

Práctica RESTful

SISTEMAS ORIENTADOS A SERVICIOS

DLSIIS

*Vidal Peña, Arturo
Melero Deza, Javier*

10 de julio de 2019

Índice de contenidos

1 Resumen del diseño del servicio	1
1.1 Métodos de usuarios	2
1.1.1 Crear un usuario	2
1.1.2 Obtener info del usuario	3
1.1.3 Obtener todos los usuarios	4
1.1.4 Modificar un usuario	5
1.1.5 Eliminar un usuario	6
1.2 Métodos de amistad	7
1.2.1 Crear una amistad	7
1.2.2 Obtener amigos de un usuario	8
1.3 Métodos de mensajes	9
1.3.1 Crear un mensaje	9
1.3.2 Obtener info del mensaje	10
1.3.3 Obtener todos los mensajes de un usuario	11
1.3.4 Modificar un mensaje	12
1.3.5 Eliminar un mensaje	13
1.4 Métodos de mensajes privados	14
1.4.1 Crear un mensaje privado	14
1.4.2 Obtener info del mensaje privado	15
1.4.3 Obtener todos los mensajes privados de un usuario	16
1.5 Fichero YAML del servicio	17
1.5.1 Definición de los métodos de usuario	18
1.5.2 Definición de los métodos de mensajes	22
1.5.3 Definición de los métodos de mensajes privados	27
1.5.4 Definición de los métodos de amistades	30
1.5.5 Definición de los esquemas	32
2 Capturas de la ejecución en el cliente Postman	34
2.1 Métodos de usuarios	34
2.1.1 POST	34

2.1.2	GET	36
2.1.3	PUT	37
2.1.4	DELETE	38
2.2	Métodos de mensajes	39
2.2.1	POST	39
2.2.2	GET	41
2.2.3	PUT	42
2.2.4	DELETE	43
2.3	Métodos de Mensajes Privados	44
2.3.1	POST	44
2.3.2	GET	44
2.4	Métodos de Amistad	46
2.4.1	POST	46
2.4.2	GET	47
2.4.3	DELETE	48
2.5	Obtención de mensajes similar a FaceBook	49
2.6	Aplicación móvil	50
3	ANEXO: Problemas encontrados	51

1. Resumen del diseño del servicio

Social API v1 OAS3

A Simple Social API

Servers

http://localhost:8080/social_api/api

User operations:

<code>GET</code>	/usuarios	Get users list.
<code>GET</code>	/usuarios/{id}	Get user info
<code>PUT</code>	/usuarios/{id}	Change user info.
<code>DELETE</code>	/usuarios/{id}	Delete user.
<code>POST</code>	/	Create new user.

Message operations:

<code>GET</code>	/usuarios/{id}/mensajes	Get list of messages associated to a user.
<code>POST</code>	/usuarios/{id}/mensajes	Create a new message associated to the user.
<code>GET</code>	/usuarios/{id}/mensajes/{id_msg}	Get message info.
<code>PUT</code>	/usuarios/{id}/mensajes/{id_msg}	Modify an existing message.
<code>DELETE</code>	/usuarios/{id}/mensajes/{id_msg}	Delete message.

Private message operations:

<code>GET</code>	/usuarios/{id}/privados	Get private messages list.
<code>POST</code>	/usuarios/{id}/privados	Create new private message.
<code>GET</code>	/usuarios/{id}/privados/{id_msg}	Get private message info.

User friends operations:

<code>GET</code>	/usuarios/{id}/amigos	Get friends' list.
<code>POST</code>	/usuarios/{id}/amigos	Add a friend.

Schemas

>

1.1. Métodos de usuarios

1.1.1. Crear un usuario

POST / Create new user.

Parameters

No parameters

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Try it out

Media type: application/json

Controls Accept header.

Example Value Schema

```
{ "nombre": "javi", "rol": "self", "url": "http://localhost:8080/social_api/api/users/2" }
```

Media type: application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value Schema

```
ERROR 404: NOT FOUND
```

Media type: application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.1.2. Obtener info del usuario

GET /usuarios Get users list.

Parameters

Name	Description
indicio_n <small>string (query)</small>	Substring of the users we want in the list. indicio_n - Substring of the users we want in the list.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
500	Internal Server Error.	No links

Try it out

Media type: application/json

Example Value: Schema

```
{ "users": [ { "nombre": "Javi", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/1" }, { "nombre": "Javilon", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/2" } ] }
```

Media type: application/json

Example Value: Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value: Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.1.3. Obtener todos los usuarios

GET /usuarios/{id} Get user info

Parameters

Name Description

Id required Id of the user we want the info.

integer (path)

Try it out

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Media type application/json

Example Value Schema

```
{ "aficiones": "juega al apex", "edad": 22, "id": 1, "nombre": "Javi", "nombre_usuario": "javilines" }
```

Media type application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

Media type application/json

Example Value Schema

```
ERROR 404: NOT FOUND
```

Media type application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.1.4. Modificar un usuario

PUT /usuarios/{id} Change user info.

Try It out

Parameters

Name	Description
id <small>required</small> integer (path)	The user's id. id - The user's id.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
500	Internal Server Error.	No links

Media type: application/json

Example Value: Schema

```
{  
  "nombre": "Javi",  
  "rol": "self",  
  "url": "http://localhost:8080/social_api/api/usuarios/1"  
}
```

Media type: application/json

Example Value: Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value: Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.1.5. Eliminar un usuario

DELETE /usuarios/{id} Delete user.

Parameters

Name Description
id required The user's id.
integer
(path) id - The user's id.

Responses

Code	Description	Links
200	OK. Media type: application/json Example Value Schema: <code>200: OK</code>	No links
400	Bad Request. Media type: application/json Example Value Schema: <code>ERROR 400: BAD REQUEST</code>	No links
404	Not Found. Media type: application/json Example Value Schema: <code>ERROR 404: NOT FOUND</code>	No links
500	Internal Server Error. Media type: application/json Example Value Schema: <code>ERROR 500: INTERNAL SERVER ERROR</code>	No links

Try it out

1.2. Métodos de amistad

1.2.1. Crear una amistad

POST /usuarios/{id}/amigos Add a friend.

Parameters

Name Description

id required User id.
integer
(path) id - User id.

Responses

Code	Description	Links
200	OK	No links
	Media type application/json Controls Accept header.	
	Example Value Schema	
	{ "users": [{ "nombre": "amigo", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/1" }, { "nombre": "amigo", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/3" }] }	
400	Bad Request.	No links
	Media type application/json Controls Accept header.	
	Example Value Schema	
	ERROR 400: BAD REQUEST	
500	Internal Server Error.	No links
	Media type application/json Controls Accept header.	
	Example Value Schema	
	ERROR 500: INTERNAL SERVER ERROR	

1.2.2. Obtener amigos de un usuario

GET /usuarios/{id}/amigos Get friends' list.

Parameters

Name Description

id required User id.
integer
(path) id - User id.

Responses

Code	Description	Links
200	OK.	No links
	Media type application/json	
	Example Value Schema	
	<pre>{ "users": [{ "nombre": "amigo", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/1" }, { "nombre": "amigo", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/3" }] }</pre>	
400	Bad Request.	No links
	Media type application/json	
	Example Value Schema	
	<pre>ERROR 400: BAD REQUEST</pre>	
500	Internal Server Error.	No links
	Media type application/json	
	Example Value Schema	
	<pre>ERROR 500: INTERNAL SERVER ERROR</pre>	

1.3. Métodos de mensajes

1.3.1. Crear un mensaje

POST /usuarios/{id}/mensajes Create a new message associated to the user.

Parameters

Name	Description
Id <small>* required</small> <small>integer (path)</small>	Id of the user that creates the message. id - Id of the user that creates the message.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
500	Internal Server Error.	No links

Try it out

Media type: application/json

Controls Accept header.

Example Value Schema

```
{  
  "users": [  
    {  
      "real": "self",  
      "url": "http://localhost:8080/social_api/api/usuarios/1/mensajes/3"  
    }  
  ]  
}
```

Media type: application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.3.2. Obtener info del mensaje

GET /usuarios/{id}/mensajes/{id_msg} Get message info.

Parameters

Try it out

Name	Description
id <small>required</small> integer (path)	User id. id - User id.
id_msg <small>required</small> integer (path)	Id of the specific message of the user. id_msg - Id of the specific message of the user.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Media type: application/json

Example Value Schema

```
{ "contenido": "hola me llamo pingu", "fecha_envio": "2019-04-21T16:23:19.507Z(UTC)", "id": 1, "id_usuario": 1 }
```

Media type: application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value Schema

```
ERROR 404: NOT FOUND
```

Media type: application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.3.3. Obtener todos los mensajes de un usuario

GET /usuarios/{id}/mensajes Get list of messages associated to a user.

Parameters

Name	Description
id <small>required</small> <small>integer (path)</small>	The id of the user from whom retrieve the messages. id - The id of the user from whom retrieve the messages.

Responses

Code	Description	Links
200	OK	No links
	Media type application/json	
	Example Value Schema	
	<pre>{ "users": [{ "nombre": "amigo", "rel": "self", "url": "http://localhost:8080/social_api/api/usuarios/1" }] }</pre>	
400	Bad Request.	No links
	Media type application/json	
	Example Value Schema	
	<pre>ERROR 400: BAD REQUEST</pre>	
404	Not Found.	No links
	Media type application/json	
	Example Value Schema	
	<pre>ERROR 404: NOT FOUND</pre>	
500	Internal Server Error.	No links
	Media type application/json	
	Example Value Schema	
	<pre>ERROR 500: INTERNAL SERVER ERROR</pre>	

1.3.4. Modificar un mensaje

PUT /usuarios/{id}/mensajes/{id_msg} Modify an existing message.

