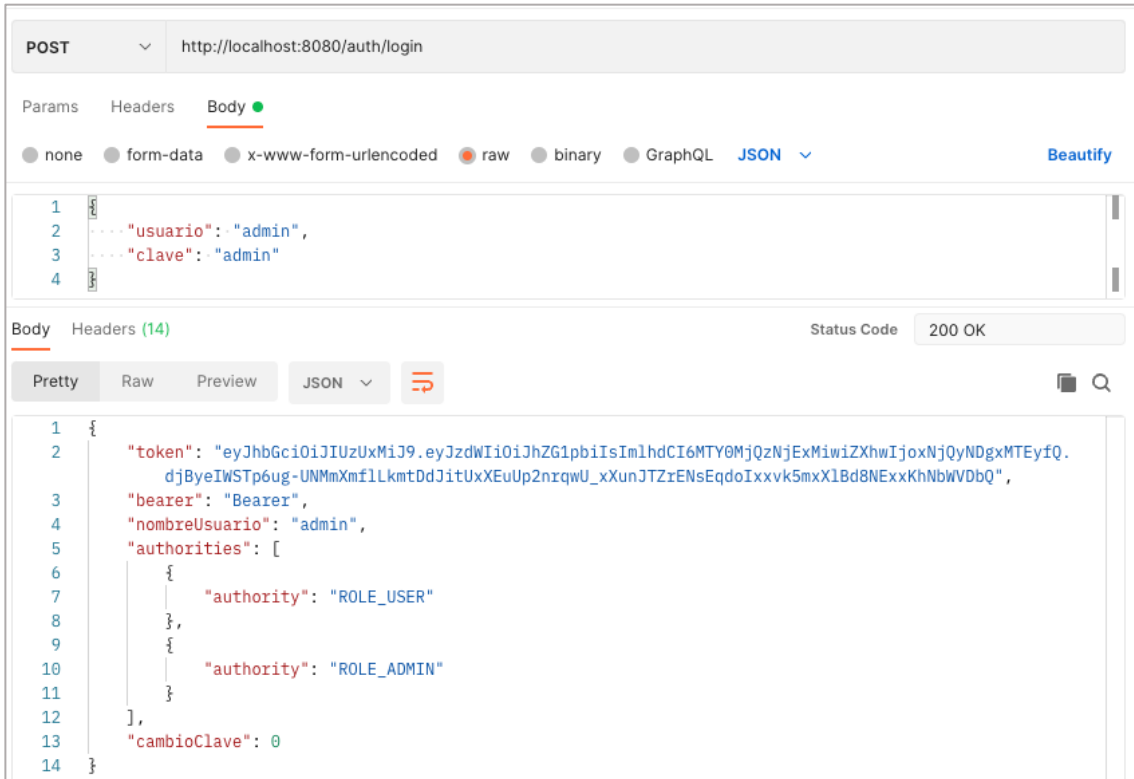
**Descripción:** En este apéndice se registran todos los servicios creados durante el desarrollo de la aplicación web, probando su funcionamiento por medio de la herramienta Postman. Se podrá observar el tipo de petición, los datos que procesa la petición y la respuesta que la aplicación envía al componente frontend. El documento está organizado por las entidades que se definieron en el proyecto: Usuario, Paciente, Cama, e Ingreso.

## Servicios

## 1. Usuario

### 1.1. Iniciar sesión

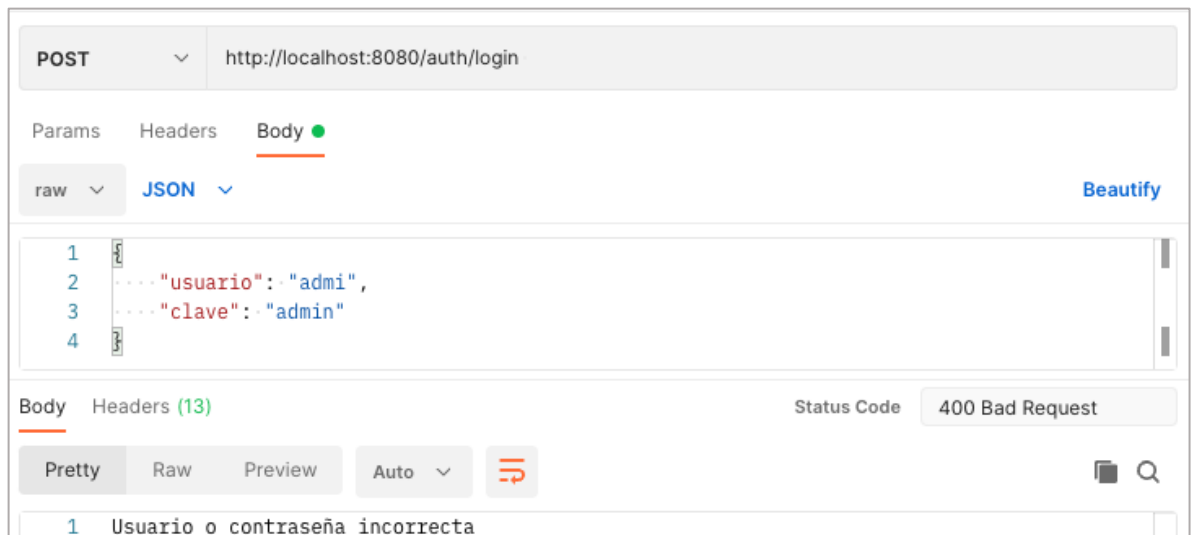
En el servicio de iniciar sesión, se deben ingresar los datos de usuario y su respectiva clave.



## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En el caso de que el usuario existe en el sistema y que su clave coincide se devuelve el objeto Usuario, donde el campo más importante es el token, que le asigna permisos al usuario en los diferentes procesos de la aplicación.

Un segundo caso es que el usuario haya ingresado erróneamente los datos, por lo que obtendrá un aviso y no podrá iniciar sesión.



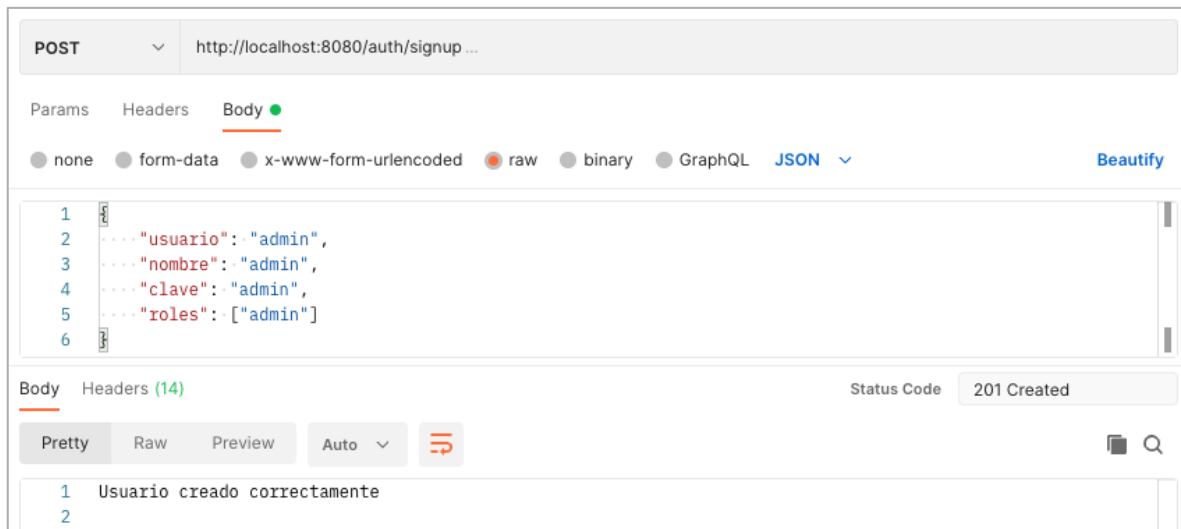
### 1.2. Registrarse

El servicio de registrar requiere de usuario, nombre del usuario, clave, y los roles que se le asignarán: superadmin, admin o user.

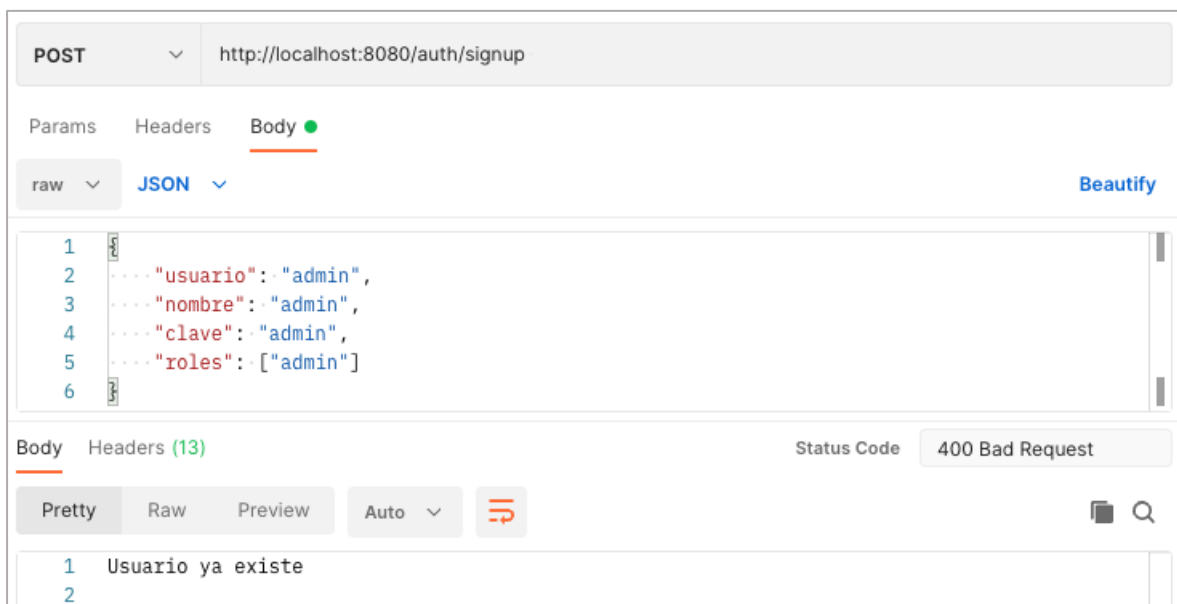
Es importante tener en cuenta que los usuarios solo pueden ser registrados por otro usuario con rol de *superdamin*.

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En este primer ejemplo, podemos ver que el usuario fue creado de forma satisfactoria.

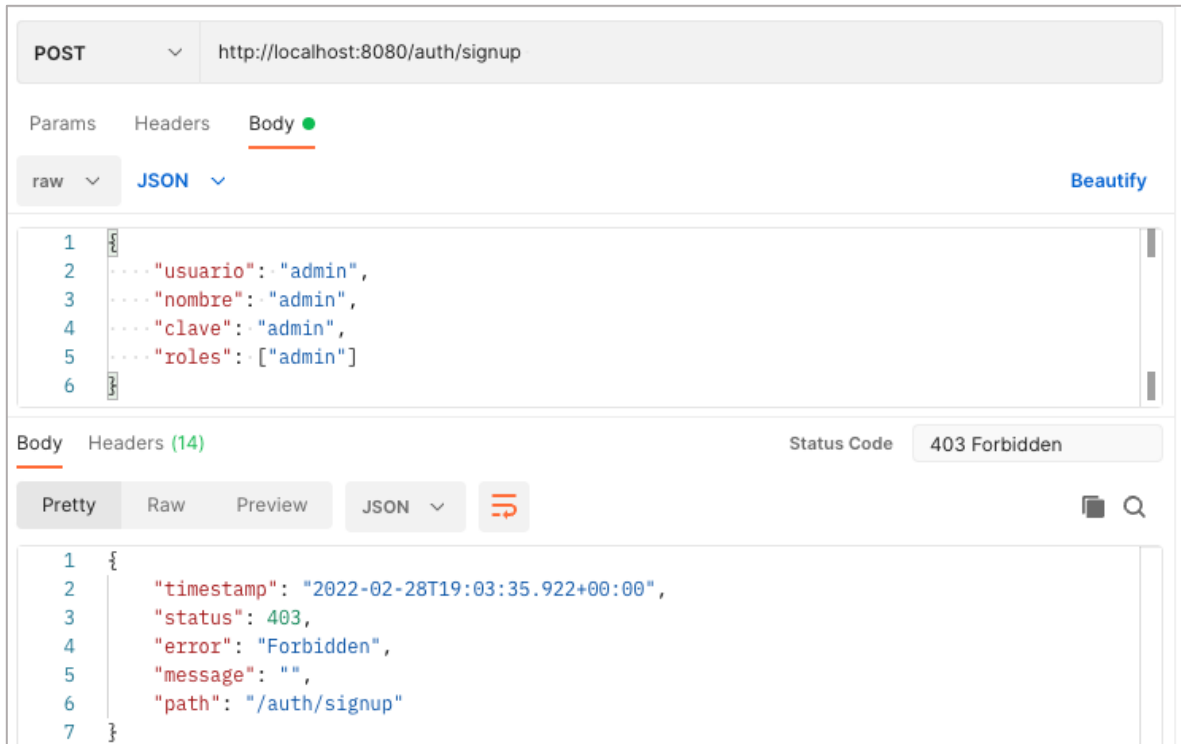


En un segundo caso, si el usuario ya existe dentro de la base de datos se obtiene una petición fallida, y se indica por medio de un mensaje de advertencia como podemos ver en la siguiente imagen. Para el ejemplo, intentaremos registrar otro usuario con username “admin” como el que hemos registrado anteriormente.



## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

Por último, en caso de que un usuario sin rol de *superadmin* intente llevar a cabo esta petición sería una solicitud no permitida.



