Index:

Note:
THE ASSIGNMENT WAS DONE DURING MY COMPUTER ENGINEERING BACHELOR DEGREE AT THE UNIVERSIDAD POLITECNICA DE MADRID, SPAIN

1. Description of the web Service API

En el servicio proporcionado los usuarios, además de realizar otras acciones, podrán publicar mensajes en su página personal y podrán ser amigos de otros usuarios (relación de amistad recíproca) para poder ver los mensajes de éstos, a los que además podrán enviar mensajes privados (que no puede ver nadie más que los implicados en el envío, emisor y receptor).

El servicio debe soportar a través de esa API las siguientes operaciones:

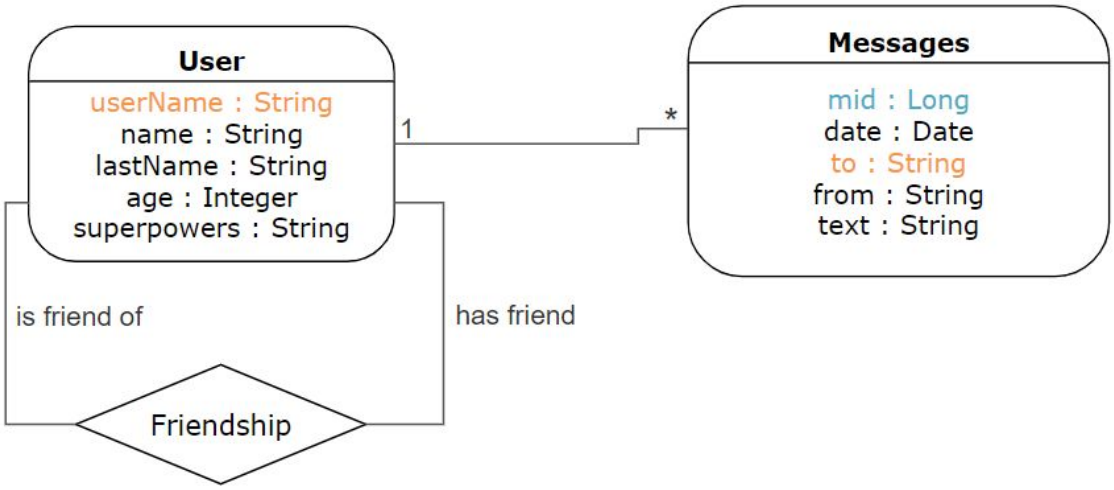
- Añadir un nuevo usuario a la red.
- Ver los datos básicos de un usuario.
- Cambiar datos básicos de nuestro perfil de usuario (excepto nombre de usuario).
- Obtener una lista de todos los usuarios existentes en la red social. Esta lista debe permitir ser filtrada por patrón de nombre (eg. Buscar todos los usuarios que contengan “Mar” en su nombre, “Mario”, “María”...etc.)
- Publicar un nuevo mensaje en la página personal de un usuario.
- Eliminar un mensaje propio.
- Editar un mensaje propio.
- Obtener una lista de todos los mensajes de un usuario en su página personal. Además, esta lista debe permitir la opción de ser filtrada por fecha o limitar la cantidad de información obtenida por número de mensajes (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)
- Añadir un nuevo amigo
- Eliminar un amigo
- Obtener una lista de todos nuestros amigos. Además, esta lista debe permitir la opción de ser filtrada por el patrón de nombre o limitar la cantidad de información obtenida por número de amigos (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)
- Enviar un mensaje personal a otro usuario
- Borrar nuestro perfil de la red social
- Obtener una lista con los últimos mensajes de las páginas de nuestros amigos ordenados por fecha (similar a como lo muestra Facebook). Esta lista debe permitir la opción de ser filtrada por la búsqueda de contenido de texto (patrón) en el mensaje.
- Consultar fácilmente la descripción necesaria para una aplicación móvil que queremos realizar, que muestre los datos básicos de un usuario, su último mensaje en la página, el número de amigos y los 10 últimos mensajes de las páginas de sus amigos que se han actualizado.

2. Identification of all the resources for the service

Basándonos en la descripción dada en el enunciado de la práctica podemos deducir que los recursos a los que se les va a aplicar las operaciones descritos en ella son:

a. User (Usuario)
b. Friend (Amigo)
c. Message (Mensaje)

Por tanto, el diagrama/modelo Entidad Relación (EER) quedaría así:



En el se puede observar que tenemos dos entidades User y Messages a través de las cuales un usuario puede enviar varios mensajes pero un mensaje sólo puede pertenecer a un usuario. Friend no lo hemos considerado como una entidad sino como una relación que tienen los Users. Todas las entidades descritas están formadas por una serie de atributos que las definen.

A continuación se encuentran los diseños de los nuevos tipos de documentos XML schema necesarios para el servicio. Dado que cada caso es concreto a según qué información se pretende actualizar o crear desde cero, en los cuerpos de las peticiones se encuentran ejemplos de uso:

Añadir un nuevo usuario a la red

URI	http://socialHero.es/webapi/users
Metodo	POST
Cuerpo de la petición	<xs:element name="**user**"> <xs:complexType> <xs:sequence> <xs:element name="**userName**" type="xs:string" /> <xs:element name="**name**" type="xs:string" /> <xs:element name="**lastName**" type="xs:string" /> <xs:element name="**age**" type="xs:positiveInteger" /> <xs:element name="**superPowers**" type="xs:string" /> <xs:simpleType name="**friends**"> <xs:list itemType="xs:user"/> </xs:simpleType> <xs:simpleType name="**personalPage**"> <xs:list itemType="xs:message"/> </xs:simpleType> </xs:sequence> </xs:complexType> </xs:element>
Devuelve	200 = OK; 500 Internal Error