Parameters

Name	Description
Id <small>required</small> Integer (path)	User id. User id.
Id_msg <small>required</small> Integer (path)	Id of the specific message of the user. id_msg - Id of the specific message of the user.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Try it out

Media type: application/json

Example Value: Schema

```
{  
  "rel": "self",  
  "url": "http://localhost:8888/social_api/api/usuarios/1/mensajes/2"  
}
```

Media type: application/json

Example Value: Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value: Schema

```
ERROR 404: NOT FOUND
```

Media type: application/json

Example Value: Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.3.5. Eliminar un mensaje

DELETE /usuarios/{id}/mensajes/{id_msg} Delete message.

Parameters

Name **Description**

Id **required** User id.
integer
(path) **id** - User id.

id_msg **required** Id of the specific message of the user.
integer
(path) **id_msg** - Id of the specific message of the user.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Media type application/json
Controls Accept header.

Example Value **Schema**

```
{  
    "rel": "self",  
    "url": "http://localhost:8080/social_api/api/usuarios/1/mensajes/1"  
}
```

Media type application/json
Controls Accept header.

Example Value **Schema**

```
ERROR 400: BAD REQUEST
```

Media type application/json
Controls Accept header.

Example Value **Schema**

```
ERROR 404: NOT FOUND
```

Media type application/json
Controls Accept header.

Example Value **Schema**

```
ERROR 500: INTERNAL SERVER ERROR
```

1.4. Métodos de mensajes privados

1.4.1. Crear un mensaje privado

POST /usuarios/{id}/privados Create new private message.

Parameters

Name Description

Id * required User id.
integer
(path)
id - User id.

Responses

Code Description Links

200 OK No links

Media type application/json
Controls Accept header.

Example Value Schema

```
{  
  "rel": "self",  
  "url": "http://localhost:8080/social_api/api/usuarios/3/privados/4"  
}
```

400 Bad Request. No links

Media type application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

500 Internal Server Error. No links

Media type application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.4.2. Obtener info del mensaje privado

GET /usuarios/{id}/privados/{id_msg} Get private message info.

Parameters

Try it out

Name	Description
id <small>required</small> <small>integer</small> (path)	User id. Id - User id.
id_msg <small>required</small> <small>integer</small> (path)	Id of the specific private message of the user. Id_msg - Id of the specific private message of the user.

Responses

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
404	Not Found.	No links
500	Internal Server Error.	No links

Media type: application/json

Example Value Schema

```
{ "contenido": "como estas", "destinatario": 3, "fecha_envio": "2019-07-09T22:00:00Z[UTC]", "id": 4 }
```

Media type: application/json

Example Value Schema

```
ERROR 400: BAD REQUEST
```

Media type: application/json

Example Value Schema

```
ERROR 404: NOT FOUND
```

Media type: application/json

Example Value Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.4.3. Obtener todos los mensajes privados de un usuario

The screenshot shows a REST API documentation page for the `/usuarios/{id}/privados` endpoint. The endpoint is described as "Get private messages list".

Parameters:

Name	Description
Id <small>required</small>	User id.
Integer	(path)
id - User id.	

Responses:

Code	Description	Links
200	OK	No links
400	Bad Request.	No links
500	Internal Server Error.	No links

Example Value: application/json

Example Value: Schema

```
{  "privates": [    {      "rel": "self",      "url": "http://localhost:8080/social_api/api/usuarios/3/privados/1"    },    {      "rel": "self",      "url": "http://localhost:8080/social_api/api/usuarios/3/privados/2"    },    {      "rel": "self",      "url": "http://localhost:8080/social_api/api/usuarios/3/privados/3"    }  ]}
```

Example Value: application/json

Example Value: Schema

```
ERROR 400: BAD REQUEST
```

Example Value: application/json

Example Value: Schema

```
ERROR 500: INTERNAL SERVER ERROR
```

1.5. Fichero YAML del servicio

El siguiente código se ha realizado usando la herramienta Swagger Editor. El fichero se encuentra adjunto a este documento y las capturas previas.

```
1 openapi: '3.0.0'
2 info:
3   description: A Simple Social API
4   title: Social API
5   version: v1
6
7 servers:
8   - url: 'http://localhost:8080/social_api/api'
9
10 paths:
```

1.5.1. Definición de los métodos de usuario

```
1 /usuarios:
2   get:
3     parameters:
4       - name: indicion
5         in: query
6         description: "Substring of the users we want in
7           the list."
8       schema:
9         type: string
10      tags:
11        - "User operations"
12      summary: "Get users list."
13      responses:
14        '200':
15          description: OK
16          content:
17            application/json:
18              schema:
19                $ref: '#/components/schemas/Users'
20            example:
21              users:
22                - nombre: "Javi"
23                  rel: "self"
24                  url: "http://localhost:8080/social_api/
25                    api/usuarios/1"
26                - nombre: "Javilon"
27                  rel: "self"
28                  url: "http://localhost:8080/social_api/
29                    api/usuarios/2"
30
31        '400':
32          description: Bad Request .
33          content:
34            application/json:
35              example: 'ERROR 400: BAD REQUEST'
36        '500':
37          description: Internal Server Error .
38          content:
39            application/json:
40              example: 'ERROR 500: INTERNAL SERVER ERROR'
41
42 /usuarios/{id}:
43   get:
44     parameters:
45       - name: id
46         in: path
47         description: 'Id of the user we want the info.'
```

```

45     schema:
46         type: integer
47         required: true
48     tags:
49         - "User operations"
50     summary: 'Get user info'
51     responses:
52         '200':
53             description: OK
54             content:
55                 application/json:
56                     schema:
57                         $ref: '#/components/schemas/UserInfo'
58             example:
59                 aficiones: "juega al apex"
60                 edad: 22
61                 id: 1
62                 nombre: "Javi"
63                 nombre_usuario: "javilinos"
64         '400':
65             description: Bad Request .
66             content:
67                 application/json:
68                     example: 'ERROR 400: BAD REQUEST'
69         '404':
70             description: Not Found .
71             content:
72                 application/json:
73                     example: 'ERROR 404: NOT FOUND'
74         '500':
75             description: Internal Server Error .
76             content:
77                 application/json:
78                     example: 'ERROR 500: INTERNAL SERVER ERROR'
79
80     put:
81         parameters:
82             - name: id
83                 in: path
84                 description: "The user's id."
85             schema:
86                 type: integer
87                 required: true
88         tags:
89             - "User operations"
90         summary: "Change user info."
91         responses:
92             '200':

```

```

93     description: OK
94     content:
95         application/json:
96             schema:
97                 $ref: '#/components/schemas/UserInfo'
98             example:
99                 nombre: "Javi"
100                rel: "self"
101                url: "http://localhost:8080/social_api/api/
102                  usuarios/1"
103            '400':
104                description: Bad Request .
105                content:
106                    application/json:
107                        example: 'ERROR 400: BAD REQUEST'
108            '500':
109                description: Internal Server Error .
110                content:
111                    application/json:
112                        example: 'ERROR 500: INTERNAL SERVER ERROR'
113
114 delete:
115     parameters:
116         - name: id
117             in: path
118             description: "The user's id."
119             schema:
120                 type: integer
121                 required: true
122             tags:
123                 - "User operations"
124             summary: "Delete user."
125             responses:
126                 '200':
127                     description: OK
128                     content:
129                         application/json:
130                             example: "200: OK"
131                 '400':
132                     description: Bad Request .
133                     content:
134                         application/json:
135                             example: 'ERROR 400: BAD REQUEST'
136                 '404':
137                     description: Not Found .
138                     content:
139                         application/json:

```

```

140      '500':
141          description: Internal Server Error.
142          content:
143              application/json:
144                  example: 'ERROR 500: INTERNAL SERVER ERROR'
145
146      /:
147          post:
148              tags:
149                  - "User operations"
150              summary: "Create new user."
151              responses:
152                  '200':
153                      description: OK
154                      content:
155                          application/json:
156                          schema:
157                              $ref: '#/components/schemas/User'
158                          example:
159                              nombre: "javi"
160                              rel: "self"
161                              url: "http://localhost:8080/social_api/api/
162                                  usuarios/2"
163
164                  '400':
165                      description: Bad Request.
166                      content:
167                          application/json:
168                              example: 'ERROR 400: BAD REQUEST'
169
170                  '404':
171                      description: Not Found.
172                      content:
173                          application/json:
174                              example: 'ERROR 404: NOT FOUND'
175
176                  '500':
177                      description: Internal Server Error.
178                      content:
179                          application/json:
180                              example: 'ERROR 500: INTERNAL SERVER ERROR'