The screenshot shows a REST client interface with a POST request to `http://localhost:8080/auth/signup`. The request body is a JSON object: `{ "usuario": "admin", "nombre": "admin", "clave": "admin", "roles": ["admin"] }`. The response status is `403 Forbidden`. The response body is a JSON object: `{ "timestamp": "2022-02-28T19:03:35.922+00:00", "status": 403, "error": "Forbidden", "message": "", "path": "/auth/signup" }`.

```
POST http://localhost:8080/auth/signup

Body
raw JSON Beautify

1 {
2   "usuario": "admin",
3   "nombre": "admin",
4   "clave": "admin",
5   "roles": ["admin"]
6 }
```

```
Body Headers (14) Status Code 403 Forbidden

Pretty Raw Preview JSON

1 {
2   "timestamp": "2022-02-28T19:03:35.922+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "",
6   "path": "/auth/signup"
7 }
```

### 1.3. Cambiar clave

El servicio de cambiar clave necesita dos argumentos dentro del mismo URL de la petición. Para este ejemplo, la clave antigua será “admin” y la clave nueva será “admin123”. Si los datos ingresados son correctos, el cambio de clave se realiza satisfactoriamente y se envía un mensaje de éxito.



The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/auth/cambiar-clave/old=admin/new=admin123...`. The response status is `200 OK`. The response body is a JSON object: `{ "mensaje": "Cambio de contraseña exitoso" }`.

```
PUT http://localhost:8080/auth/cambiar-clave/old=admin/new=admin123 ...

Params Headers Body

Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

1 {
2   "mensaje": "Cambio de contraseña exitoso"
3 }
4
```

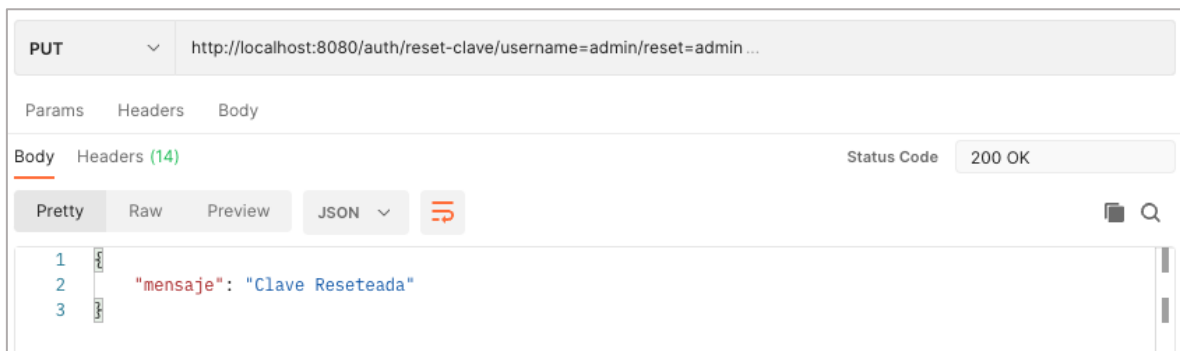
## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En caso de que la contraseña antigua no coincida con la que está almacenada en la base de datos, no se completa la operación y se envía un mensaje de advertencia.



### 1.4. Restablecer clave

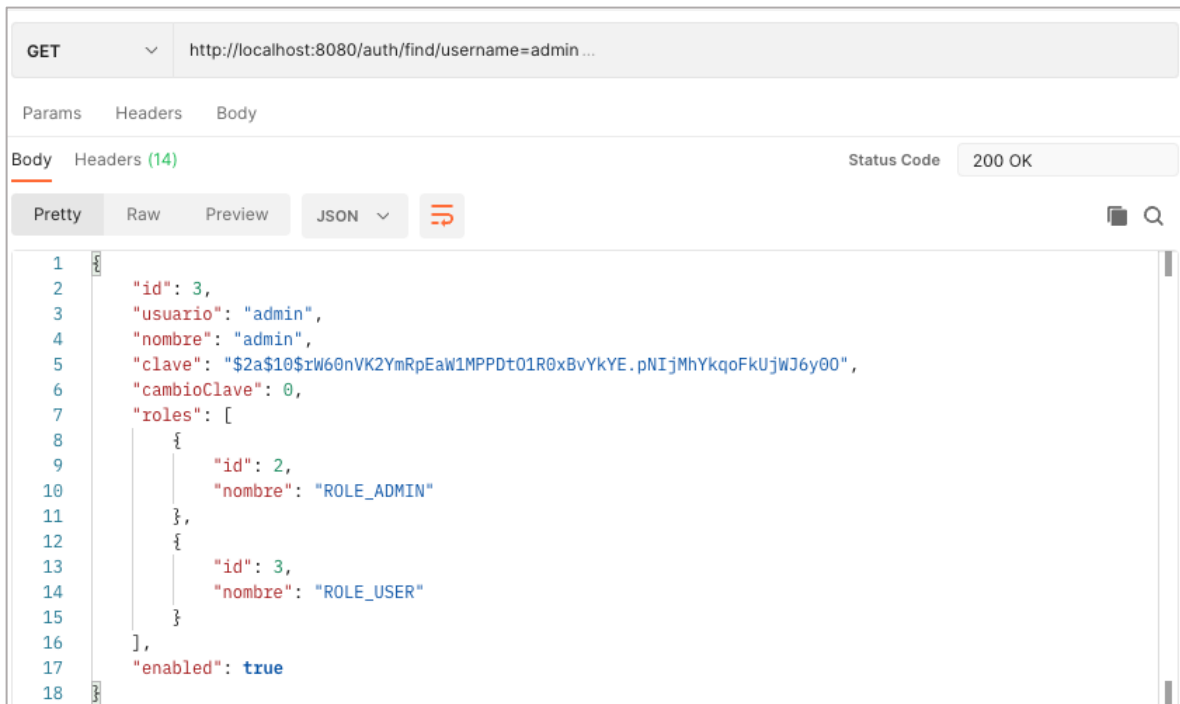
El servicio de restablecer clave se hace desde un perfil de *superadmin*, y en el URL de la petición se especifica el usuario y la clave que se le asignará. Como desde el servicio de cambio de clave se modificó la clave del usuario “admin”, podemos restablecerla a la clave original “admin”. La petición devuelve un mensaje cuando se ha completado.



## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

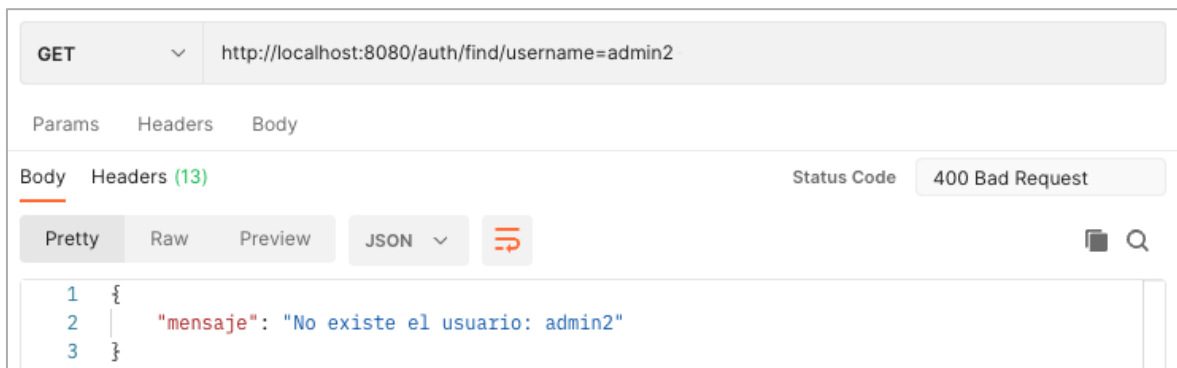
### 1.5. Buscar usuario por username

El servicio de búsqueda de usuario necesita el username del usuario para poder realizar la petición. Para este ejemplo, utilizaremos de nuevo el usuario “admin”. La petición nos devuelve el objeto Usuario y podemos observar la información más relevante.



```
GET http://localhost:8080/auth/find/username=admin...
Status Code: 200 OK
Body: Headers (14)
Pretty Raw Preview JSON
1 {
2   "id": 3,
3   "usuario": "admin",
4   "nombre": "admin",
5   "clave": "$2a$10$rw60nVK2YmRpEaw1MPPDt01R0xBvYkYE.pNIjMhYkqoFkUjWJ6y00",
6   "cambioClave": 0,
7   "roles": [
8     {
9       "id": 2,
10      "nombre": "ROLE_ADMIN"
11    },
12    {
13      "id": 3,
14      "nombre": "ROLE_USER"
15    }
16  ],
17   "enabled": true
18 }
```

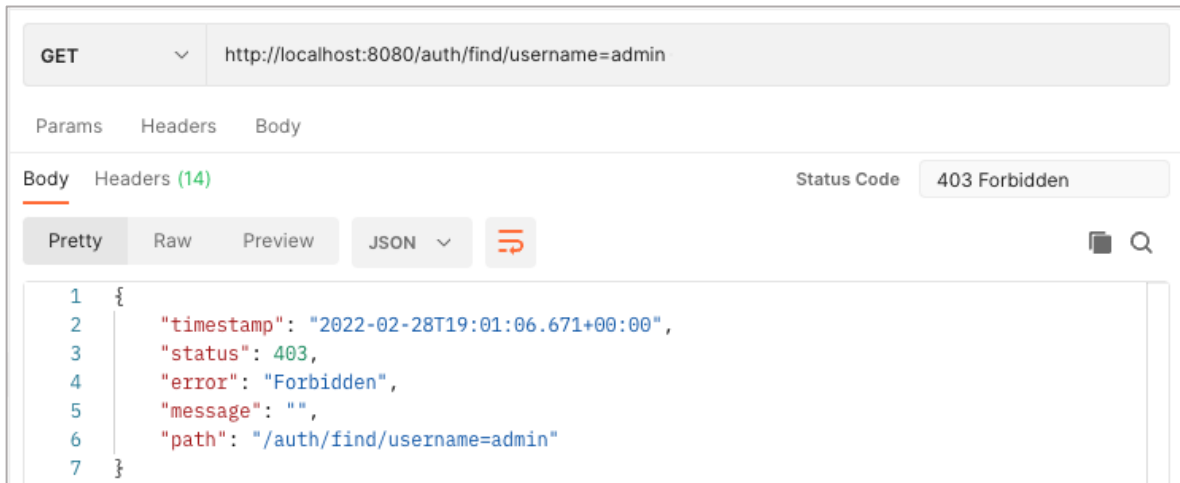
En caso de que el usuario no exista, se envía como respuesta una advertencia. Para el ejemplo, se intenta buscar un usuario “admin2”.



```
GET http://localhost:8080/auth/find/username=admin2
Status Code: 400 Bad Request
Body: Headers (13)
Pretty Raw Preview JSON
1 {
2   "mensaje": "No existe el usuario: admin2"
3 }
```

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

También es un servicio que solo puede ser llamado por un usuario *superadmin*, en caso contrario se envía una respuesta de no permitido.



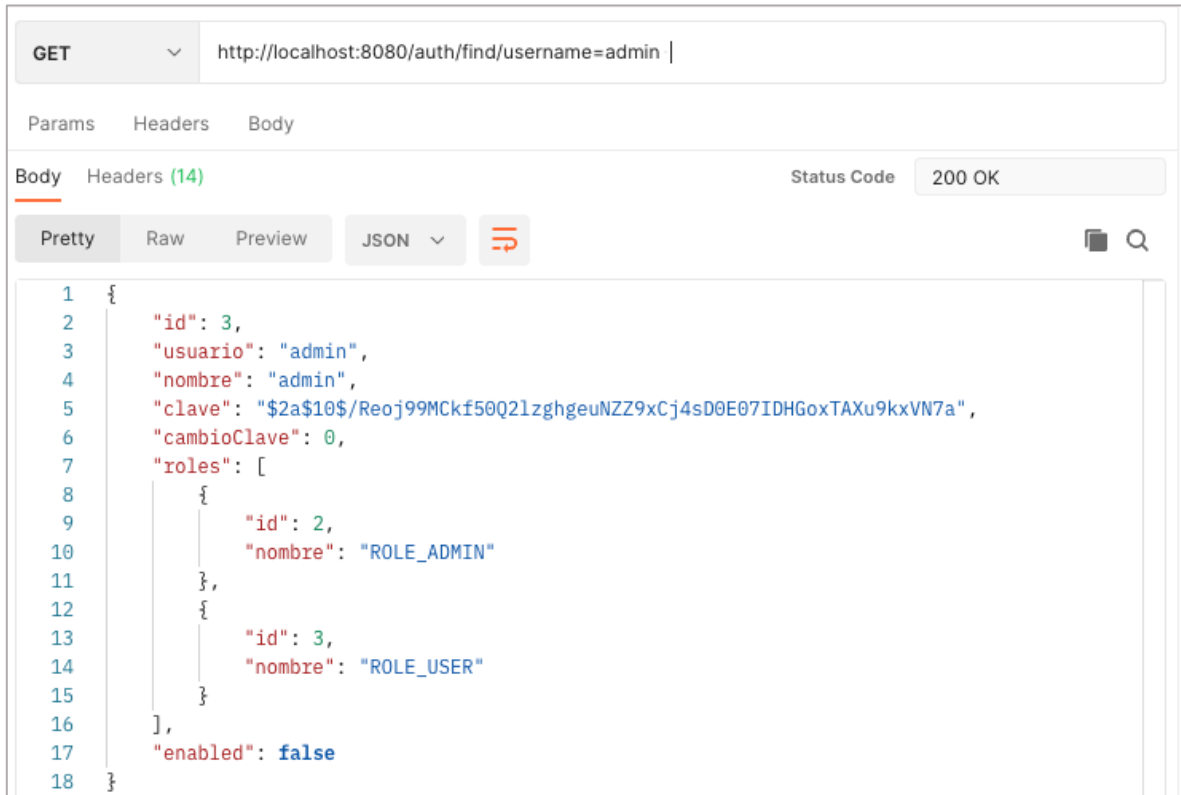
### 1.6. Deshabilitar usuario

El servicio de deshabilitar usuario necesita que se especifique el username en el URL de la petición. Además, debe realizarlo un perfil con rol de *superadmin*. Para el ejemplo, se utilizará de nuevo el usuario “admin”.



## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

Podemos comprobar que el usuario si fue deshabilitado haciendo uso del servicio de búsqueda de usuario, y verificando el valor “*false*” en el campo “*enabled*”, que significa deshabilitado.



```
GET http://localhost:8080/auth/find/username=admin |

Params Headers Body

Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

1 {
2   "id": 3,
3   "usuario": "admin",
4   "nombre": "admin",
5   "clave": "$2a$10$/Reoj99Mckf50Q2lzhgeuNZZ9xCj4sD0E07IDHGoxTAXu9kxVN7a",
6   "cambioClave": 0,
7   "roles": [
8     {
9       "id": 2,
10      "nombre": "ROLE_ADMIN"
11    },
12    {
13      "id": 3,
14      "nombre": "ROLE_USER"
15    }
16  ],
17  "enabled": false
18 }
```

En el caso de que el usuario no exista, la petición no termina con éxito, y se envía un mensaje de advertencia. En el ejemplo se intentará deshabilitar un usuario “admin2”.



```
PUT http://localhost:8080/auth/deshabilitar/admin2

Params Headers Body

Body Headers (13) Status Code 400 Bad Request

Pretty Raw Preview JSON

1 {
2   "mensaje": "El usuario admin2 no existe"
3 }
```



## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En caso de que el servicio sea llamado por un usuario con rol diferente a *superadmin* se recibe una respuesta de no permitido.



The screenshot shows a REST client interface with the following details:

- Method: PUT
- URL: http://localhost:8080/auth/deshabilitar/admin
- Body tab selected, showing Headers (14)
- Status Code: 403 Forbidden
- Response body (JSON):

```
1 {  
2   "timestamp": "2022-02-28T19:41:20.017+00:00",  
3   "status": 403,  
4   "error": "Forbidden",  
5   "message": "",  
6   "path": "/auth/deshabilitar/admin2"  
7 }
```

### 1.7. Habilitar usuario

El servicio de habilitar usuario necesita que se especifique el username en el URL de la petición. Además, debe realizarlo un perfil con rol de *superadmin*. Para el ejemplo, se utilizará de nuevo el usuario “admin”.



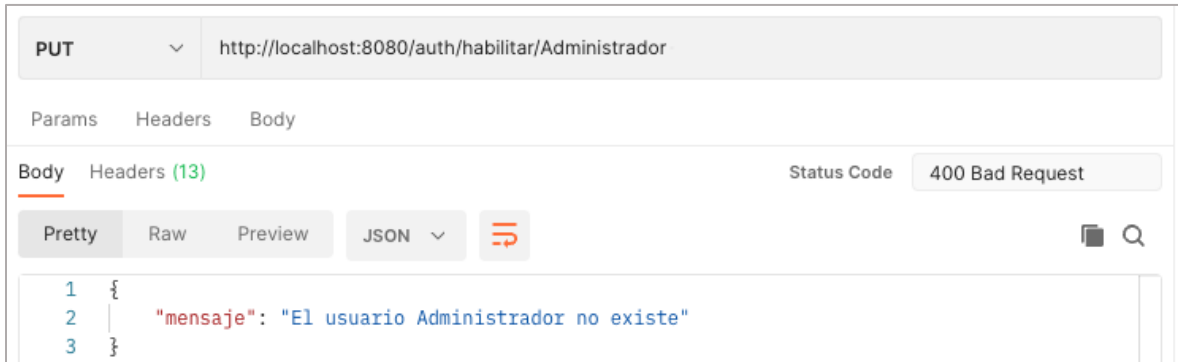
The screenshot shows a REST client interface with the following details:

- Method: PUT
- URL: http://localhost:8080/auth/habilitar/admin...
- Body tab selected, showing Headers (14)
- Status Code: 200 OK
- Response body (JSON):

```
1 {  
2   "mensaje": "Usuario admin habilitado"  
3 }
```

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En caso de que el usuario que se quiere habilitar, la petición falla y se envía un mensaje de advertencia.



El usuario que realice esta petición debe ser *superadmin*, en caso contrario se obtiene como respuesta una solicitud no permitida.

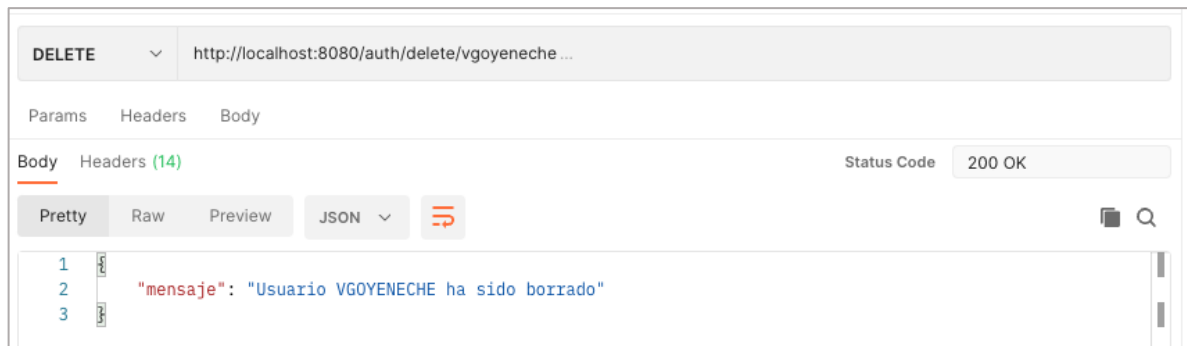


### 1.8. Eliminar usuario

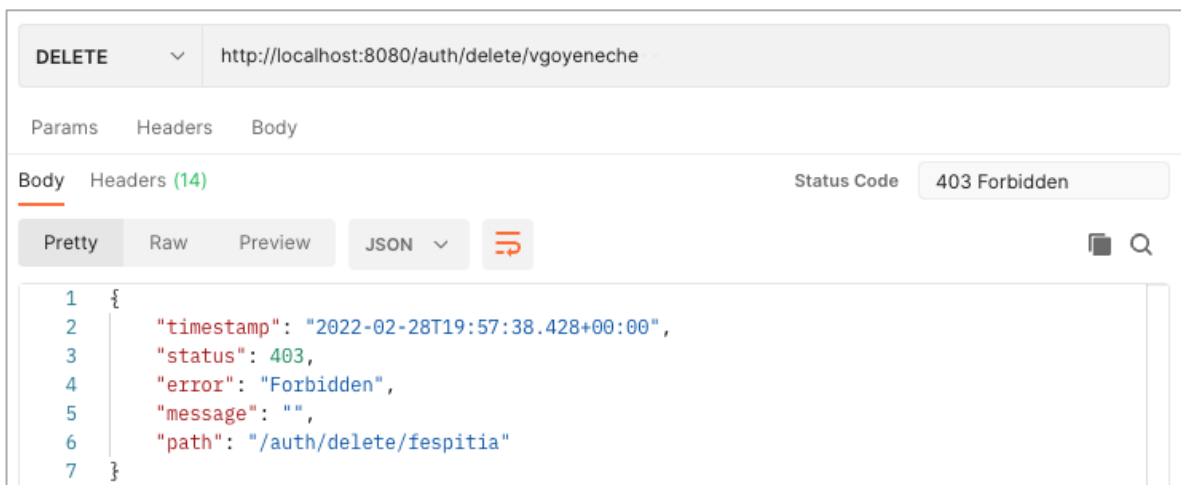
El servicio de eliminar usuario requiere del *username* del usuario en la URL de la petición. En el ejemplo, se eliminará el usuario “vgoyeneche”.

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

Si la solicitud termina con éxito, se envía un mensaje que indica que el usuario fue borrado satisfactoriamente.



Solo los usuarios con rol *superadmin* pueden eliminar los usuarios, en caso contrario se envía una respuesta de acción no permitida.



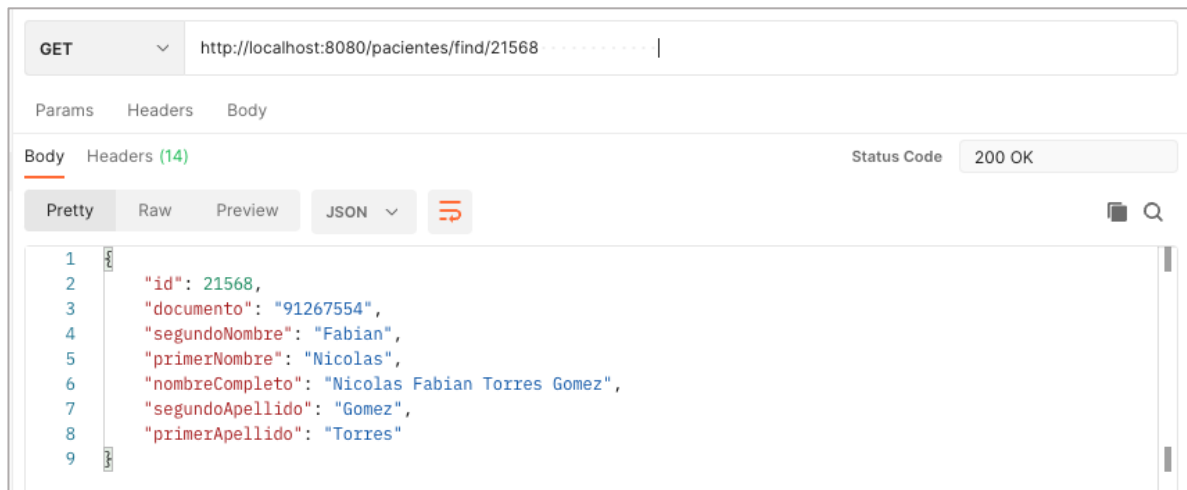
En el caso de que el usuario no exista, la petición falla y se envía un mensaje de advertencia.



## 2. Paciente

### 2.1. Buscar paciente por número de ingreso

El servicio de buscar paciente por número de ingreso acepta en el URL el número. En caso de que el número sea válido, la petición se completa de manera satisfactoria y se obtiene el objeto paciente con la información relevante. En el ejemplo, buscaremos un paciente identificado con el número “21568”.

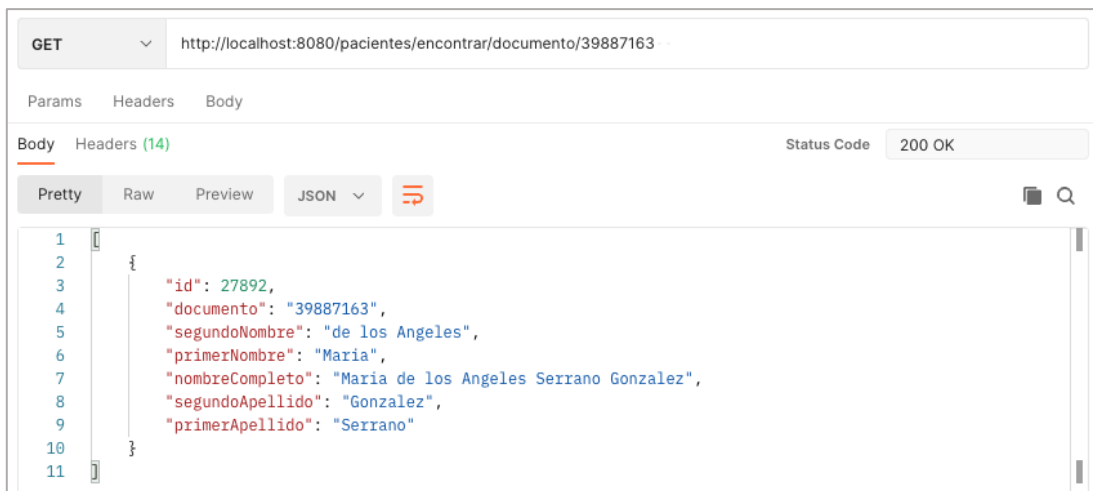


En caso de que no haya pacientes registrados con ese número, la petición falla y se envía un mensaje de advertencia.



### 2.2. *Buscar paciente por número de documento*

El servicio de buscar paciente por número de documento acepta en el URL el número de documento. En caso de que el número sea válido, la petición se completa de manera satisfactoria y se obtiene el objeto paciente con la información relevante. Para el ejemplo, buscaremos un paciente con número de documento “39887163”.



The screenshot shows a REST client interface with a GET request to `http://localhost:8080/pacientes/encontrar/documento/39887163`. The response status is 200 OK. The response body is a JSON object containing patient information.

```
1 {  
2   {  
3     "id": 27892,  
4     "documento": "39887163",  
5     "segundoNombre": "de los Angeles",  
6     "primerNombre": "Maria",  
7     "nombreCompleto": "Maria de los Angeles Serrano Gonzalez",  
8     "segundoApellido": "Gonzalez",  
9     "primerApellido": "Serrano"  
10  }  
11 }
```

En caso de que no se encuentren pacientes registrados cuyo documento coincida con el número ingresado se muestra un mensaje alusivo a la situación.