Ver los datos básicos de un usuario

URI	http://socialHero.es/webapi/users/{userName}
Metodo	GET
Cadena de Consulta	userName= CaptainAmerica, BlackWidow, Spiderman, Ironman, ...
Devuelve	200 = OK + POX ó JSON 500 Internal Error

Cambiar datos básicos de nuestro perfil de usuario (excepto nombre de usuario)

URI	http://socialHero.es/webapi/users/{userName}
Metodo	PUT
Cuerpo de la petición	Se puede modificar cualquier atributo menos el nombre de usuario: <xs:element name="**user**"> <xs:complexType> <xs:sequence> <xs:element name="**userName**" type="xs:string" /> <xs:element name="**name**" type="xs:string" /> <xs:element name="**lastName**" type="xs:string" /> <xs:element name="**age**" type="xs:positiveInteger" /> <xs:element name="**superPowers**" type="xs:string" /> <xs:simpleType name="**friends**"> <xs:list itemType="xs:user"/> </xs:simpleType> <xs:simpleType name="**personalPage**"> <xs:list itemType="xs:message"/> </xs:simpleType> </xs:sequence> </xs:complexType> </xs:element>
Devuelve	200 = OK+POX (tipo MIME o application/XML + referencia al XMLSchema, etc.) 500 Internal Error

Obtener lista de todos los usuarios existentes y debe permitir ser filtrada por patrón de nombre

URI	http://socialHero.es/webapi/users, http://socialHero.es/webapi/users?str=<string>
Metodo	GET
Cadena de Consulta	str= String //userName= CaptainAmerica, BlackWidow, Spiderman, Ironman, ...
Devuelve	200 = OK + POX ó JSON 500 Internal Error

Publicar un nuevo mensaje en la página personal de un usuario

URI	http://socialHero.es/webapi/users/{userName}/messages
Metodo	POST
Cuerpo de la petición	<xs:element name="**message**"> <xs:complexType> <xs:attribute type="xs:string" name="**date**"/> <xs:sequence> <xs:element type="xs:string" name="**from**"/> <xs:element name="**body**"> <xs:complexType> <xs:sequence> <xs:element type="xs:string" name="**text**"/> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:schema>
Devuelve	200 = OK , 500 Internal Error

Eliminar un mensaje propio

URI	http://socialHero.es/webapi/users/{userName}/messages/{mid}
Metodo	DELETE
Devuelve	200 = OK, 500 Internal Error

Editar un mensaje propio

URI	http://socialHero.es/webapi/users/{userName}/messages/{mid}
Metodo	PUT
Cuerpo de la petición	<xs:element name="**message**"> <xs:complexType> <xs:attribute type="xs:string" name="**date**"/> <xs:sequence> <xs:element type="xs:string" name="**from**"/> <xs:element name="**body**"> <xs:complexType> <xs:sequence> <xs:element type="xs:string" name="**text**"/> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:schema>
Devuelve	200 = OK, 500 Internal Error

Obtener lista de todos los mensajes de un usuario en su página personal. Además, esta lista debe permitir la opción de ser filtrada por fecha o limitar la cantidad de información obtenida por número de mensajes

URI	http://socialHero.es/webapi/users/{userName}/messages, http://socialHero.es/webapi/users/{userName}/messages?start=<int>, size=<int>
Metodo	GET
Cadena de Consulta	start= int, size= int
Devuelve	200 = OK + POX ó JSON 500 Internal Error

Añadir un nuevo amigo (usuario añadir a usuario)

URI	http://socialHero.es/webapi/users/{userName}/friends?userName=USERNAME
Metodo	POST
Cuerpo de la petición	<xs:element name="**user**"> <xs:complexType> <xs:sequence> <xs:element name="**userName**" type="xs:string" /> <xs:element name="**name**" type="xs:string" /> <xs:element name="**lastName**" type="xs:string" /> <xs:element name="**age**" type="xs:positiveInteger" /> <xs:element name="**superPowers**" type="xs:string" /> <xs:simpleType name="**friends**"> <xs:list itemType="xs:user"/> </xs:simpleType> <xs:simpleType name="**personalPage**"> <xs:list itemType="xs:message"/> </xs:simpleType> </xs:sequence> </xs:complexType> </xs:element>
Devuelve	200 = OK, 500 Internal Error

Eliminar un amigo

URI	http://socialHero.es/webapi/users/{userName}/friends/{userName}
Metodo	DELETE
Devuelve	200 = OK, 500 Internal Error

Obtener una lista de todos nuestros amigos. Además, esta lista debe permitir la opción de ser filtrada por nombre o limitar la cantidad de información obtenida por número de amigos

URI	http://socialHero.es/webapi/users/{userName}/friends, http://socialHero.es/webapi/users/{userName}/friends?str=<String>&start=<Int>&size=<Int>
Metodo	GET
Cadena de Consulta	str= userName, start= int, size= int
Devuelve	200 = OK + POX ó JSON 500 Internal Error

Enviar un mensaje personal a otro usuario

URI	http://socialHero.es/webapi/users/{userName}/messages
Metodo	POST
Cuerpo de la petición	<xs:element name="**message**"> <xs:complexType> <xs:attribute type="xs:string" name="**date**"/> <xs:sequence> <xs:element type="xs:string" name="**from**"/> <xs:element name="**body**"> <xs:complexType> <xs:sequence> <xs:element type="xs:string" name="**text**"/> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:schema>
Devuelve	200 = OK, 500 Internal Error

Borrar nuestro perfil de la red social

URI	http://socialHero.es/webapi/users/{userName}
Metodo	DELETE
Devuelve	200 = OK, 500 Internal Error

Obtener lista con últimos mensajes de las páginas de nuestros amigos ordenados por fecha Esta lista debe permitir la opción de ser filtrada por la búsqueda de contenido de texto en el mensaje