```

1.5.2. Definición de los métodos de mensajes

```
1 /usuarios/{id}/mensajes:
2   get:
3     parameters:
4       - name: id
5         in: path
6         description: "The id of the user from whom
7           retrieve the messages."
8       schema:
9         type: integer
10        required: true
11     tags:
12       - "Message operations"
13     summary: "Get list of messages associated to a user."
14     responses:
15       '200':
16         description: OK
17         content:
18           application/json:
19             schema:
20               $ref: '#/components/schemas/Users'
21             example:
22               users:
23                 - nombre: "amigo"
24                   rel: "self"
25                   url: "http://localhost:8080/social_api/
26                     api/usuarios/1"
27       '400':
28         description: Bad Request .
29         content:
30           application/json:
31             example: 'ERROR 400: BAD REQUEST'
32       '404':
33         description: Not Found .
34         content:
35           application/json:
36             example: 'ERROR 404: NOT FOUND'
37       '500':
38         description: Internal Server Error .
39         content:
40           application/json:
41             example: 'ERROR 500: INTERNAL SERVER ERROR'
42     post:
43       parameters:
44         - name: id
45           in: path
46           description: "Id of the user that creates the
47             message ."
```

```

45     schema:
46         type: integer
47         required: true
48     tags:
49         - "Message operations"
50     summary: "Create a new message associated to the user
51         ."
52     responses:
53         '200':
54             description: OK
55             content:
56                 application/json:
57                     schema:
58                         $ref: '#/components/schemas/Users'
59             example:
60                 users:
61                     - rel: "self"
62                         url: "http://localhost:8080/social_api/
63                             api/usuarios/1/mensajes/3"
64         '400':
65             description: Bad Request .
66             content:
67                 application/json:
68                     example: 'ERROR 400: BAD REQUEST'
69         '500':
70             description: Internal Server Error .
71             content:
72                 application/json:
73                     example: 'ERROR 500: INTERNAL SERVER ERROR'
74
75     /usuarios/{id}/mensajes/{id_msg}:
76         get:
77             parameters:
78                 - name: id
79                     in: path
80                     description: 'User id .'
81                     schema:
82                         type: integer
83                         required: true
84                 - name: id_msg
85                     in: path
86                     description: 'Id of the specific message of the
87                         user .'
88                     schema:
89                         type: integer
90                         required: true
91             tags:
92                 - "Message operations"

```

```

90     summary: "Get message info."
91     responses:
92       '200':
93         description: OK
94         content:
95           application/json:
96             schema:
97               $ref: '#/components/schemas/Message'
98             example:
99               contenido: "hola me llamo pingu"
100              fecha_envio: "2019-04-21T16:23:13.50Z[UTC]"
101              "
102              id: 1
103              id_usuario: 1
104        '400':
105          description: Bad Request .
106          content:
107            application/json:
108              example: "ERROR 400: BAD REQUEST"
109        '404':
110          description: Not Found .
111          content:
112            application/json:
113              example: "ERROR 404: NOT FOUND"
114        '500':
115          description: Internal Server Error .
116          content:
117            application/json:
118              example: "ERROR 500: INTERNAL SERVER ERROR"
119
120     put:
121       parameters:
122         - name: id
123           in: path
124           description: 'User id .'
125           schema:
126             type: integer
127             required: true
128         - name: id_msg
129           in: path
130           description: 'Id of the specific message of the
131             user .'
132           schema:
133             type: integer
134             required: true
135       tags:
136         - "Message operations :"
137       summary: "Modify an existing message ."

```

```

136     responses:
137         '200':
138             description: OK
139             content:
140                 application/json:
141                     schema:
142                         $ref: '#/components/schemas/Message'
143             example:
144                 rel: "self"
145                 url: "http://localhost:8080/social_api/api/
146                     usuarios/1/mensajes/2"
147
148         '400':
149             description: Bad Request .
150             content:
151                 application/json:
152                     example: "ERROR 400: BAD REQUEST"
153
154         '404':
155             description: Not Found .
156             content:
157                 application/json:
158                     example: "ERROR 404: NOT FOUND"
159
160         '500':
161             description: Internal Server Error .
162             content:
163                 application/json:
164                     example: "ERROR 500: INTERNAL SERVER ERROR"
165
166
167     delete:
168         parameters:
169             - name: id
170                 in: path
171                 description: 'User id .'
172                 schema:
173                     type: integer
174                     required: true
175             - name: id_msg
176                 in: path
177                 description: 'Id of the specific message of the
178                     user .'
179                 schema:
180                     type: integer
181                     required: true
182
183         tags:
184             - "Message operations :"
185         summary: "Delete message ."
186         responses:
187             '200':
188                 description: OK

```

```
182     content:
183       application/json:
184         schema:
185           $ref: '#/components/schemas/Message'
186         example:
187           rel: "self"
188           url: "http://localhost:8080/social_api/api/
189             usuarios/1/mensajes/1"
190           '400':
191             description: Bad Request.
192             content:
193               application/json:
194                 example: "ERROR 400: BAD REQUEST"
195           '404':
196             description: Not Found.
197             content:
198               application/json:
199                 example: "ERROR 404: NOT FOUND"
200           '500':
201             description: Internal Server Error.
202             content:
203               application/json:
204                 example: "ERROR 500: INTERNAL SERVER ERROR"
```

1.5.3. Definición de los métodos de mensajes privados

```
1 /usuarios/{id}/privados:
2   get:
3     parameters:
4       - name: id
5         in: path
6         description: 'User id.'
7         schema:
8           type: integer
9           required: true
10    tags:
11      - "Private message operations"
12    summary: "Get private messages list."
13    responses:
14      '200':
15        description: OK
16        content:
17          application/json:
18            schema:
19              $ref: '#/components/schemas/Message'
20            example:
21              privates:
22                - rel: "self"
23                  url: "http://localhost:8080/social_api/
24                    api/usuarios/3/privados/1"
25                - rel: "self"
26                  url: "http://localhost:8080/social_api/
27                    api/usuarios/3/privados/2"
28                - rel: "self"
29                  url: "http://localhost:8080/social_api/
30                    api/usuarios/3/privados/3"
31      '400':
32        description: Bad Request .
33        content:
34          application/json:
35            example: "ERROR 400: BAD REQUEST"
36      '500':
37        description: Internal Server Error .
38        content:
39          application/json:
40            example: "ERROR 500: INTERNAL SERVER ERROR"
41
42    post:
43      parameters:
44        - name: id
45          in: path
46          description: 'User id.'
47          schema:
```

```

45         type: integer
46         required: true
47     tags:
48         - "Private message operations"
49     summary: "Create new private message."
50     responses:
51         '200':
52             description: OK
53             content:
54                 application/json:
55                     schema:
56                         $ref: '#/components/schemas/Message'
57                     example:
58                         rel: "self"
59                         url: "http://localhost:8080/social_api/api/
60                             usuarios/3/privados/4"
61         '400':
62             description: Bad Request .
63             content:
64                 application/json:
65                     example: "ERROR 400: BAD REQUEST"
66         '500':
67             description: Internal Server Error .
68             content:
69                 application/json:
70                     example: "ERROR 500: INTERNAL SERVER ERROR"
71
72     /usuarios/{id}/privados/{id_msg}:
73         get:
74             parameters:
75                 - name: id
76                     in: path
77                     description: 'User id .'
78                     schema:
79                         type: integer
80                         required: true
81                 - name: id_msg
82                     in: path
83                     description: 'Id of the specific private message
84                         of the user .'
85                     schema:
86                         type: integer
87                         required: true
88             tags:
89                 - "Private message operations"
90             summary: "Get private message info."
91             responses:
92                 '200':

```

```
91      description: OK
92      content:
93          application/json:
94              schema:
95                  $ref: '#/components/schemas/Message'
96      example:
97          contenido: "como estas"
98          destinatario: 3
99          fecha_envio: "2019-07-09T22:00:00Z[UTC]"
100         id: 4
101     '400':
102         description: Bad Request .
103         content:
104             application/json:
105                 example: "ERROR 400: BAD REQUEST"
106     '404':
107         description: Not Found .
108         content:
109             application/json:
110                 example: "ERROR 404: NOT FOUND"
111     '500':
112         description: Internal Server Error .
113         content:
114             application/json:
115                 example: "ERROR 500: INTERNAL SERVER ERROR"
```

1.5.4. Definición de los métodos de amistades

```
1 /usuarios/{id}/amigos:
2   get:
3     parameters:
4       - name: id
5         in: path
6         description: 'User id.'
7       schema:
8         type: integer
9         required: true
10      tags:
11        - "User friends operations"
12      summary: "Get friends' list."
13      responses:
14        '200':
15          description: OK
16          content:
17            application/json:
18              schema:
19                $ref: '#/components/schemas/Message'
20            example:
21              users:
22                - nombre: "amigo"
23                  rel: "self"
24                  url: "http://localhost:8080/social_api/
25                      api/usuarios/1"
26                - nombre: "amigo"
27                  rel: "self"
28                  url: "http://localhost:8080/social_api/
29                      api/usuarios/3"
30        '400':
31          description: Bad Request .
32          content:
33            application/json:
34              example: "ERROR 400: BAD REQUEST"
35        '500':
36          description: Internal Server Error .
37          content:
38            application/json:
39              example: "ERROR 500: INTERNAL SERVER ERROR"
40
41      post:
42        parameters:
43          - name: id
44            in: path
45            description: 'User id.'
46          schema:
47            type: integer
```

```

46     required: true
47   tags:
48     - "User friends operations"
49   summary: "Add a friend."
50   responses:
51     '200':
52       description: OK
53       content:
54         application/json:
55           schema:
56             $ref: '#/components/schemas/Message'
57       example:
58         users:
59           - nombre: "amigo"
60             rel: "self"
61             url: "http://localhost:8080/social_api/
62               api/usuarios/1"
63           - nombre: "amigo"
64             rel: "self"
65             url: "http://localhost:8080/social_api/
66               api/usuarios/3"
67     '400':
68       description: Bad Request.
69       content:
70         application/json:
71           example: "ERROR 400: BAD REQUEST"
72     '500':
73       description: Internal Server Error.
74       content:
75         application/json:
76           example: "ERROR 500: INTERNAL SERVER ERROR"

```

1.5.5. Definición de los esquemas

```
1 components:
2   schemas:
3     Users:
4       type: array
5       items:
6         anyOf:
7           - $ref: "#/components/schemas/User"
8
9     UserInfo:
10    type: object
11    properties:
12      aficiones:
13        type: string
14        description: "The user's hobbies.."
15      edad:
16        type: integer
17        description: "The user's age."
18      id:
19        type: integer
20        description: "The user's id."
21      nombre:
22        type: string
23        description: "The user's first name."
24      nombre_usuario:
25        type: string
26        description: "The user's username."
27
28     User:
29       type: object
30       properties:
31         nombre:
32           type: string
33           description: "The user's first name."
34         rel:
35           type: string
36           description: "The relationship with the calling
37             user."
38         url:
39           type: string
40           description: "The user's URL."
41
42     Message:
43       type: object
44       properties:
45         contenido:
46           type: string
47           description: "The message's contents."
```

```
47     fecha:  
48         type: string  
49         description: "The date the message was sent."  
50     id:  
51         type: integer  
52         description: "The message identifier."  
53     id_usuario:  
54         type: integer  
55         description: "The id of the user who sent the  
message."
```

2. Capturas de la ejecución en el cliente Postman

Hay que poner detalles tanto de la invocación como del resultado de la operación.