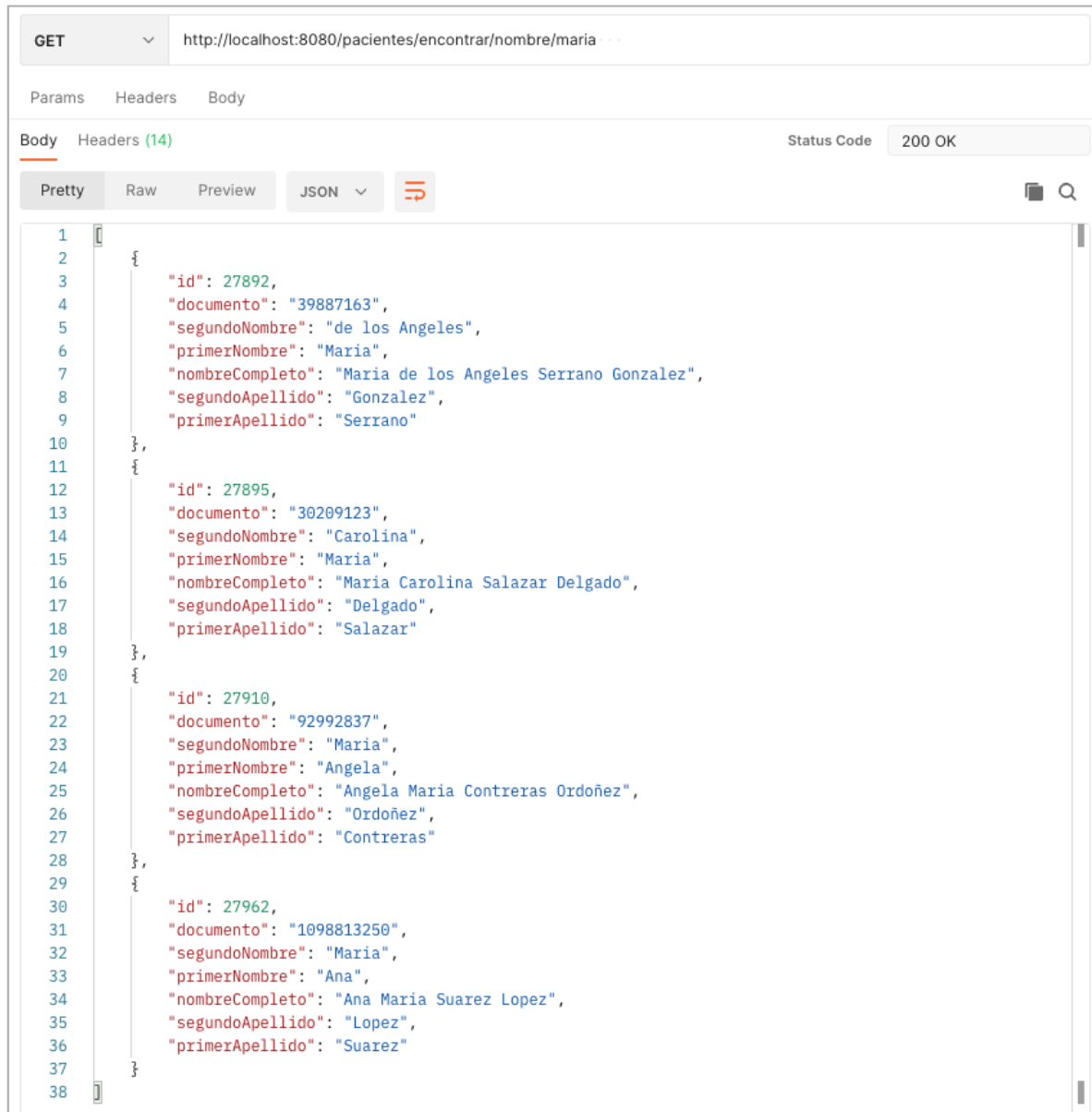


The screenshot shows a REST client interface with a GET request to `http://localhost:8080/pacientes/encontrar/documento/39887162`. The response status is 400 Bad Request. The response body is a JSON object containing an error message.

```
1 {  
2   "mensaje": "No se ha(n) encontrado paciente(s) identificado(s) con número de  
3             documento: 39887162"  
4 }
```

### 2.3. Buscar paciente por nombre

El servicio de buscar paciente por nombre acepta en el URL el nombre. La petición se completa de manera satisfactoria y se obtiene el objeto paciente con la información relevante. Para el ejemplo, buscaremos los pacientes cuyo nombre contengan con “maria”.



```
GET http://localhost:8080/pacientes/encontrar/nombre/maria

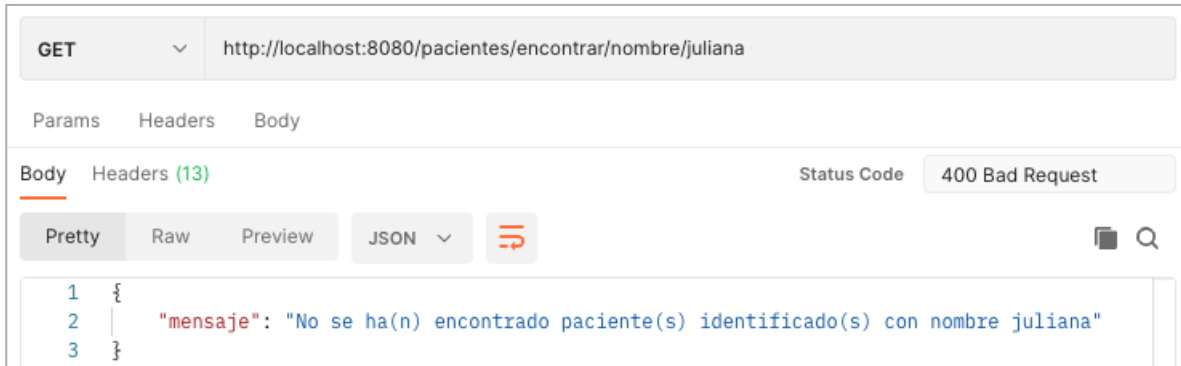
Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

1 {
2   {
3     "id": 27892,
4     "documento": "39887163",
5     "segundoNombre": "de los Angeles",
6     "primerNombre": "Maria",
7     "nombreCompleto": "Maria de los Angeles Serrano Gonzalez",
8     "segundoApellido": "Gonzalez",
9     "primerApellido": "Serrano"
10  },
11  {
12    "id": 27895,
13    "documento": "30209123",
14    "segundoNombre": "Carolina",
15    "primerNombre": "Maria",
16    "nombreCompleto": "Maria Carolina Salazar Delgado",
17    "segundoApellido": "Delgado",
18    "primerApellido": "Salazar"
19  },
20  {
21    "id": 27910,
22    "documento": "92992837",
23    "segundoNombre": "Maria",
24    "primerNombre": "Angela",
25    "nombreCompleto": "Angela Maria Contreras Ordoñez",
26    "segundoApellido": "Ordoñez",
27    "primerApellido": "Contreras"
28  },
29  {
30    "id": 27962,
31    "documento": "1098813250",
32    "segundoNombre": "Maria",
33    "primerNombre": "Ana",
34    "nombreCompleto": "Ana Maria Suarez Lopez",
35    "segundoApellido": "Lopez",
36    "primerApellido": "Suarez"
37  }
38 }
```

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

En caso de que no se encuentren pacientes registrados cuyo nombre coincida con el ingresado se muestra un mensaje alusivo a la situación.



### 3. Cama

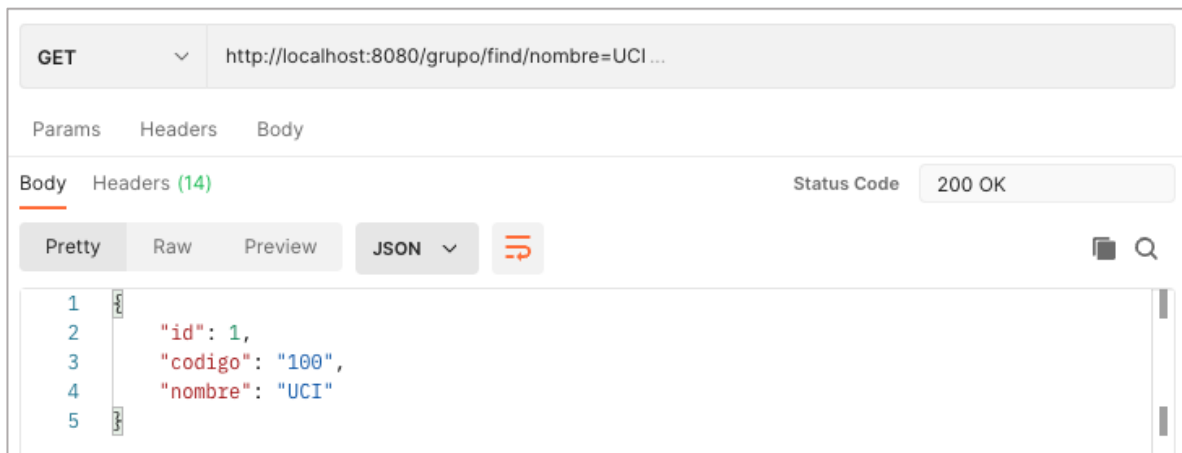
#### 3.1. Obtener todos los grupos

El servicio para obtener todos los grupos no necesita datos adicionales para poder ser llamado. Podemos observar todos los grupos registrados en el hospital.



### 3.2. *Buscar grupo por nombre*

Para el servicio de búsqueda de grupo por nombre, se ingresa en la URL el nombre del grupo que se desea buscar. Debido a que este servicio solo es llamado a partir de la lista obtenida por el servicio de búsqueda de todos los grupos, no se necesita validar si el nombre existe. En este ejemplo, se busca el grupo “UCI”, y se obtiene el objeto Grupo con la información relevante.

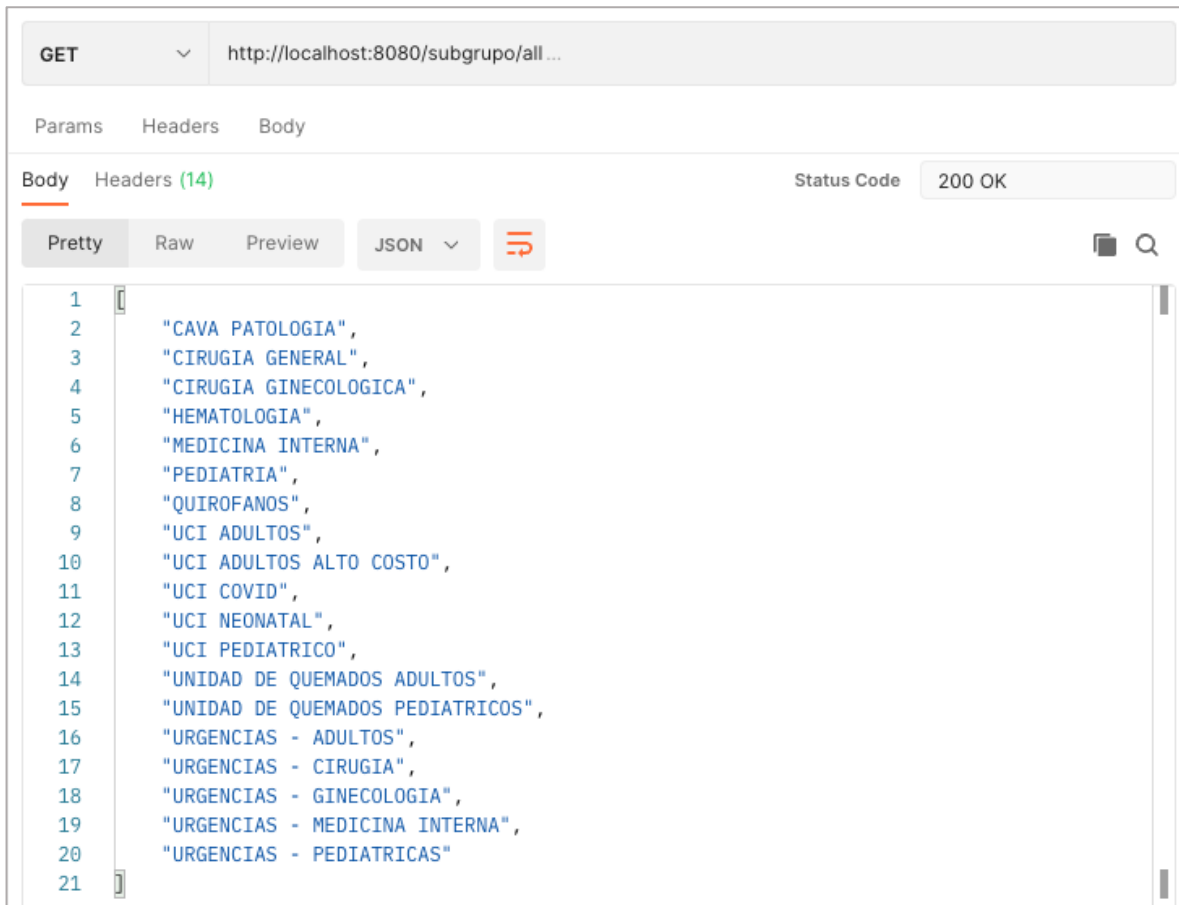




## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

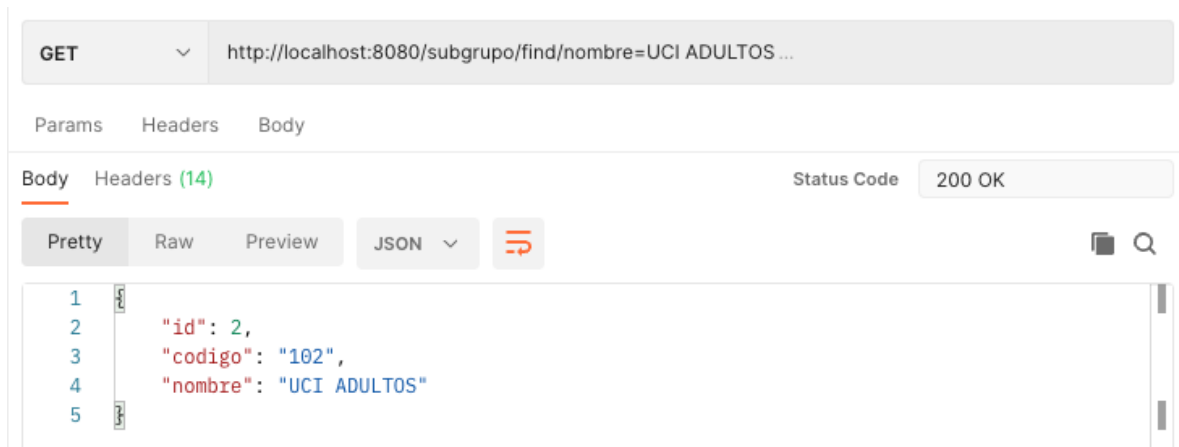
### 3.3. *Obtener todos los subgrupos*

El servicio para obtener todos los subgrupos no necesita datos adicionales para poder ser llamado. Podemos observar todos los subgrupos registrados en el hospital.