URI	http://socialHero.es/webapi/users/{userName}/friends/{friendUserName}/messages, http://socialHero.es/webapi/users/{userName}/friends/{friendUserName}/messages?str=<String>
Metodo	GET
Cadena de Consulta	str=<String>
Devuelve	200 = OK + POX ó JSON 500 Internal Error

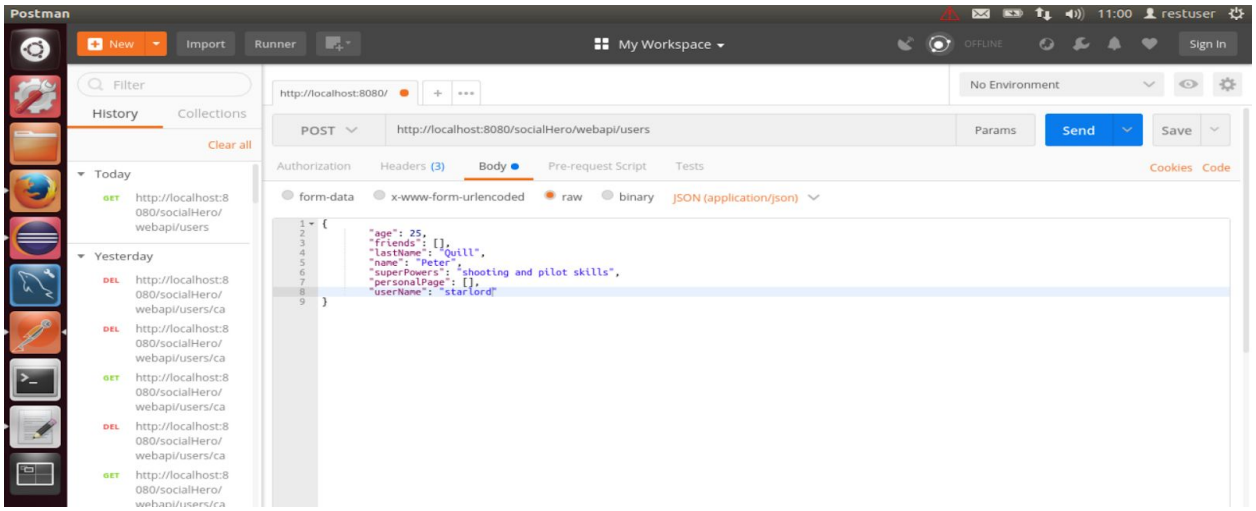
Consultar fácilmente la descripción necesaria para una aplicación móvil que queremos realizar, que muestre los datos básicos de un usuario, su último mensaje en la página, el número de amigos y los 10 últimos mensajes de las páginas de sus amigos que se han actualizado

Completar tabla

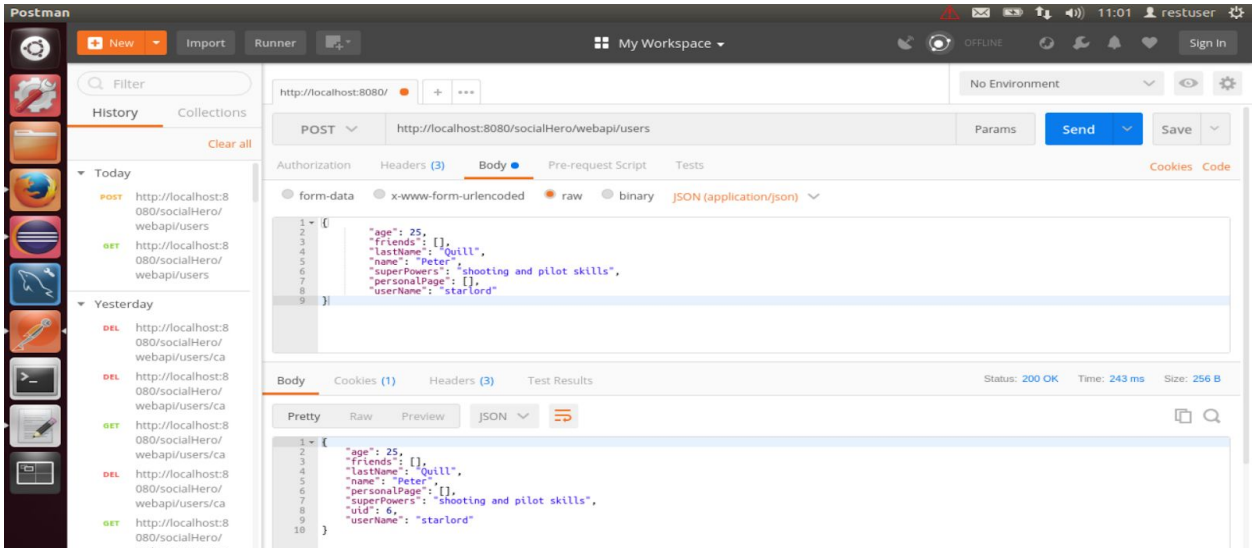
4. Screenshots of the execution of the operations mentioned previously from a REST client (Postman)