2.1. Métodos de usuarios

2.1.1. POST

The screenshot shows the Postman application interface. In the top navigation bar, 'POST' is selected under 'Method'. The URL is set to 'http://localhost:8080/social_api/api/usuarios'. The 'Body' tab is active, showing a JSON payload:

```
1 < {
2   "apellidos": "astronom",
3   "edad": 39,
4   "nombre": "Raquel",
5   "nombre_usuario": "Raquelena"
6 }
```

Below the body, the 'Test Results' section displays the response status: 'Status: 201 Created'. The response body is shown in JSON format:

```
1 < {
2   "apellidos": "astronom",
3   "edad": 39,
4   "nombre": "Raquel",
5   "url": "http://localhost:8080/social_api/api/usuarios/1"
6 }
```

Figura 1: Se añade el usuario 1 de nombre Raquel.

The screenshot shows the Postman application interface. In the top navigation bar, 'POST' is selected under 'Method'. The URL is set to 'http://localhost:8080/social_api/api/usuarios'. The 'Body' tab is active, showing a JSON payload:

```
1 < {
2   "apellidos": "Teer",
3   "edad": 22,
4   "nombre": "Javi",
5   "nombre_usuario": "javilimes"
6 }
```

Below the body, the 'Test Results' section displays the response status: 'Status: 201 Created'. The response body is shown in JSON format:

```
1 < {
2   "apellidos": "Teer",
3   "edad": 22,
4   "nombre": "Javi",
5   "url": "http://localhost:8080/social_api/api/usuarios/2"
6 }
```

Figura 2: Se añade el usuario 2 de nombre Javi.

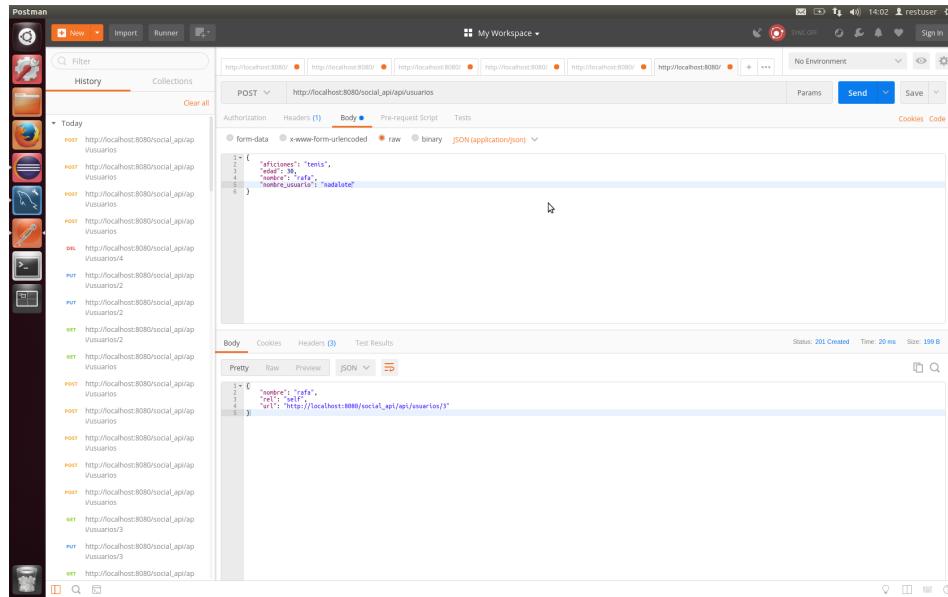


Figura 3: Se añade el usuario 3 de nombre Rafa.

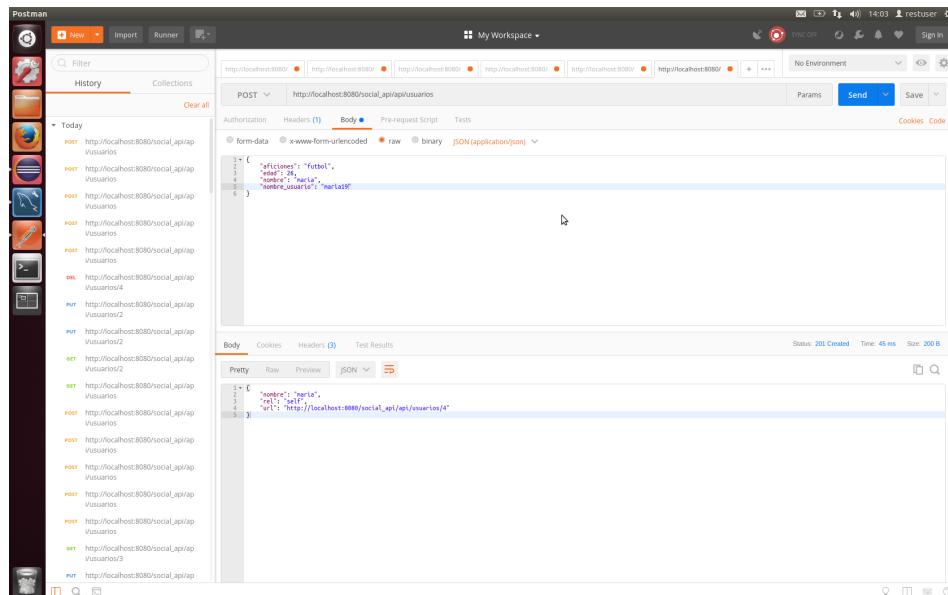


Figura 4: Se añade el usuario 4 de nombre María.

2.1.2. GET

```

[{"id": 1, "name": "raquel", "email": "raquel@social.com", "url": "http://localhost:8080/social_api/api/usuarios/1"}, {"id": 2, "name": "javi", "email": "javi@social.com", "url": "http://localhost:8080/social_api/api/usuarios/2"}, {"id": 3, "name": "rafa", "email": "rafa@social.com", "url": "http://localhost:8080/social_api/api/usuarios/3"}, {"id": 4, "name": "maria", "email": "maria@social.com", "url": "http://localhost:8080/social_api/api/usuarios/4"}]
    
```

Figura 5: Se obtienen todos los usuarios.

```

{"id": 3, "name": "raquel", "email": "raquel@social.com", "url": "http://localhost:8080/social_api/api/usuarios/3", "last_update": "2019-07-09T22:00:00Z[UTC]"}
    
```

Figura 6: Obtención del usuario 3 después de ser modificado.

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/social_api/api/usuarios`. The response status is 200 OK. The response body is a JSON array containing five user objects:

```

[{"id": 1, "nombre": "raquel", "url": "http://localhost:8080/social_api/api/usuarios/1"}, {"id": 2, "nombre": "jose", "url": "http://localhost:8080/social_api/api/usuarios/2"}, {"id": 3, "nombre": "rafa", "url": "http://localhost:8080/social_api/api/usuarios/3"}, {"id": 4, "nombre": "laura", "url": "http://localhost:8080/social_api/api/usuarios/4"}, {"id": 5, "nombre": "ana", "url": "http://localhost:8080/social_api/api/usuarios/5"}]

```

Figura 7: Se obtienen todos los usuarios después de eliminar el usuario 4 (Figura 10).

2.1.3. PUT

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/social_api/api/usuarios/3`. The request body is a JSON object:

```

{
  "nombre": "kent",
  "edad": 36,
  "fecha_creacion": "2019-07-09T22:00:00Z[UTC]",
  "id": 3,
  "nombre_usuario": "sedalote33"
}

```

The response status is 201 Created. The response body is a JSON object:

```

{
  "id": 3,
  "nombre": "rafa",
  "edad": 36,
  "url": "http://localhost:8080/social_api/api/usuarios/3"
}

```

Figura 8: Modificación del nombre de usuario 3.

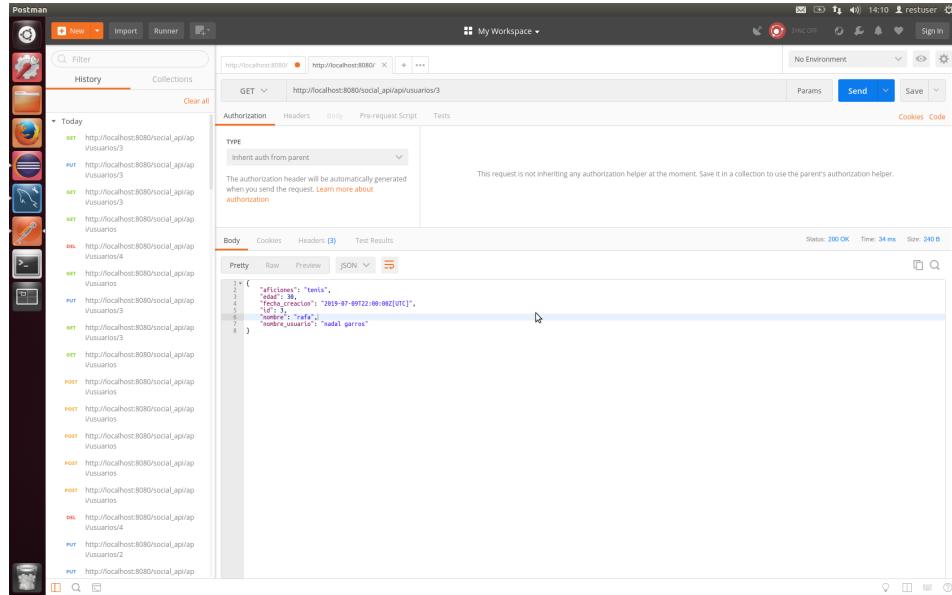


Figura 9: Obtención de los datos después de la modificación anterior.