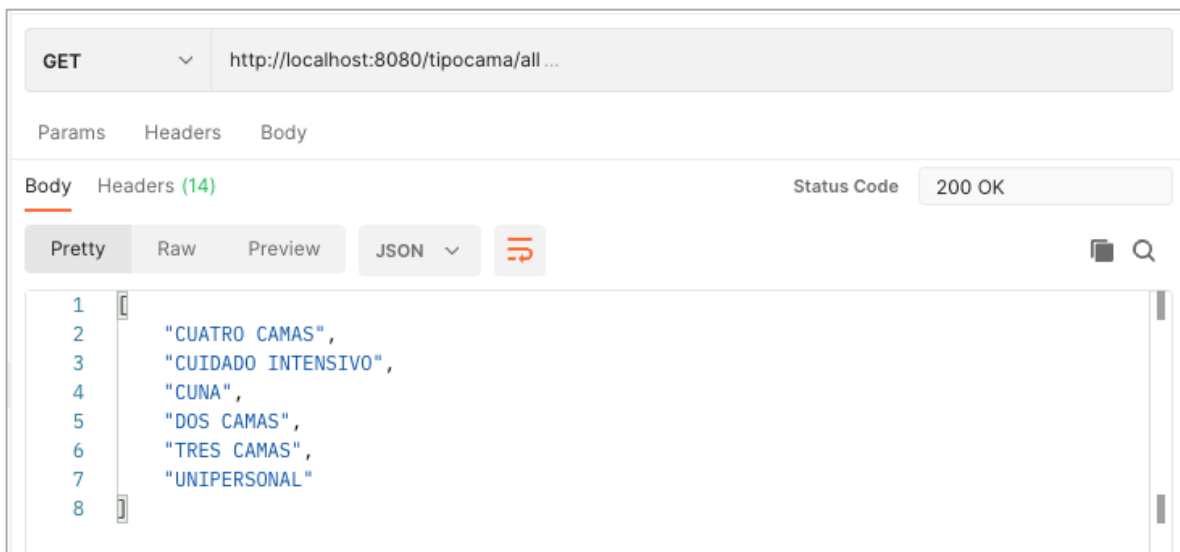
### 3.4. *Buscar subgrupo por nombre*

Para el servicio de búsqueda de subgrupo por nombre, se ingresa en la URL el nombre del subgrupo que se desea buscar. Debido a que este servicio solo es llamado a partir de la lista obtenida por el servicio de búsqueda de todos los subgrupos, no se necesita validar si el nombre existe. En este ejemplo, se busca el subgrupo “UCI ADULTOS”, y se obtiene el objeto Subgrupo con la información relevante.



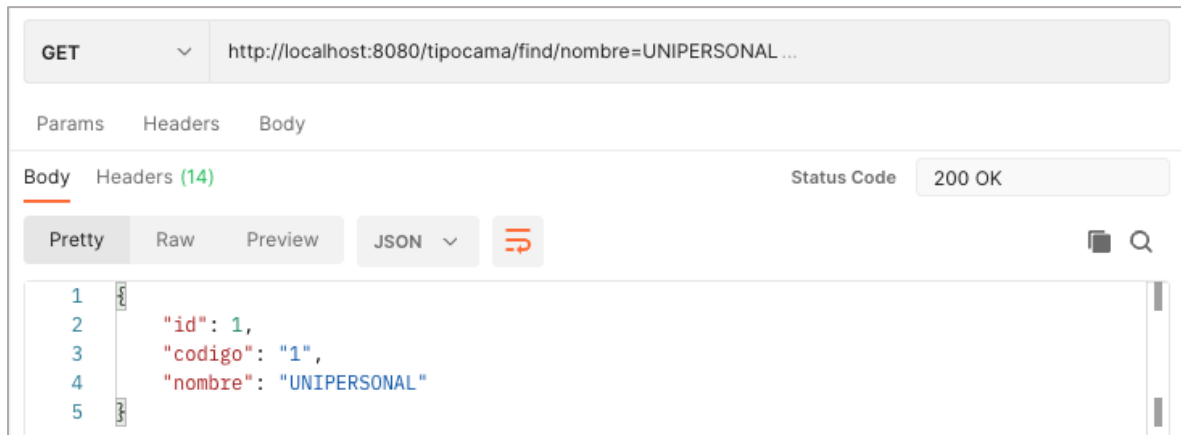
### 3.5. *Obtener todos los tipos de cama*

El servicio para obtener todos los tipos de cama no necesita datos adicionales para poder ser llamado. Podemos observar todos los tipos de cama registrados en el hospital.



### 3.6. *Buscar tipo de cama por nombre*

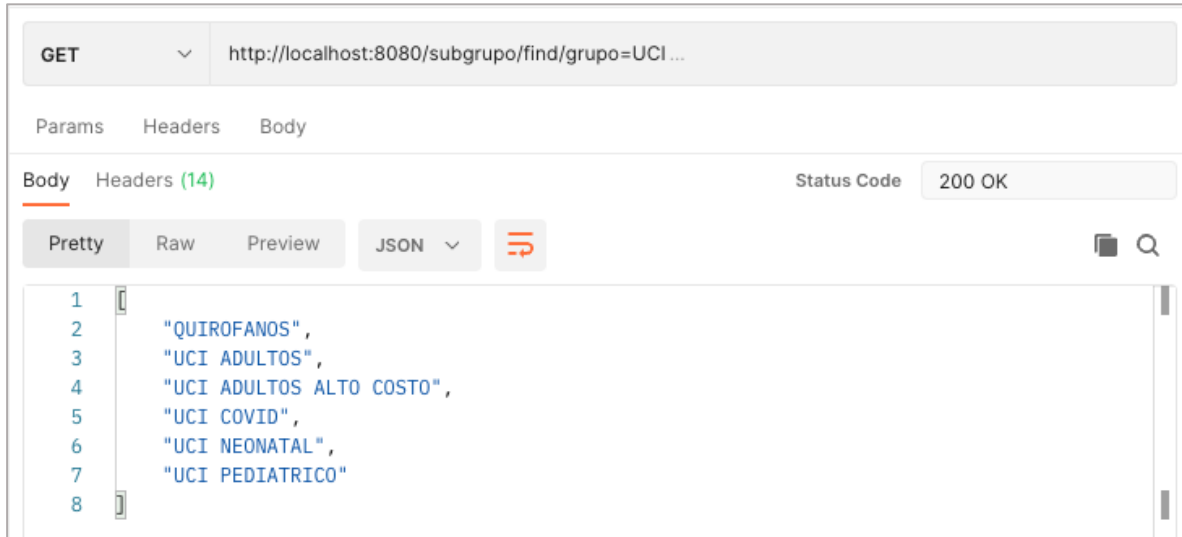
Para el servicio de búsqueda de tipo de cama por nombre, se ingresa en la URL el nombre del tipo de cama que se desea buscar. Debido a que este servicio solo es llamado a partir de la lista obtenida por el servicio de búsqueda de todos los tipos de cama, no se necesita validar si el nombre existe. En este ejemplo, se busca el tipo de cama “UNIPERSONAL”, y se obtiene el objeto Tipocama con la información relevante.



### 3.7. *Obtener subgrupos por grupo*

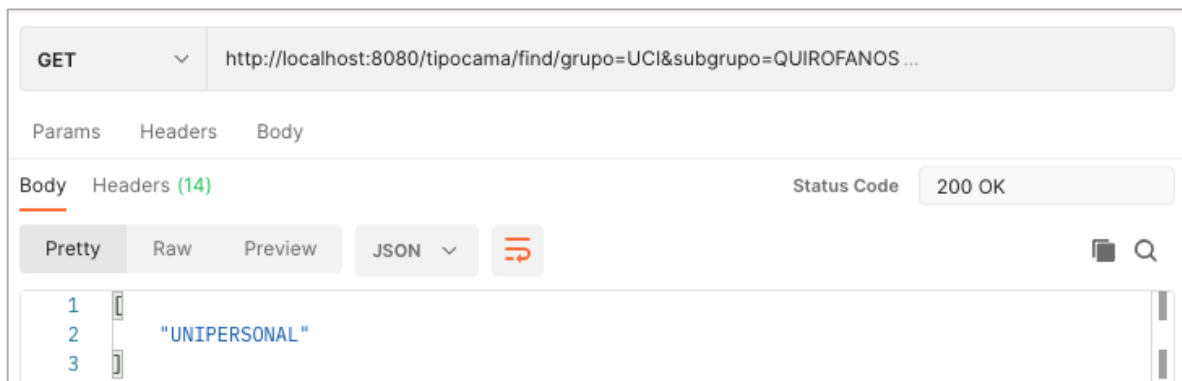
El servicio de búsqueda de subgrupos por grupo permite buscar los subgrupos relacionados a un grupo por medio del nombre del grupo. Este servicio requiere que se ingrese el nombre en el URL de la petición. De igual forma, estos nombres se obtienen a partir del resultado de búsqueda de todos los grupos, por lo que no es necesario validar que existe un grupo con el nombre ingresado. Para el ejemplo, utilizaremos el grupo “UCI” como filtro de búsqueda.

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS



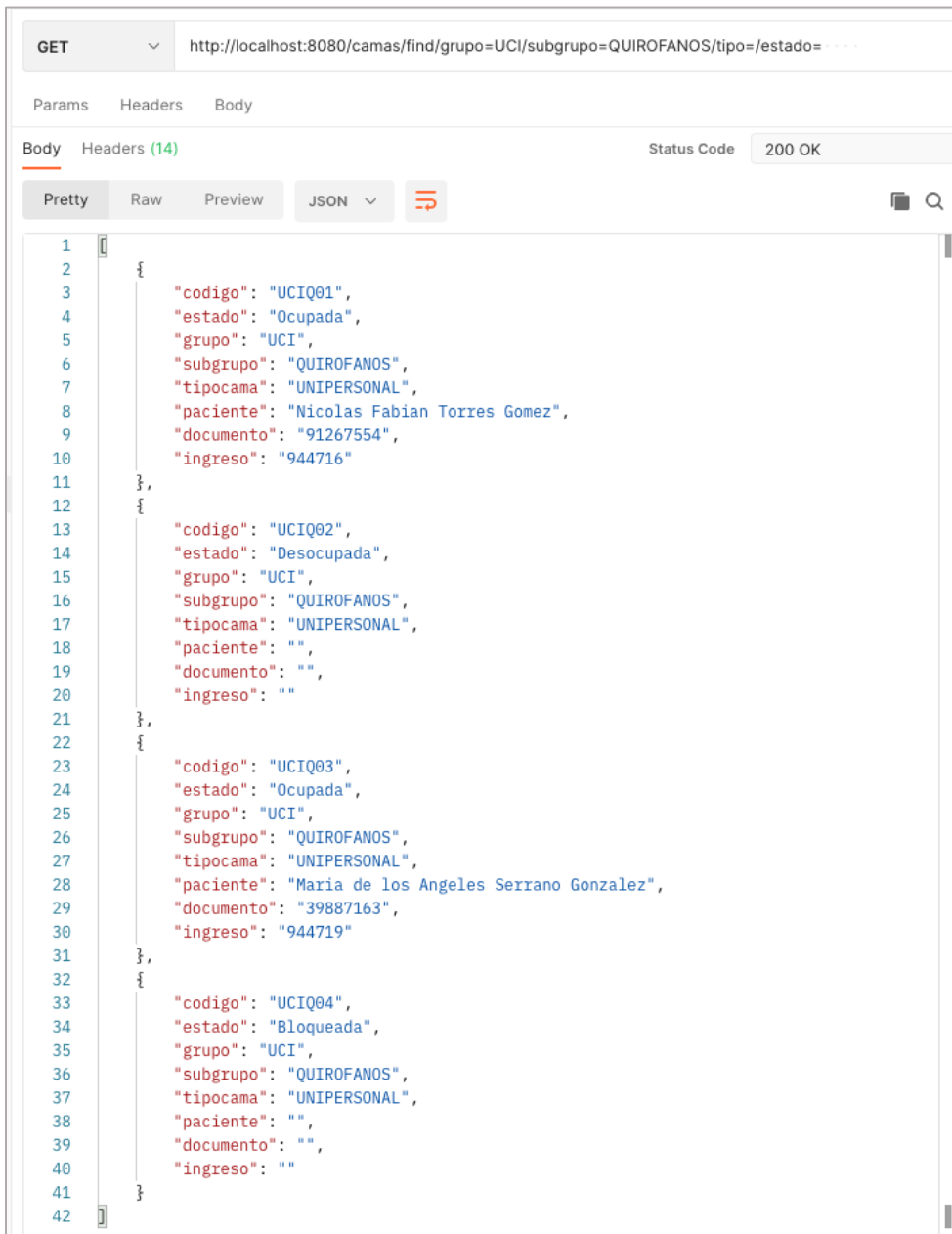
### 3.8. Obtener tipos de cama por grupo y subgrupo

El servicio de búsqueda de tipo de cama por grupo y subgrupo permite buscar los tipos de cama relacionados a un grupo y subgrupo por medio del nombre del grupo y el subgrupo. Este servicio requiere que se ingresen los nombres en el URL de la petición. De igual forma, estos nombres se obtienen a partir del resultado de búsqueda de todos los grupos y de todos los subgrupos, por lo que no es necesario validar que existen el grupo y el subgrupo ingresado. Para el ejemplo, utilizaremos el grupo “UCI” y el subgrupo “QUIROFANOS” como filtros de búsqueda.