Añadir un nuevo usuario a la red

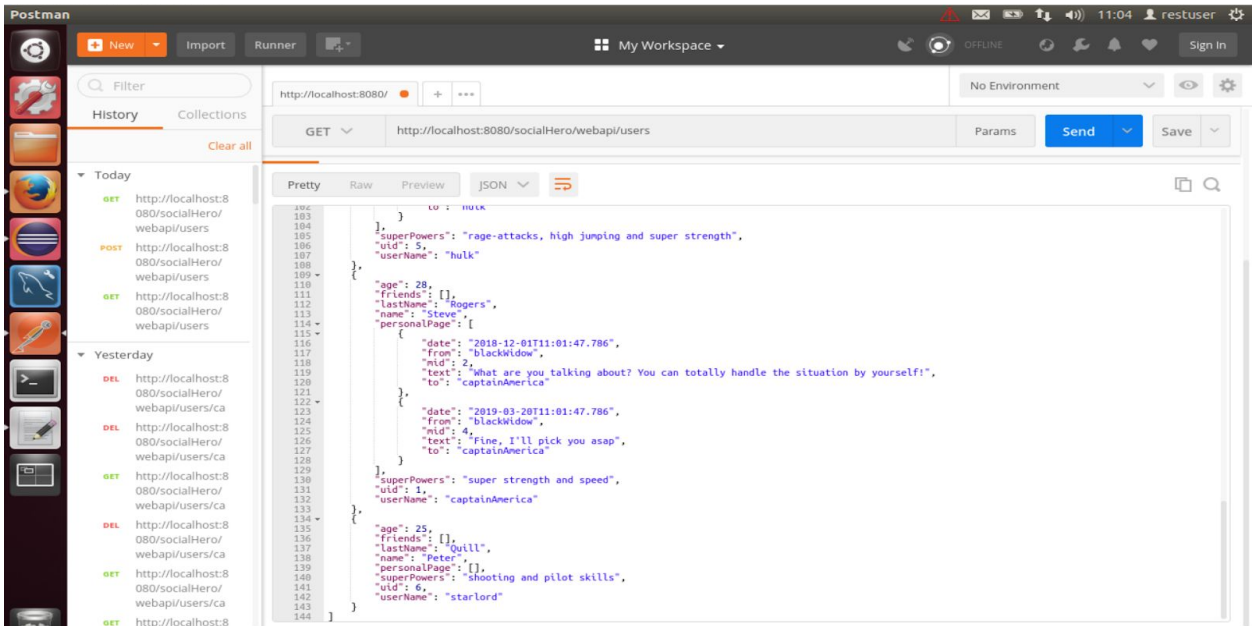
Cuerpo de la petición: en esta petición de HTTP queremos añadir el usuario con *userName* “starlord” a la red social.



Respuesta peticion POST: 200 OK + datos basicos de nuevo usuario creado

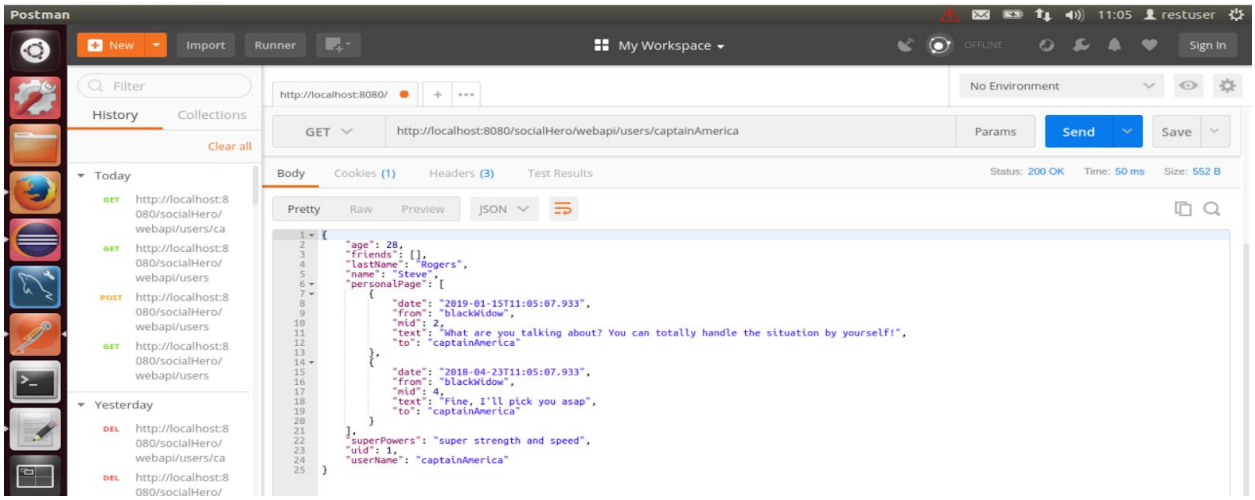


Resultado petición GET de usuarios de la red para la comprobación del usuario añadido:



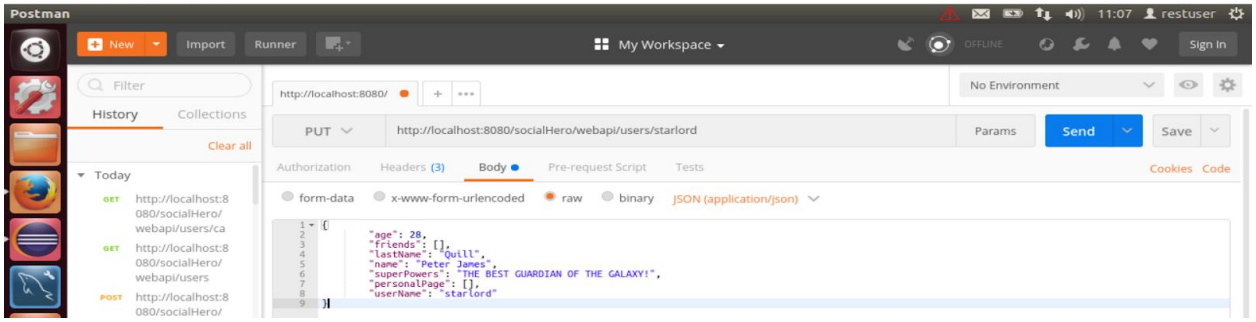
Ver datos básicos de usuario

Cuerpo de la petición GET: en esta petición de HTTP queremos obtener la información del usuario con *userName* "captainAmerica": 200 OK + datos basicos de usuario captainAmerica