2.1.4. DELETE

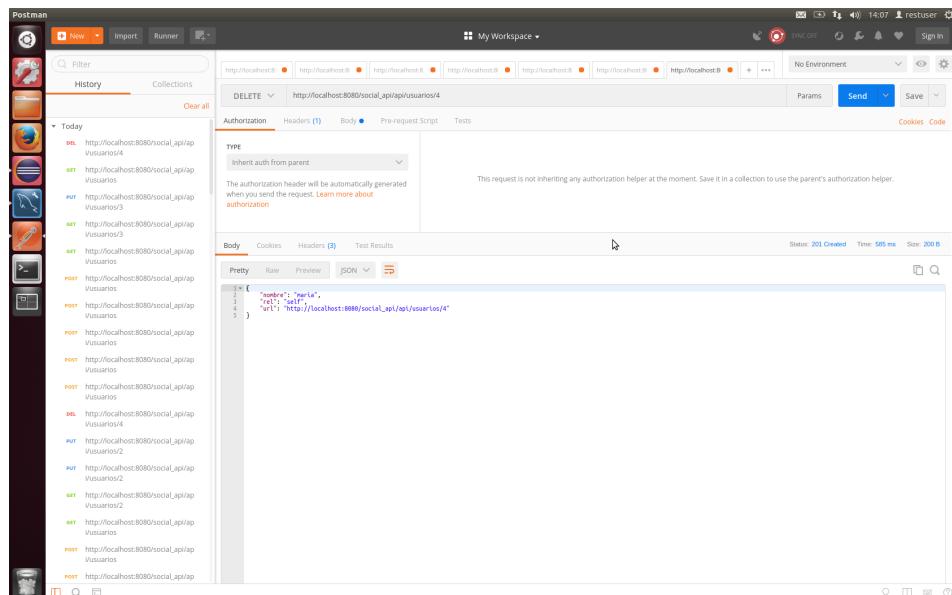


Figura 10: Eliminación del usuario 4.

2.2. Métodos de mensajes

2.2.1. POST

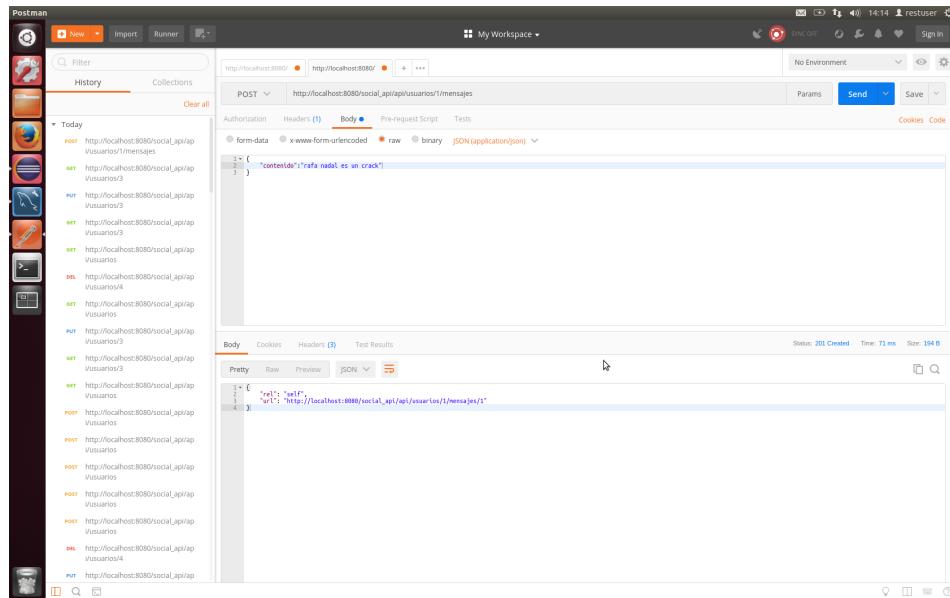


Figura 11: Creación del mensaje 1 asociado al usuario 1.

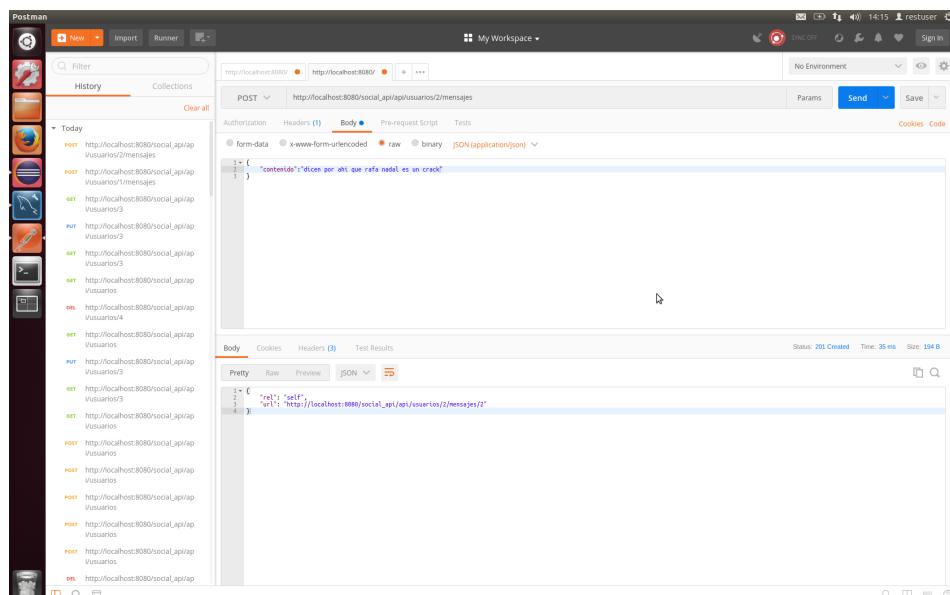


Figura 12: Creación del mensaje 2 asociado al usuario 2.

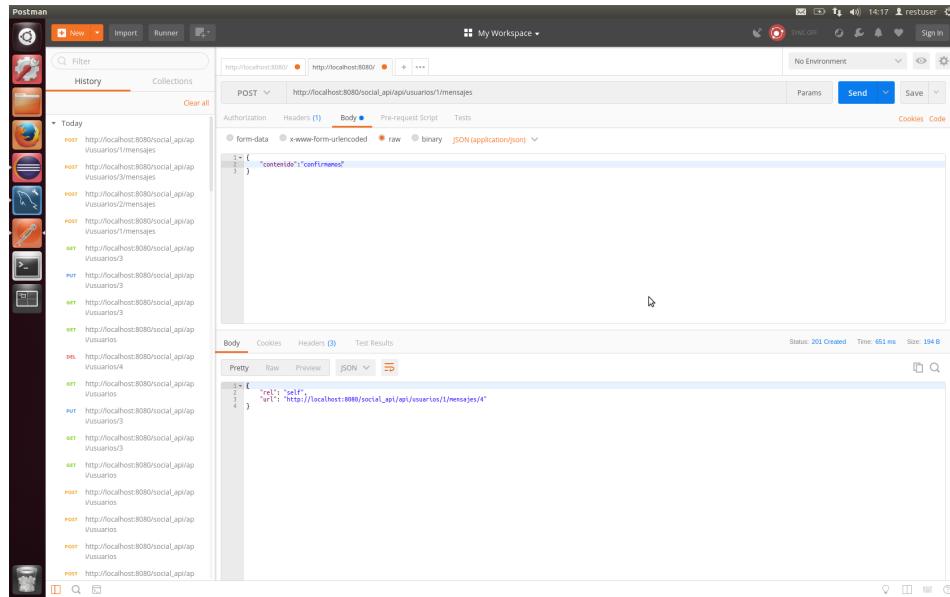


Figura 13: Creación del mensaje 4 asociado al usuario 1.

2.2.2. GET

The screenshot shows the Postman application interface. In the top bar, there are tabs for 'My Workspace' and a status indicator '14:18'. Below the tabs, there are buttons for 'Sync Env', 'Sign in', and a gear icon. The main area shows a list of requests on the left and the details of the current request on the right.

Request Details:

- Method: GET
- URL: http://localhost:8080/social_api/api/usuarios/1/messages
- Authorization: Inherit auth from parent
- Body: JSON

Response Body:

```

[{"id": 1, "text": "self", "url": "http://localhost:8080/social_api/api/usuarios/1/mensaje/1"}, {"id": 2, "text": "self", "url": "http://localhost:8080/social_api/api/usuarios/1/mensaje/2"}, {"id": 3, "text": "self", "url": "http://localhost:8080/social_api/api/usuarios/1/mensaje/3"}, {"id": 4, "text": "self", "url": "http://localhost:8080/social_api/api/usuarios/1/mensaje/4"}]
  
```

The status bar at the bottom indicates: Status: 200 OK, Time: 62 ms, Size: 267 B.