### 3.9. Buscar una cama por filtros

Para el servicio de búsqueda de camas con filtro: grupo, subgrupo, tipo de cama y estado, se deben ingresar obligatoriamente un grupo y un subgrupo por su nombre, mientras que el tipo de cama y el estado son argumentos opcionales como podemos observar en el ejemplo. En este caso, buscaremos por grupo “UCI” y subgrupo “QUIROFANOS”.



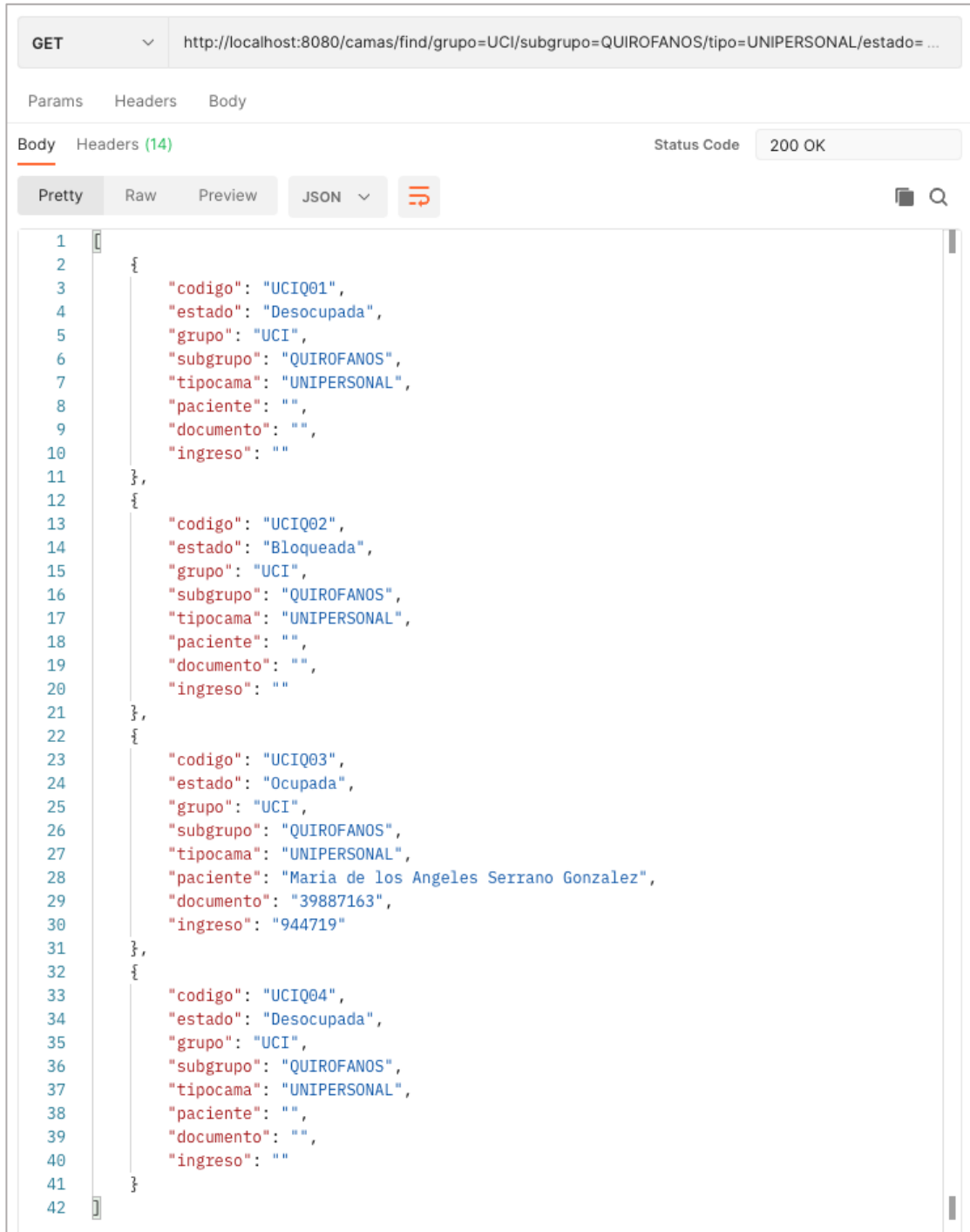
```
GET http://localhost:8080/camas/find/grupo=UCI/subgrupo=QUIROFANOS/tipo=/estado=...

Params Headers Body
Body Headers (14) Status Code 200 OK
Pretty Raw Preview JSON

1 {
2   {
3     "codigo": "UCIQ01",
4     "estado": "Ocupada",
5     "grupo": "UCI",
6     "subgrupo": "QUIROFANOS",
7     "tipocama": "UNIPERSONAL",
8     "paciente": "Nicolas Fabian Torres Gomez",
9     "documento": "91267554",
10    "ingreso": "944716"
11  },
12  {
13    "codigo": "UCIQ02",
14    "estado": "Desocupada",
15    "grupo": "UCI",
16    "subgrupo": "QUIROFANOS",
17    "tipocama": "UNIPERSONAL",
18    "paciente": "",
19    "documento": "",
20    "ingreso": ""
21  },
22  {
23    "codigo": "UCIQ03",
24    "estado": "Ocupada",
25    "grupo": "UCI",
26    "subgrupo": "QUIROFANOS",
27    "tipocama": "UNIPERSONAL",
28    "paciente": "Maria de los Angeles Serrano Gonzalez",
29    "documento": "39887163",
30    "ingreso": "944719"
31  },
32  {
33    "codigo": "UCIQ04",
34    "estado": "Bloqueada",
35    "grupo": "UCI",
36    "subgrupo": "QUIROFANOS",
37    "tipocama": "UNIPERSONAL",
38    "paciente": "",
39    "documento": "",
40    "ingreso": ""
41  }
42 }
```

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

Ahora, buscaremos por grupo “UCI”, subgrupo “QUIROFANOS”, y tipo de cama “UNIPERSONAL”.

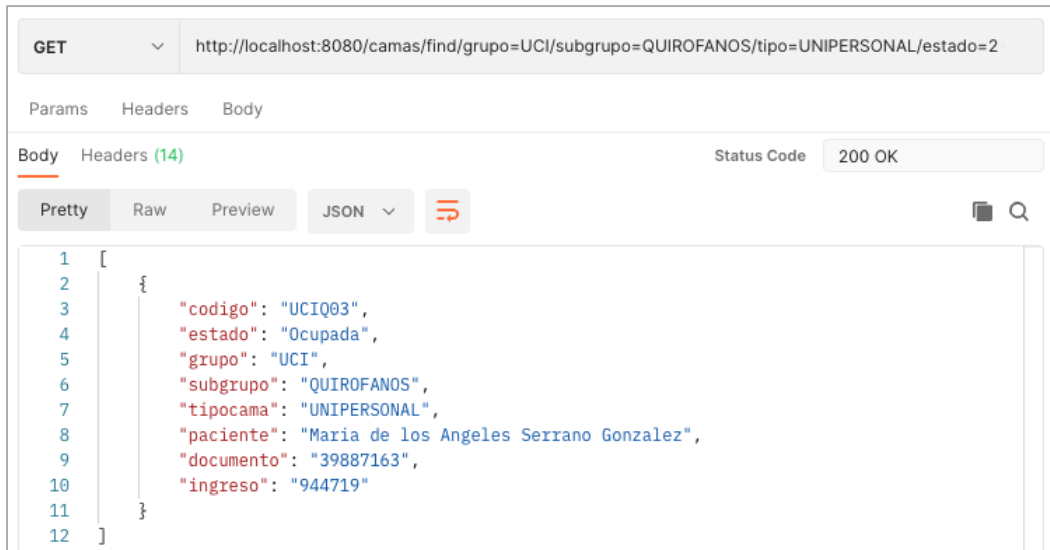


The screenshot shows a web browser window with a REST client interface. The URL bar displays a GET request to `http://localhost:8080/camas/find/grupo=UCI/subgrupo=QUIROFANOS/tipo=UNIPERSONAL/estado=...`. The interface includes tabs for Params, Headers, and Body. The Body tab is active, showing a JSON response with a status code of 200 OK. The JSON response is formatted in 'Pretty' mode and contains an array of four objects, each representing a bed record with fields like 'codigo', 'estado', 'grupo', 'subgrupo', 'tipocama', 'paciente', 'documento', and 'ingreso'.

```
1  {
2    {
3      "codigo": "UCIQ01",
4      "estado": "Desocupada",
5      "grupo": "UCI",
6      "subgrupo": "QUIROFANOS",
7      "tipocama": "UNIPERSONAL",
8      "paciente": "",
9      "documento": "",
10     "ingreso": ""
11   },
12   {
13     "codigo": "UCIQ02",
14     "estado": "Bloqueada",
15     "grupo": "UCI",
16     "subgrupo": "QUIROFANOS",
17     "tipocama": "UNIPERSONAL",
18     "paciente": "",
19     "documento": "",
20     "ingreso": ""
21   },
22   {
23     "codigo": "UCIQ03",
24     "estado": "Ocupada",
25     "grupo": "UCI",
26     "subgrupo": "QUIROFANOS",
27     "tipocama": "UNIPERSONAL",
28     "paciente": "Maria de los Angeles Serrano Gonzalez",
29     "documento": "39887163",
30     "ingreso": "944719"
31   },
32   {
33     "codigo": "UCIQ04",
34     "estado": "Desocupada",
35     "grupo": "UCI",
36     "subgrupo": "QUIROFANOS",
37     "tipocama": "UNIPERSONAL",
38     "paciente": "",
39     "documento": "",
40     "ingreso": ""
41   }
42 }
```

## COMPONENTE BACKEND DE SOFTWARE DE CENSO DE CAMAS

Ahora, buscaremos por grupo “UCI”, subgrupo “QUIROFANOS”, tipo de cama “UNIPERSONAL”, y estado “1”.



```
GET http://localhost:8080/camas/find/grupo=UCI/subgrupo=QUIROFANOS/tipo=UNIPERSONAL/estado=2

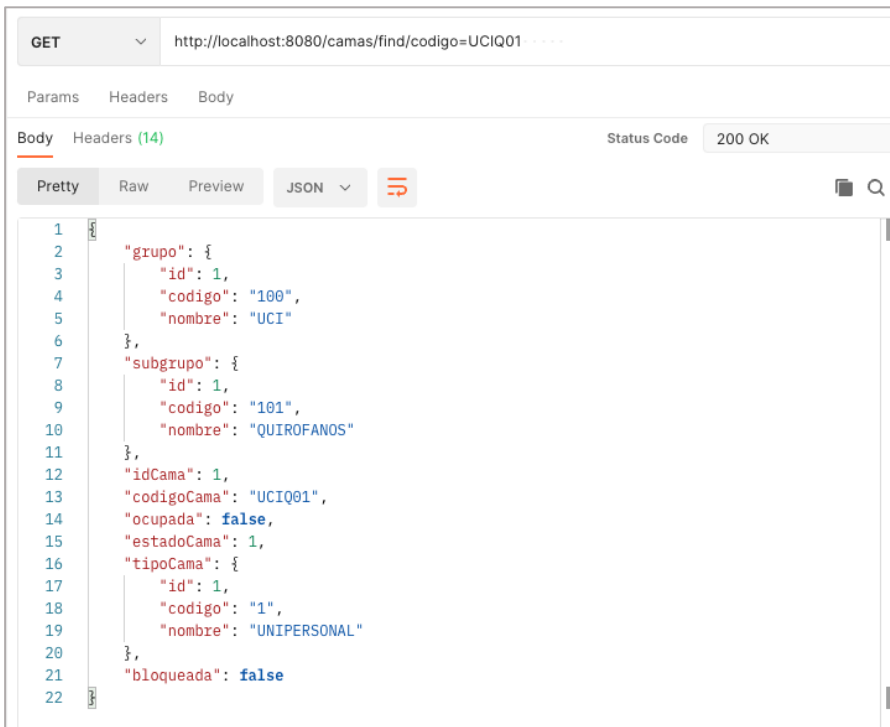
Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

1 [
2   {
3     "codigo": "UCIQ03",
4     "estado": "Ocupada",
5     "grupo": "UCI",
6     "subgrupo": "QUIROFANOS",
7     "tipocama": "UNIPERSONAL",
8     "paciente": "Maria de los Angeles Serrano Gonzalez",
9     "documento": "39887163",
10    "ingreso": "944719"
11  }
12 ]
```