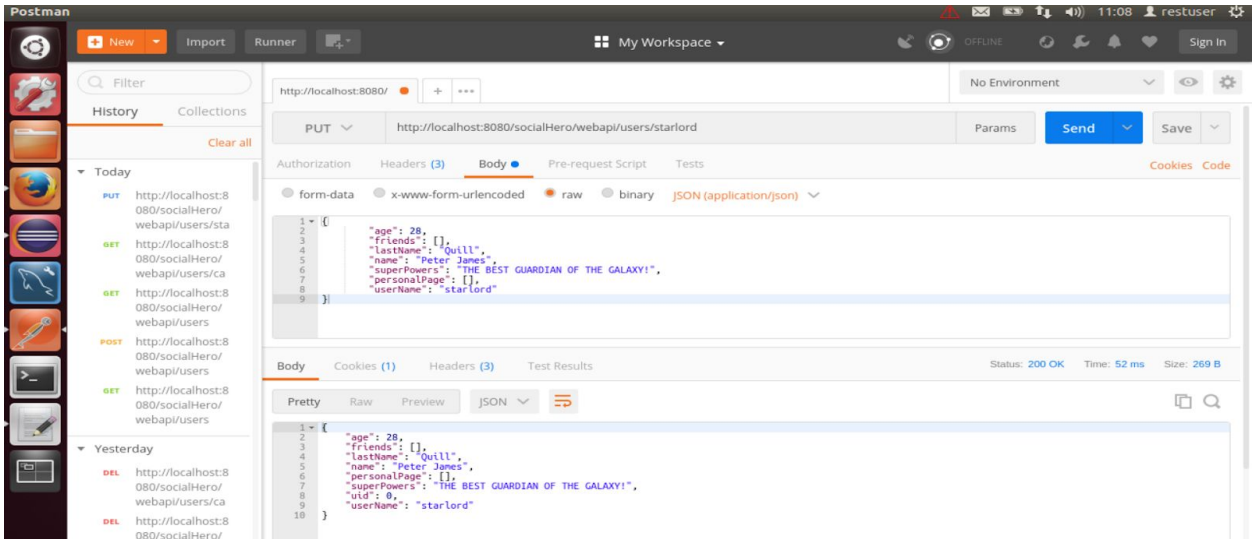


Cambiar datos básicos de perfil de usuario

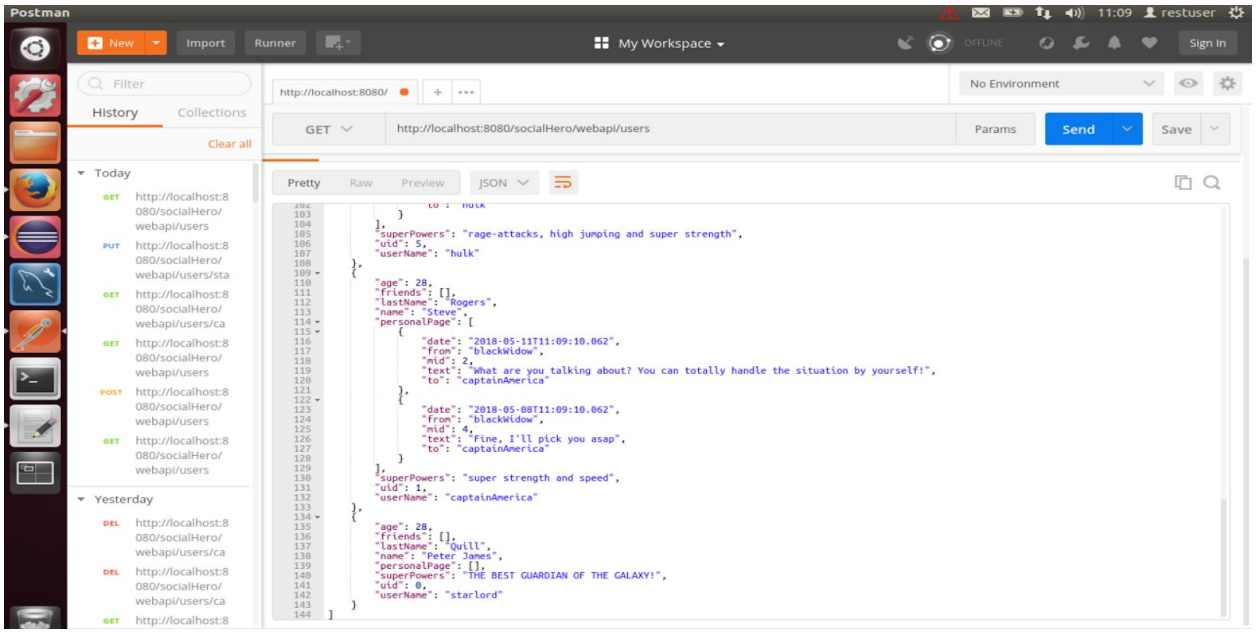
Cuerpo de la petición PUT: en esta petición de HTTP queremos modificar la edad, nombre y superpoderes del usuario con *userName* "starlord, a continuación se muestra el cuerpo con las modificaciones pendientes:



Resultado: 200 OK + usuario modificado

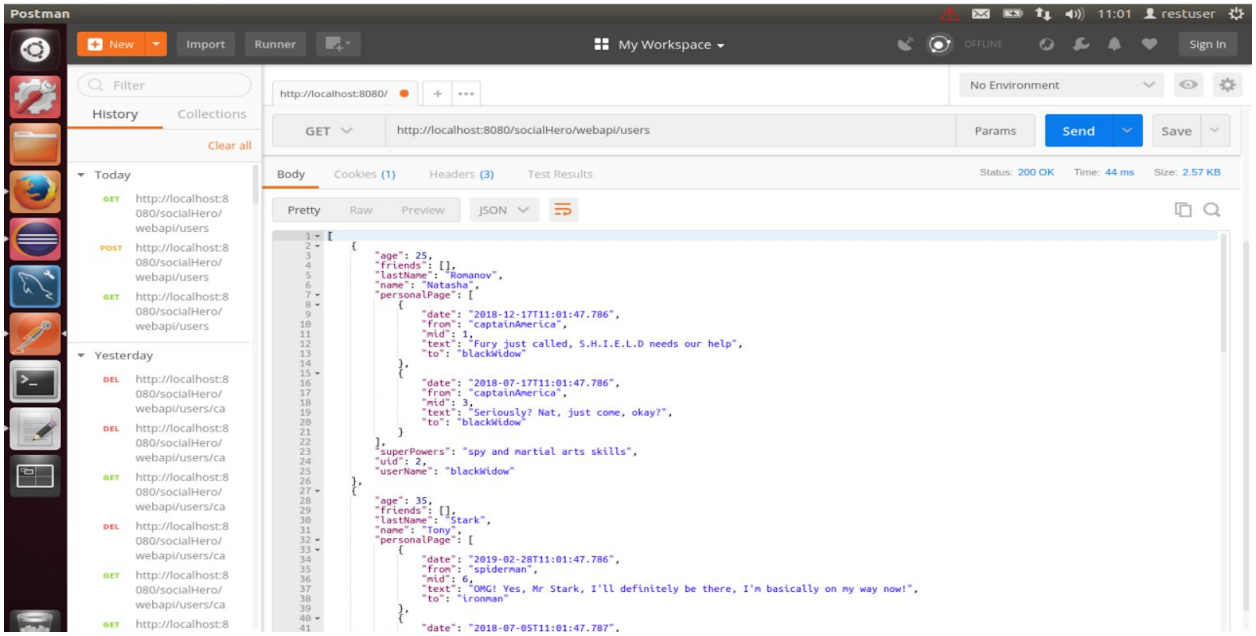


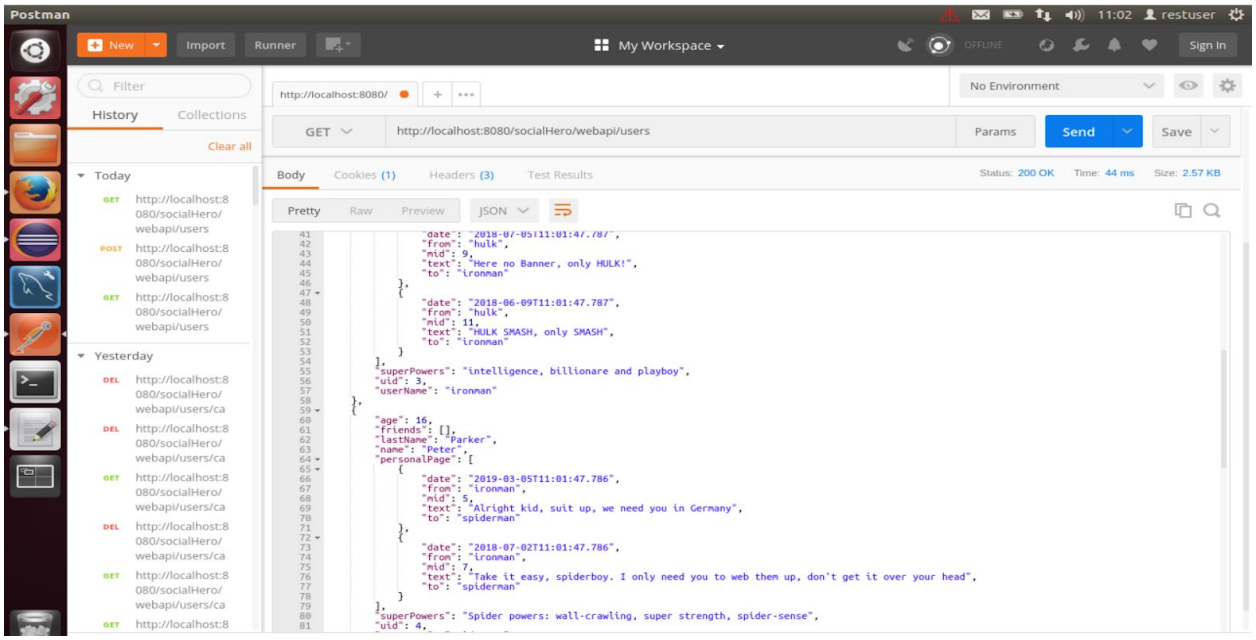
Aqui se muestra que las modificaciones han sido permanentes:

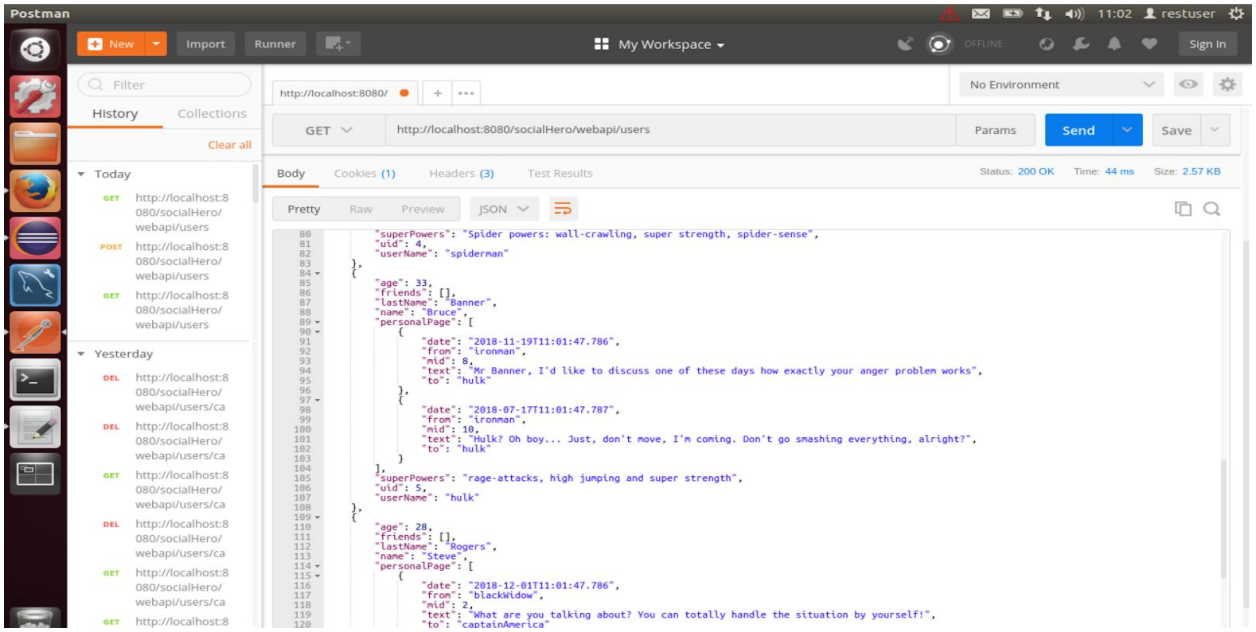


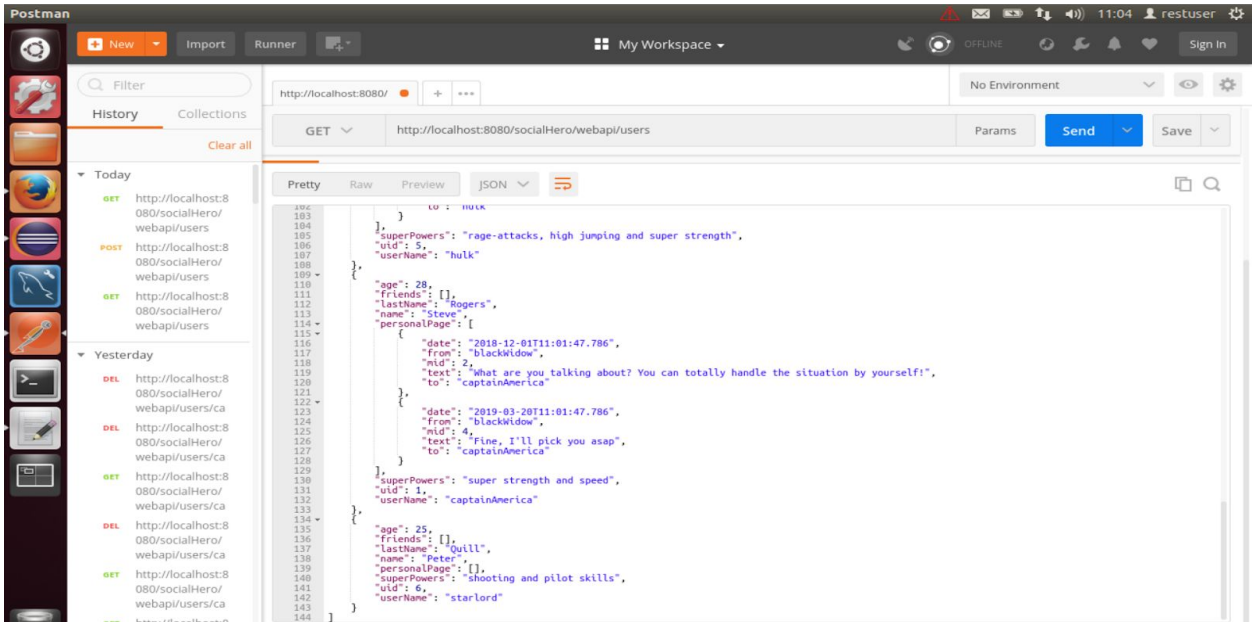
Obtener lista usuarios de la red. Ésta debe ser filtrada por patrón de nombre

Cuerpo de la petición GET: queremos obtener la lista de usuarios de la red (en este caso, el diseño de la devolución de la lista de usuarios incluye sus datos de perfil y su "página personal" con los mensajes que le han dejado otros usuarios): **200 OK + lista usuarios de red**