Figura 14: Se obtienen todos mensajes asociados al usuario 1.

The screenshot shows the Postman application interface. In the top bar, there are tabs for 'My Workspace' and a status indicator '14:19'. Below the tabs, there are buttons for 'Sync Env', 'Sign in', and a gear icon. The main area shows a list of requests on the left and the details of the current request on the right.

Request Details:

- Method: GET
- URL: http://localhost:8080/social_api/api/usuarios/1/mensaje/1
- Authorization: Inherit auth from parent
- Body: JSON

Response Body:

```

{
  "contenido": "se ha nado en un crack",
  "fecha_envio": "2019-07-09T22:40:00Z[UTC]",
  "id": 1,
  "id_usuario": 1
}
  
```

The status bar at the bottom indicates: Status: 200 OK, Time: 47 ms, Size: 211 B.

Figura 15: Obtención del mensaje 1 del usuario 1.

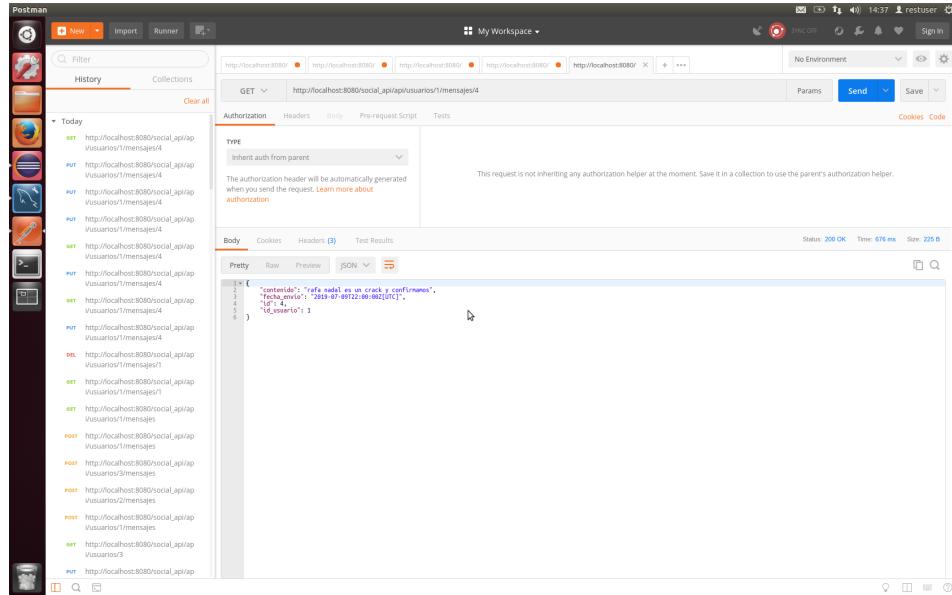


Figura 16: Obtención del mensaje 4 después de ser modificado (Figura 17).

2.2.3. PUT

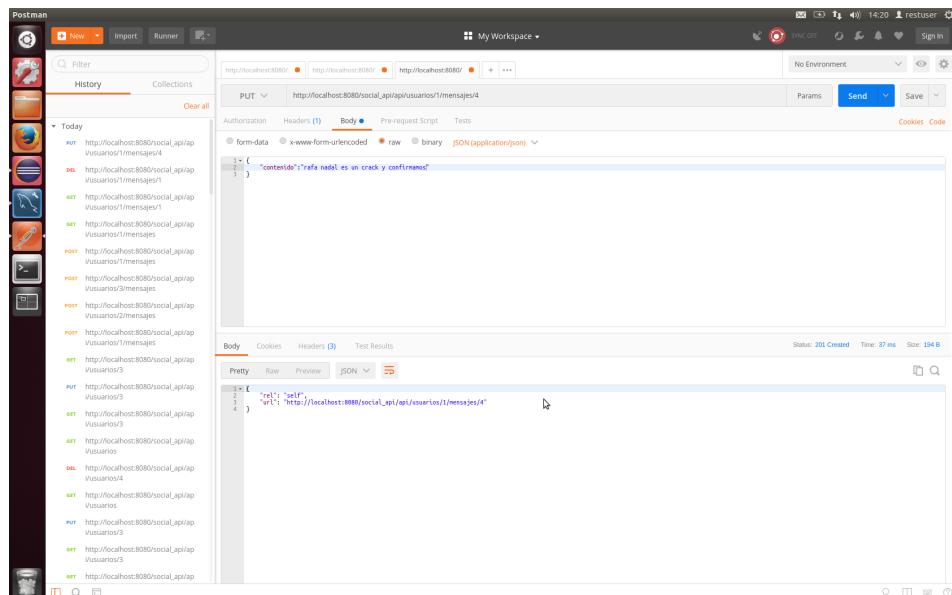


Figura 17: Modificación del mensaje 4 asociado al usuario 1.

2.2.4. DELETE

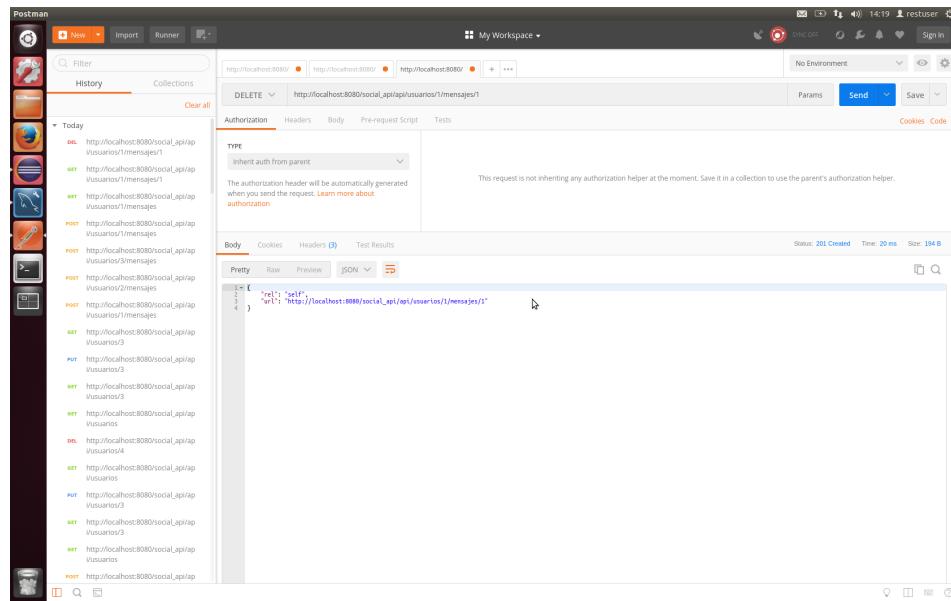


Figura 18: Eliminación del mensaje 1 del usuario 1.

2.3. Métodos de Mensajes Privados

2.3.1. POST

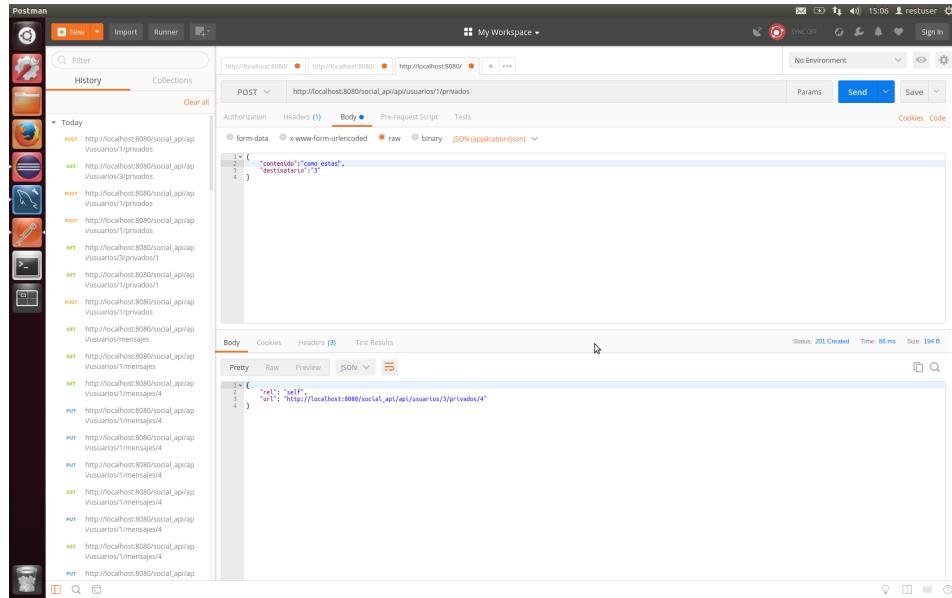


Figura 19: Creación del cuarto mensaje privado del usuario 1 al 3.