### 3.10. *Buscar cama por código*

Para el servicio de búsqueda de cama por código se ingresa el código de la cama en el URL de la petición. Para el ejemplo, buscaremos la cama identificada como “UCIQ01”.



```
GET http://localhost:8080/camas/find/codigo=UCIQ01

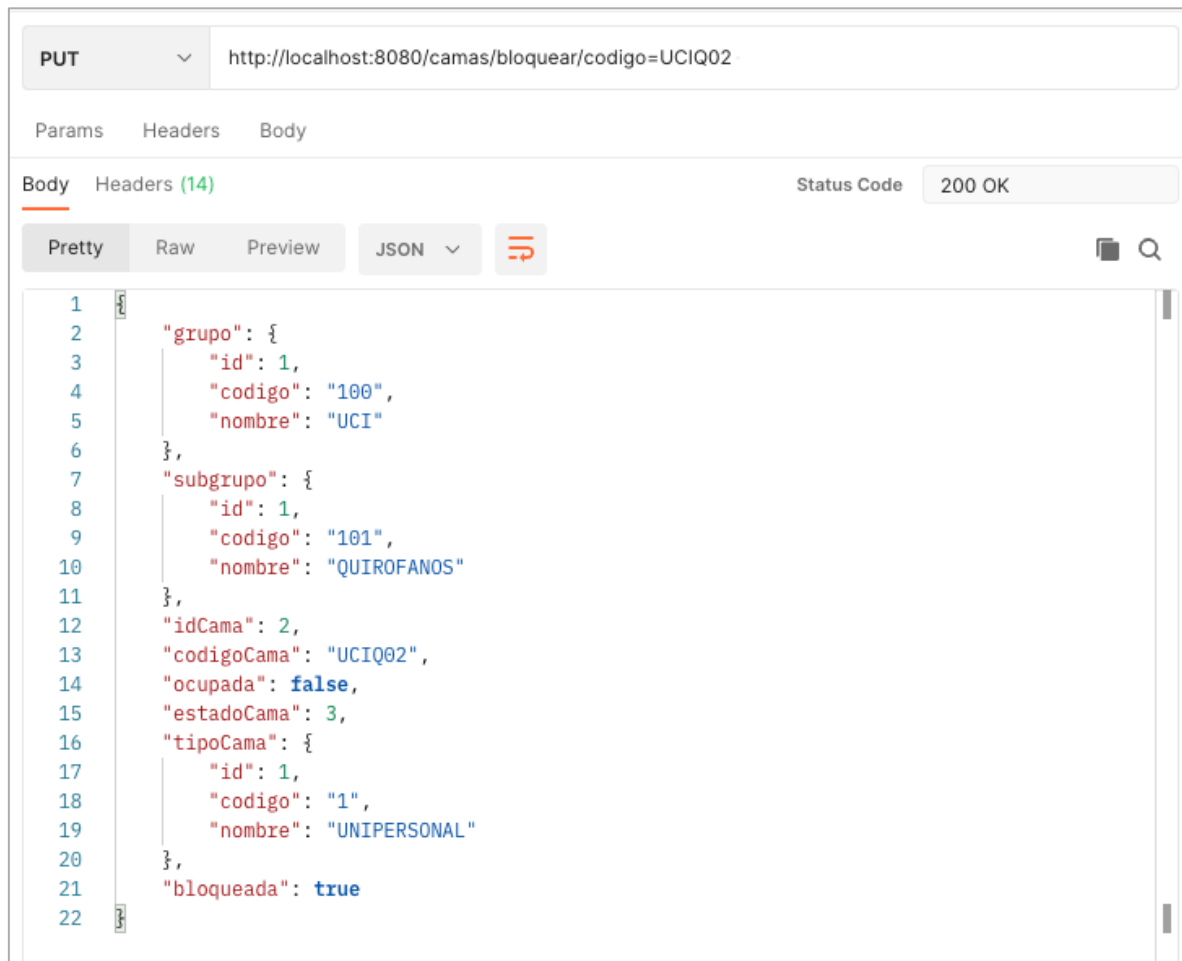
Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

1 {
2   "grupo": {
3     "id": 1,
4     "codigo": "100",
5     "nombre": "UCI"
6   },
7   "subgrupo": {
8     "id": 1,
9     "codigo": "101",
10    "nombre": "QUIROFANOS"
11  },
12  "idCama": 1,
13  "codigoCama": "UCIQ01",
14  "ocupada": false,
15  "estadoCama": 1,
16  "tipoCama": {
17    "id": 1,
18    "codigo": "1",
19    "nombre": "UNIPERSONAL"
20  },
21  "bloqueada": false
22 }
```

### 3.11. Bloquear cama

Para el servicio de bloquear cama, se ingresa el código de la cama que se desea bloquear, y se obtiene el objeto cama, donde el campo “*bloqueada*” ha cambiado a “*true*”.



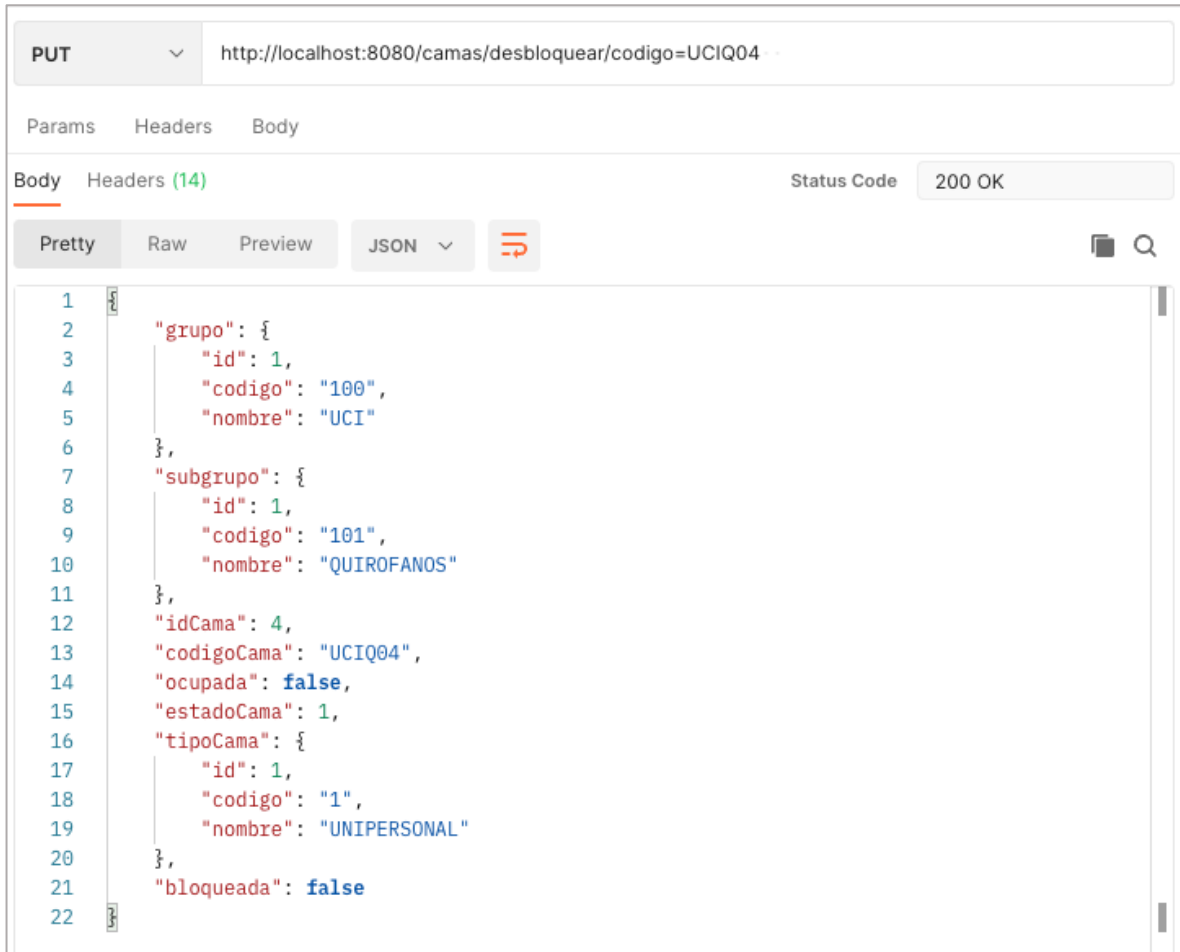
The screenshot shows a REST client interface with a PUT request to the URL `http://localhost:8080/camas/bloquear/codigo=UCIQ02`. The response status is 200 OK. The response body is a JSON object representing a bed, with the `bloqueada` field set to `true`.

```
1 {
2   "grupo": {
3     "id": 1,
4     "codigo": "100",
5     "nombre": "UCI"
6   },
7   "subgrupo": {
8     "id": 1,
9     "codigo": "101",
10    "nombre": "QUIROFANOS"
11  },
12  "idCama": 2,
13  "codigoCama": "UCIQ02",
14  "ocupada": false,
15  "estadoCama": 3,
16  "tipoCama": {
17    "id": 1,
18    "codigo": "1",
19    "nombre": "UNIPERSONAL"
20  },
21  "bloqueada": true
22 }
```



### 3.12. Desbloquear cama

Para el servicio de bloquear cama, se ingresa el código de la cama que se desea desbloquear, y se obtiene el objeto cama, donde el campo “*bloqueada*” ha cambiado a “*false*”.



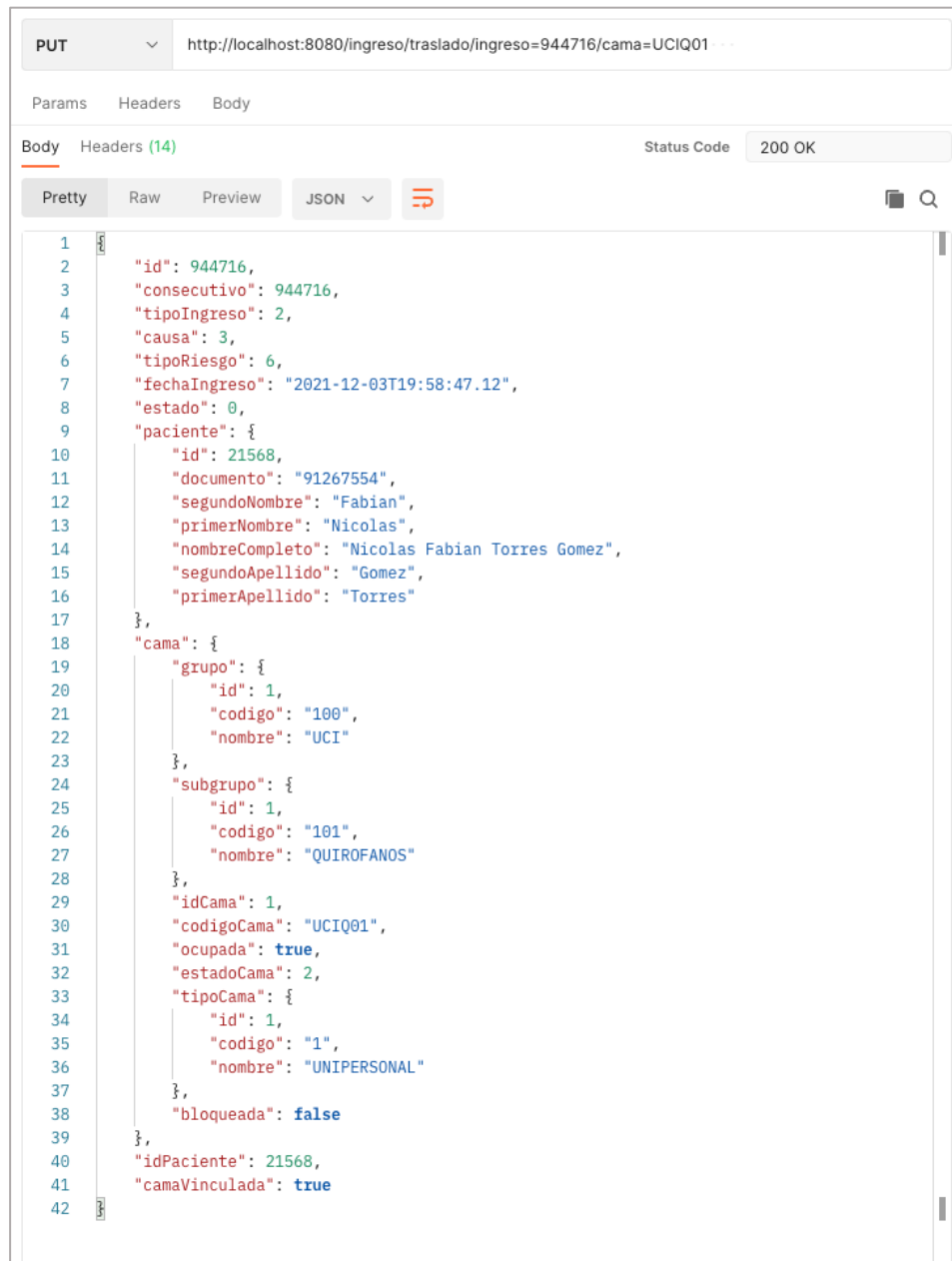
The screenshot shows a REST client interface with a PUT request to the URL `http://localhost:8080/camas/desbloquear/codigo=UCIQ04`. The response status is 200 OK. The response body is a JSON object with the following structure:

```
1  {
2    "grupo": {
3      "id": 1,
4      "codigo": "100",
5      "nombre": "UCI"
6    },
7    "subgrupo": {
8      "id": 1,
9      "codigo": "101",
10     "nombre": "QUIROFANOS"
11   },
12   "idCama": 4,
13   "codigoCama": "UCIQ04",
14   "ocupada": false,
15   "estadoCama": 1,
16   "tipoCama": {
17     "id": 1,
18     "codigo": "1",
19     "nombre": "UNIPERSONAL"
20   },
21   "bloqueada": false
22 }
```

## 4. Ingreso

### 4.1. Realizar traslado

Para el servicio de realizar traslado, se debe especificar el ingreso vinculado a un paciente que se quiere trasladar de cama, y la cama a la que se quiere trasladar. En este caso, tanto el paciente como la cama han sido validados por servicios anteriores. La petición retorna la información del paciente, y la información de la cama a la que ha sido trasladado.



```
PUT http://localhost:8080/ingreso/traslado/ingreso=944716/cama=UCIQ01

Params Headers Body
Body Headers (14) Status Code 200 OK
Pretty Raw Preview JSON ↻

1  {
2    "id": 944716,
3    "consecutivo": 944716,
4    "tipoIngreso": 2,
5    "causa": 3,
6    "tipoRiesgo": 6,
7    "fechaIngreso": "2021-12-03T19:58:47.12",
8    "estado": 0,
9    "paciente": {
10     "id": 21568,
11     "documento": "91267554",
12     "segundoNombre": "Fabian",
13     "primerNombre": "Nicolas",
14     "nombreCompleto": "Nicolas Fabian Torres Gomez",
15     "segundoApellido": "Gomez",
16     "primerApellido": "Torres"
17   },
18   "cama": {
19     "grupo": {
20       "id": 1,
21       "codigo": "100",
22       "nombre": "UCI"
23     },
24     "subgrupo": {
25       "id": 1,
26       "codigo": "101",
27       "nombre": "QUIROFANOS"
28     },
29     "idCama": 1,
30     "codigoCama": "UCIQ01",
31     "ocupada": true,
32     "estadoCama": 2,
33     "tipoCama": {
34       "id": 1,
35       "codigo": "1",
36       "nombre": "UNIPERSONAL"
37     },
38     "bloqueada": false
39   },
40   "idPaciente": 21568,
41   "camaVinculada": true
42 }
```

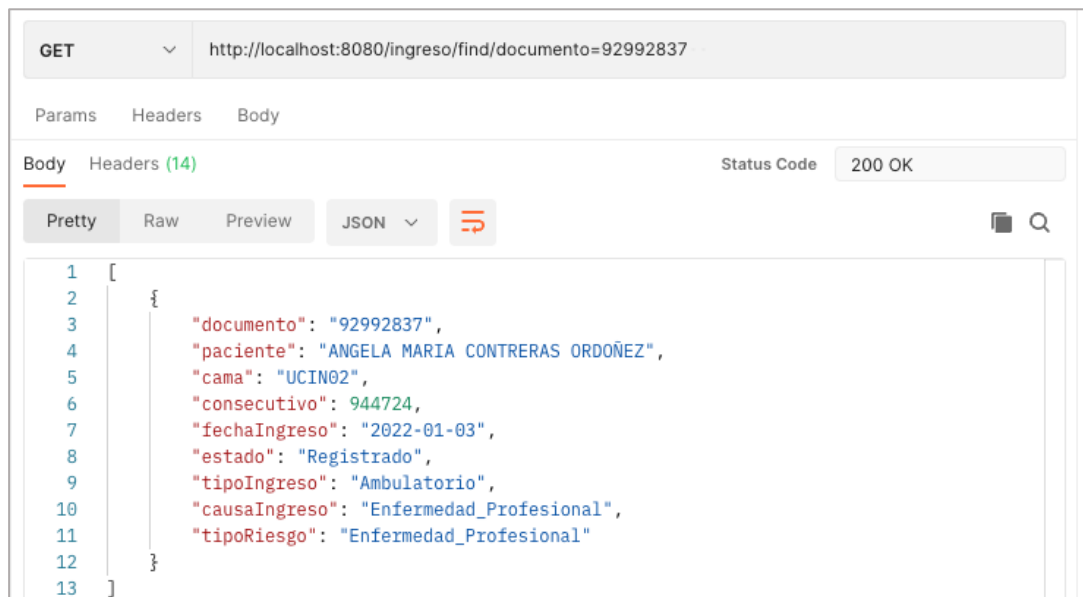
#### 4.2. Liberar una cama

Para el servicio de liberar una cama, es necesario el número del ingreso, que tiene el paciente que está ocupando dicha cama. De nuevo, este servicio solo trabaja con datos obtenidos de los resultados de servicios anteriores, por lo que se garantiza que son válidos. Se obtiene como resultado un mensaje que indica que la cama fue liberada satisfactoriamente.



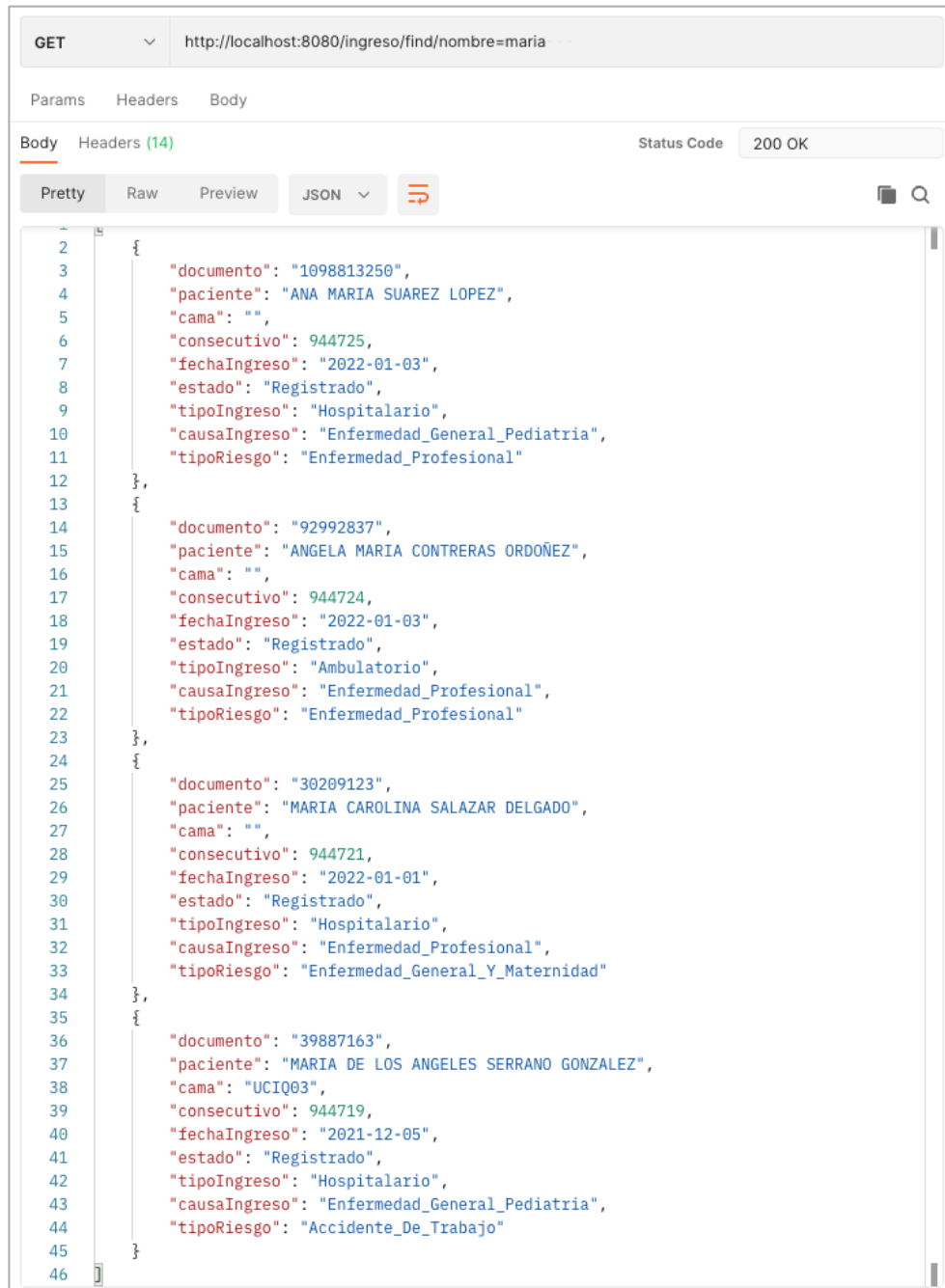
#### 4.3. Buscar ingreso por número de documento del paciente

Para el servicio de búsqueda de ingreso por número de documento, se ingresa el número de documento con el que se desea filtrar la lista de pacientes. En este caso, la petición devuelve un listado de pacientes que coincidan con la búsqueda y se obtiene información referente al ingreso vinculado con los pacientes y no solo con el paciente.



#### 4.4. Buscar ingreso por el nombre del paciente

Para el servicio de búsqueda de ingreso por nombre, se ingresa el nombre con el que se desea filtrar la lista de pacientes. En este caso, la petición devuelve un listado de pacientes que coincidan con la búsqueda y se obtiene información referente al ingreso vinculado con los pacientes y no solo con el paciente.



```
GET http://localhost:8080/ingreso/find/nombre=maria

Params Headers Body

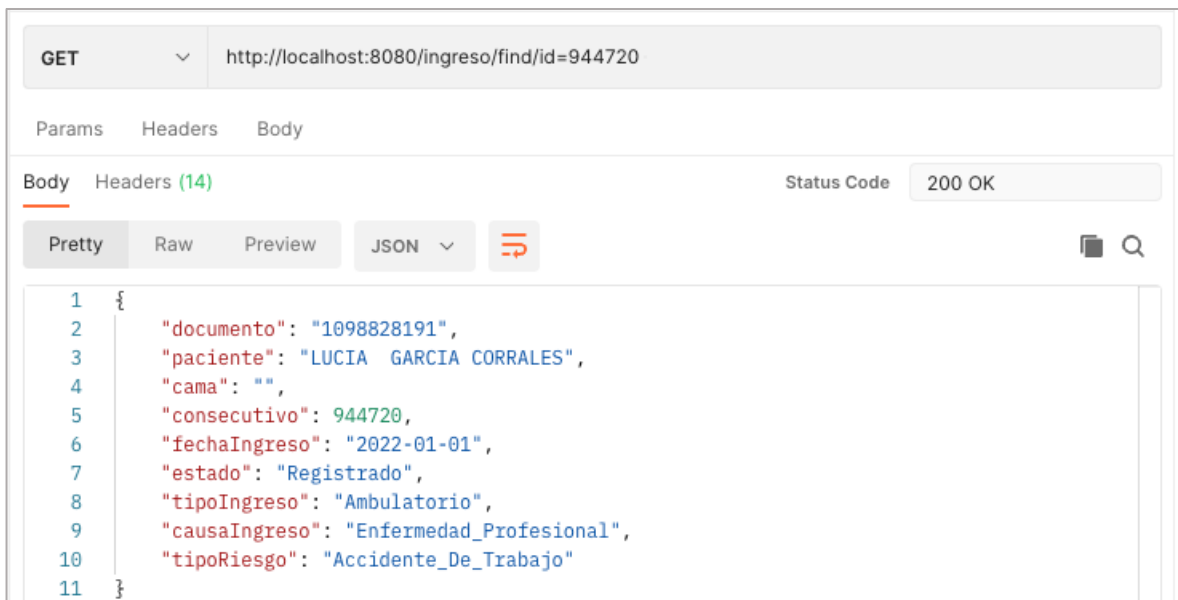
Body Headers (14) Status Code 200 OK

Pretty Raw Preview JSON

[
  {
    "documento": "1098813250",
    "paciente": "ANA MARIA SUAREZ LOPEZ",
    "cama": "",
    "consecutivo": 944725,
    "fechaIngreso": "2022-01-03",
    "estado": "Registrado",
    "tipoIngreso": "Hospitalario",
    "causaIngreso": "Enfermedad_General_Pediatrica",
    "tipoRiesgo": "Enfermedad_Profesional"
  },
  {
    "documento": "92992837",
    "paciente": "ANGELA MARIA CONTRERAS ORDOÑEZ",
    "cama": "",
    "consecutivo": 944724,
    "fechaIngreso": "2022-01-03",
    "estado": "Registrado",
    "tipoIngreso": "Ambulatorio",
    "causaIngreso": "Enfermedad_Profesional",
    "tipoRiesgo": "Enfermedad_Profesional"
  },
  {
    "documento": "30209123",
    "paciente": "MARIA CAROLINA SALAZAR DELGADO",
    "cama": "",
    "consecutivo": 944721,
    "fechaIngreso": "2022-01-01",
    "estado": "Registrado",
    "tipoIngreso": "Hospitalario",
    "causaIngreso": "Enfermedad_Profesional",
    "tipoRiesgo": "Enfermedad_General_Y_Maternidad"
  },
  {
    "documento": "39887163",
    "paciente": "MARIA DE LOS ANGELES SERRANO GONZALEZ",
    "cama": "UCIQ03",
    "consecutivo": 944719,
    "fechaIngreso": "2021-12-05",
    "estado": "Registrado",
    "tipoIngreso": "Hospitalario",
    "causaIngreso": "Enfermedad_General_Pediatrica",
    "tipoRiesgo": "Accidente_De_Trabajo"
  }
]
```

### 4.5. *Buscar ingreso por número de ingreso*

Para el servicio de búsqueda de ingreso por número de ingreso, se ingresa número que identifica el ingreso o “*consecutivo*” con el que se desea filtrar la lista de pacientes. En este caso, la petición devuelve un listado de pacientes que coincidan con la búsqueda y se obtiene información referente al ingreso vinculado con los pacientes y no solo con el paciente.



The screenshot shows a REST client interface with a GET request to `http://localhost:8080/ingreso/find/id=944720`. The response status is 200 OK. The response body is displayed in JSON format, showing details for a patient and their admission.

```
1 {
2   "documento": "1098828191",
3   "paciente": "LUCIA GARCIA CORRALES",
4   "cama": "",
5   "consecutivo": 944720,
6   "fechaIngreso": "2022-01-01",
7   "estado": "Registrado",
8   "tipoIngreso": "Ambulatorio",
9   "causaIngreso": "Enfermedad_Profesional",
10  "tipoRiesgo": "Accidente_De_Trabajo"
11 }
```