2.3.2. GET

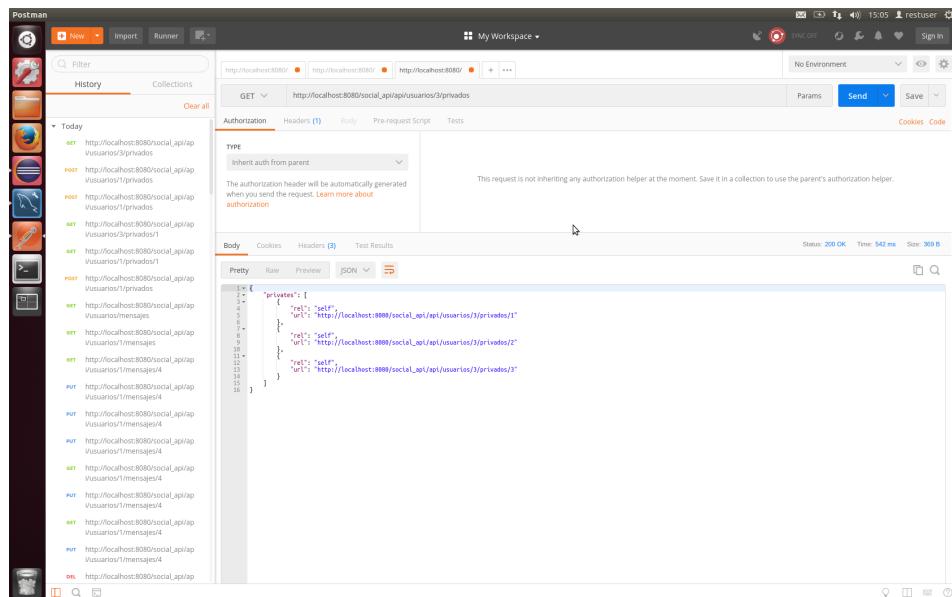


Figura 20: Obtención de todos los mensajes privados escritos por el usuario 3.

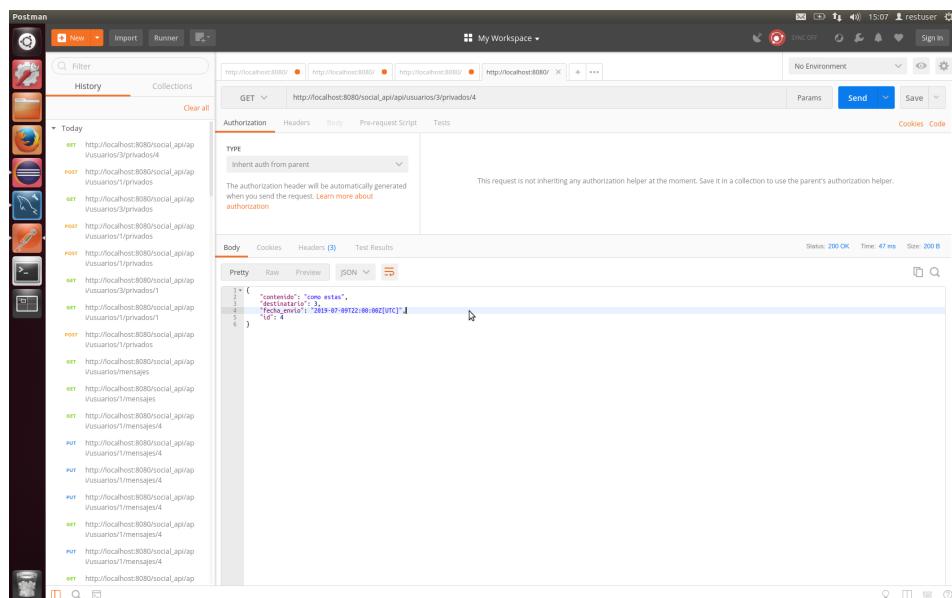


Figura 21: Obtención del mensaje privado 4 del usuario 3.

2.4. Métodos de Amistad

2.4.1. POST

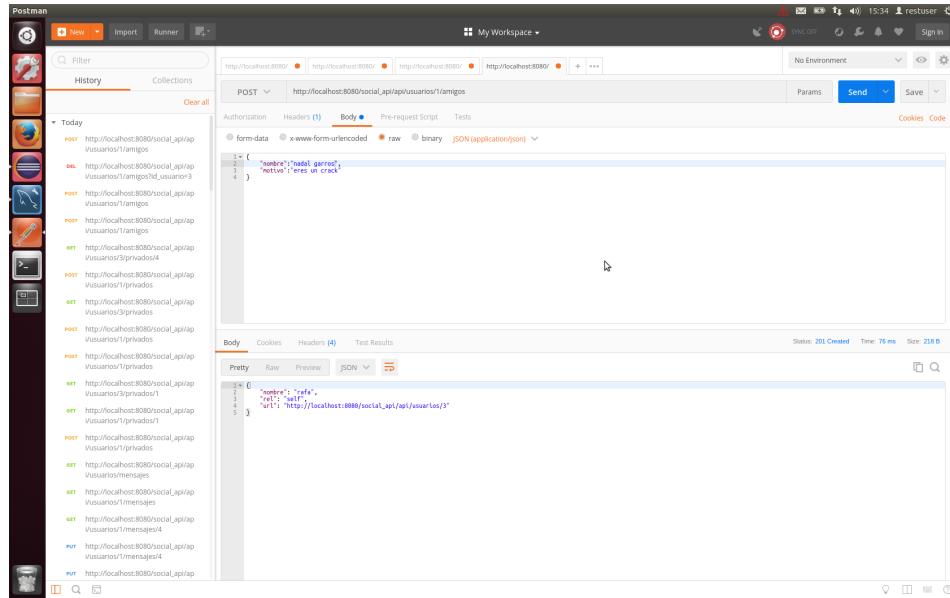


Figura 22: Se añade el usuario 3 como amigo del usuario 1.

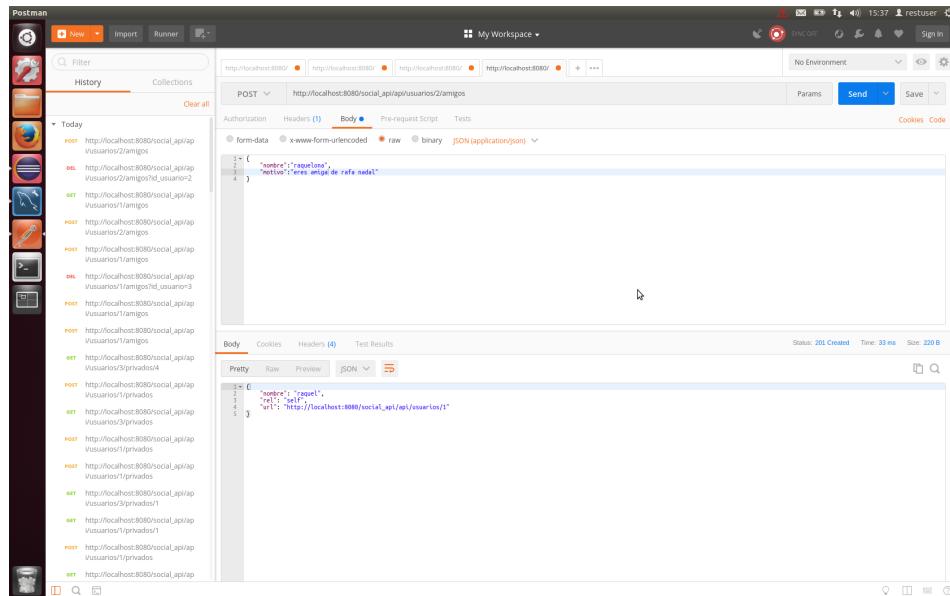


Figura 23: Se añade el usuario 1 como amigo del usuario 2.

2.4.2. GET

The screenshot shows the Postman application interface. In the top navigation bar, 'My Workspace' is selected. The main area displays a GET request to `http://localhost:8080/social_api/api/usuarios/1/amigos`. The 'Authorization' tab is open, showing 'Inherit auth from parent'. The 'Body' tab shows a JSON response with two users:

```

{
  "users": [
    {
      "nombre": "Rafa",
      "url": "http://localhost:8080/social_api/api/usuarios/7"
    },
    {
      "nombre": "Javi",
      "url": "http://localhost:8080/social_api/api/usuarios/2"
    }
  ]
}

```

The status bar at the bottom indicates 'Status: 200 OK'.

Figura 24: Obtención de todos los usuarios amigos del usuario 1.

The screenshot shows the Postman application interface. In the top navigation bar, 'My Workspace' is selected. The main area displays a GET request to `http://localhost:8080/social_api/api/usuarios/3/amigos`. The 'Authorization' tab is open, showing 'Inherit auth from parent'. The 'Body' tab shows a JSON response with three users:

```

{
  "users": [
    {
      "nombre": "Axel",
      "url": "http://localhost:8080/social_api/api/usuarios/7"
    },
    {
      "nombre": "Ivan",
      "url": "http://localhost:8080/social_api/api/usuarios/10"
    },
    {
      "nombre": "Paco",
      "url": "http://localhost:8080/social_api/api/usuarios/3"
    }
  ]
}

```

The status bar at the bottom indicates 'Status: 200 OK'.

Figura 25: Obtención de todos los usuarios amigos del usuario 3.

The screenshot shows the Postman interface with a list of requests in the history on the left. A specific GET request to `http://localhost:8080/social_api/api/usuarios/1/amigos` is selected. The response status is 200 OK, time 26 ms, size 206 B. The JSON response body is displayed in the preview tab:

```

{
  "users": [
    {
      "nombre": "rafaf",
      "rel": "self",
      "url": "http://localhost:8080/social_api/api/usuarios/1"
    }
  ]
}

```

Figura 26: Obtención de los amigos del usuario 1 después de eliminar uno de ellos (Figura 27).

2.4.3. DELETE

The screenshot shows the Postman interface with a list of requests in the history on the left. A specific DELETE request to `http://localhost:8080/social_api/api/usuarios/1/amigos?id_usuario=2` is selected. The response status is 200 OK, time 56 ms, size 160 B. The JSON response body is displayed in the preview tab:

```

{
  "users": [
    {
      "nombre": "Removed friend",
      "rel": "self"
    }
  ]
}

```

Figura 27: Eliminación del usuario 2 de los amigos del usuario 1.

2.5. Obtención de mensajes similar a FaceBook

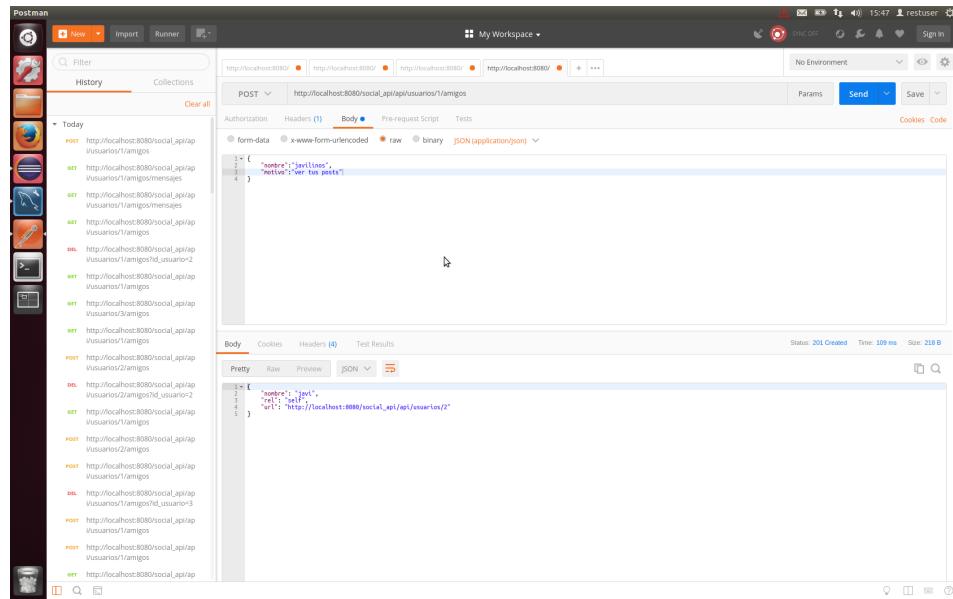


Figura 28: Obtención de los amigos del usuario 1.

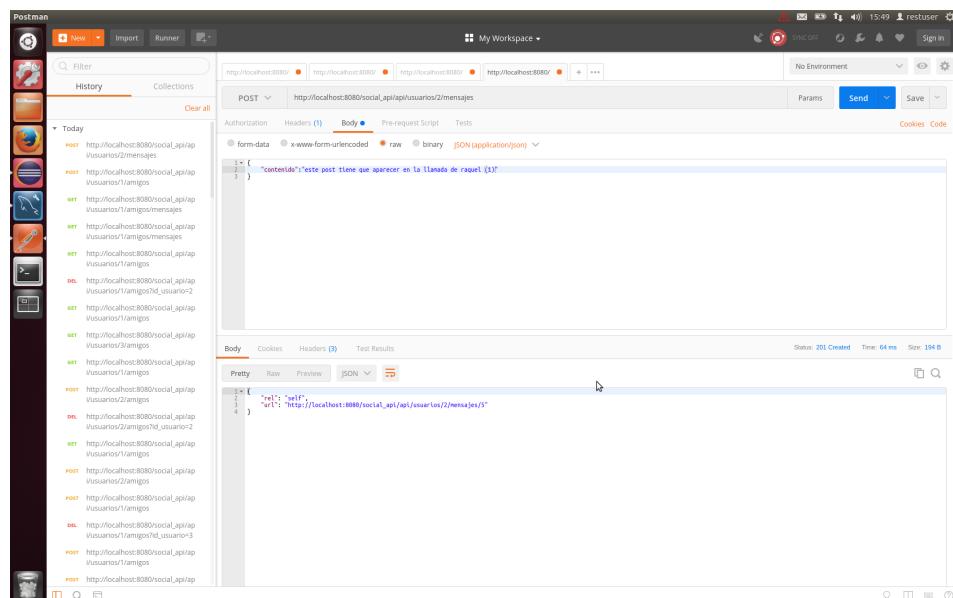


Figura 29: Creación de un mensaje del usuario 2.

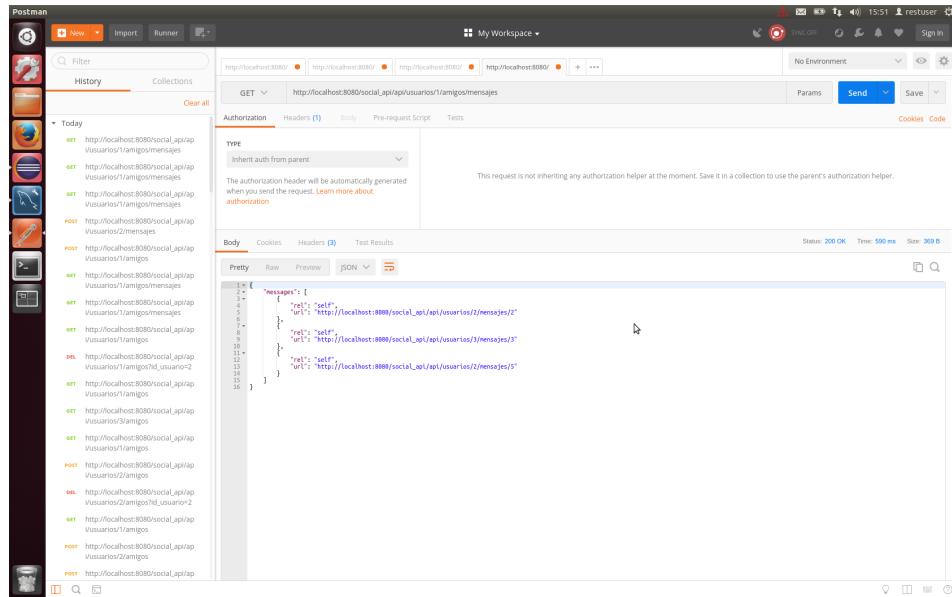


Figura 30: Obtención de los mensajes de los amigos del usuario 1.

2.6. Aplicación móvil

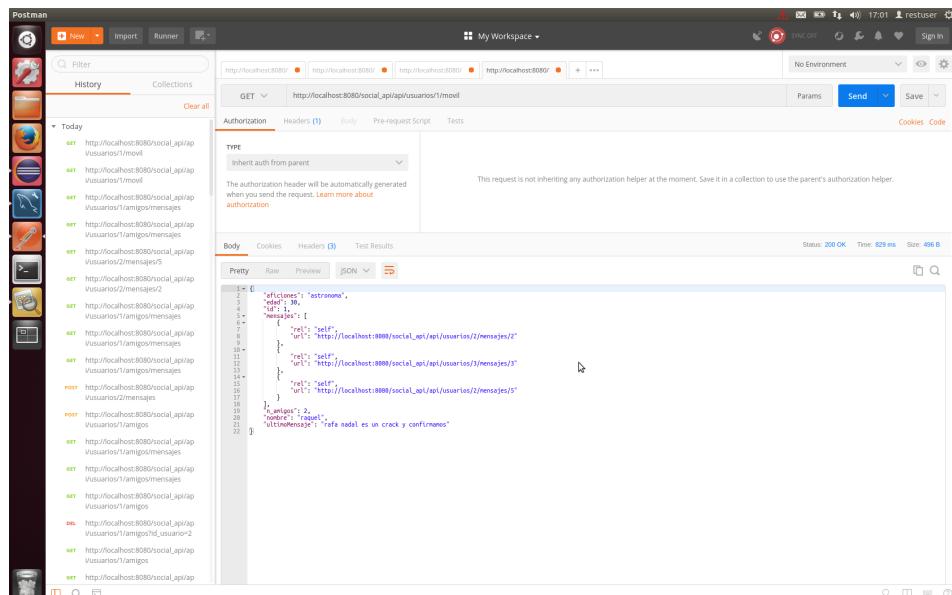


Figura 31: Obtención de la información del usuario 1 simulando una aplicación móvil.

3. ANEXO: Problemas encontrados

Los fallos encontrados referentes a la Máquina virtual son:

- Con la aplicación Postman, al probar los métodos HTTP.
- Con la aplicación Eclipse, que algunas veces no encontraba las URIs implementadas.
- Con la baja calidad de las capturas de pantalla del SO utilizado (probablemente por la antigüedad del mismo).

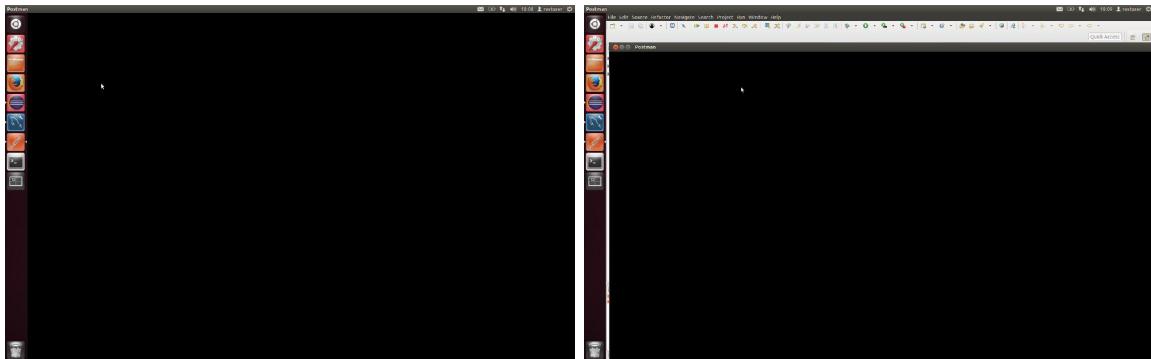


Figura 32: Fallo de la aplicación Postman al probar métodos.