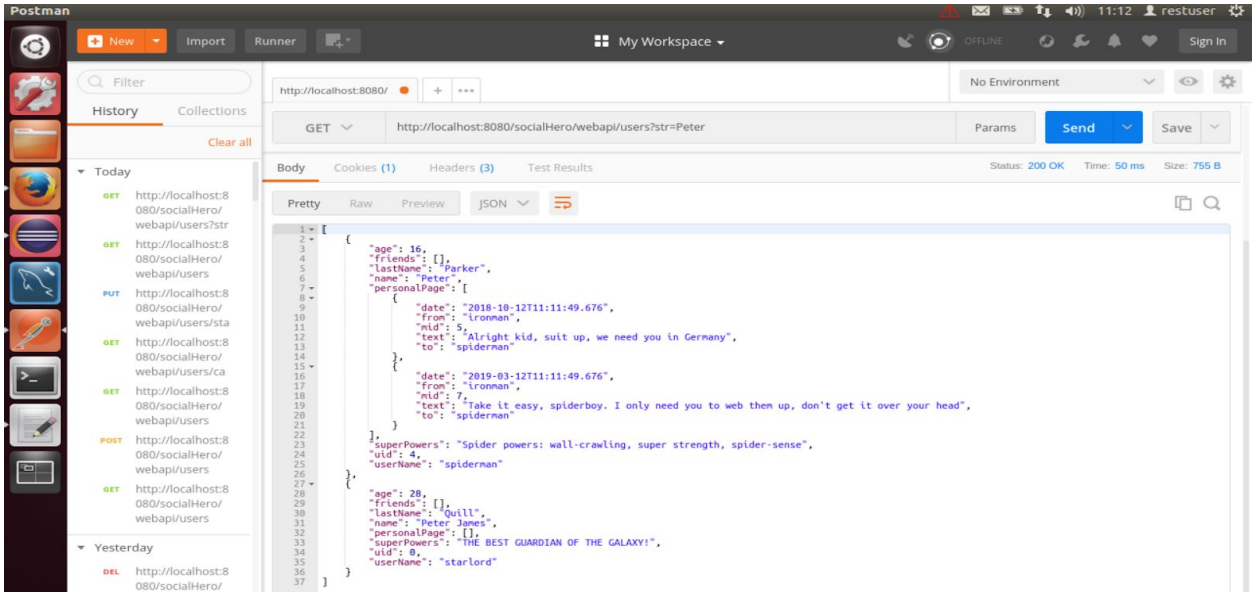






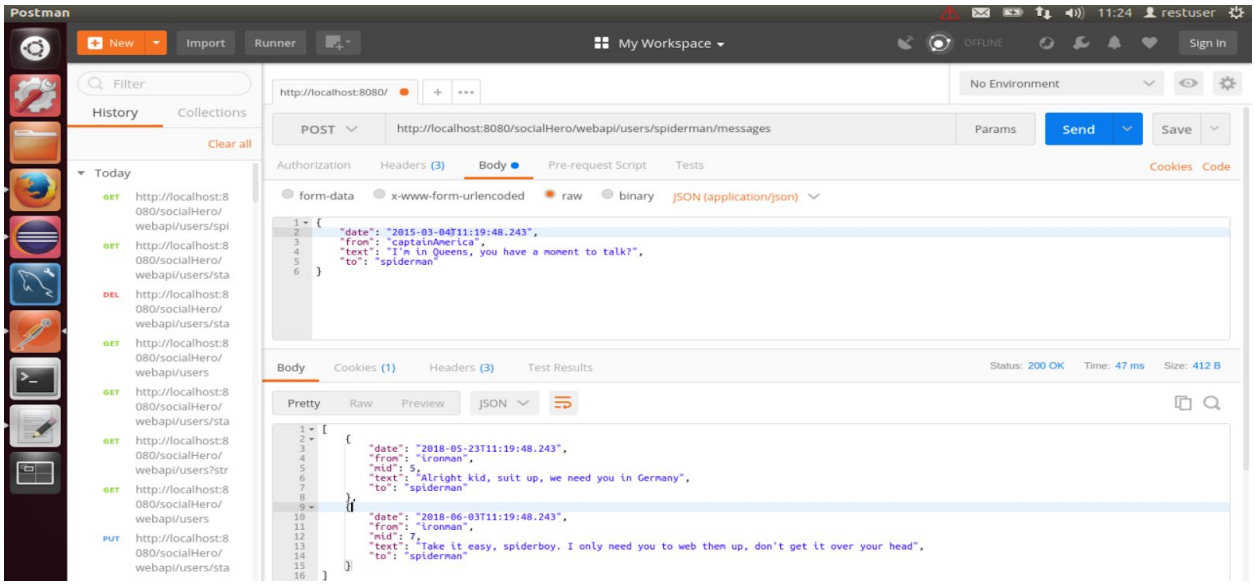
CASO DE FILTRADO DE PATRÓN POR NOMBRE

Cuerpo de la petición: en esta petición de HTTP queremos obtener la lista de usuarios que contengan el patrón de nombre filtrado “Peter”:

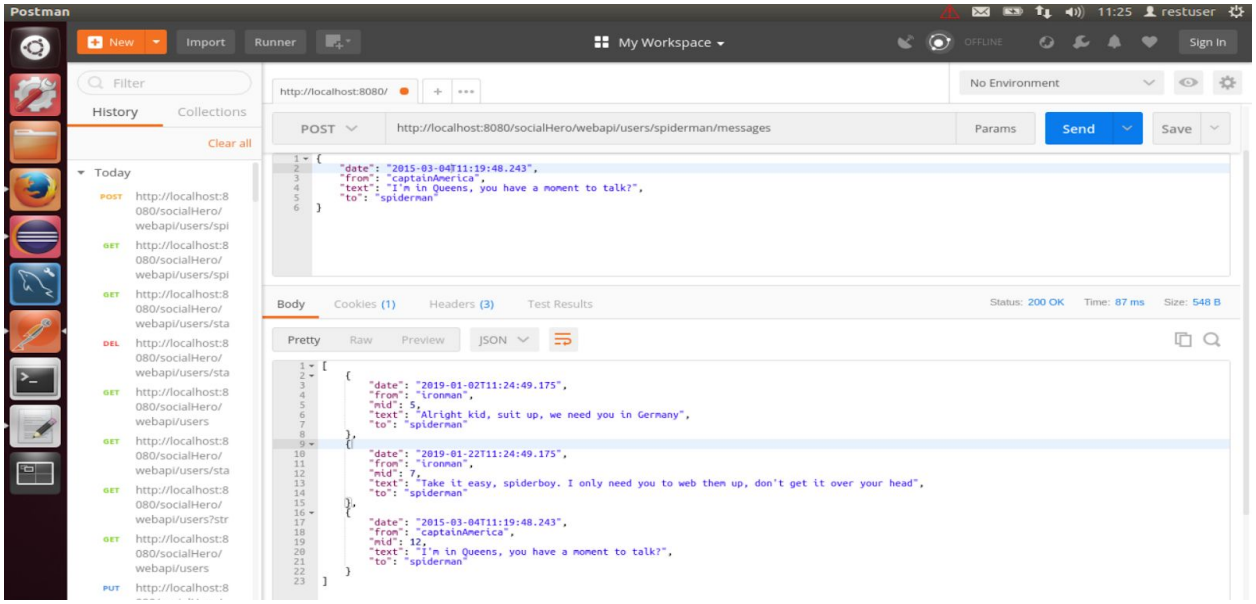


Publicar un nuevo mensaje en la página personal de un usuario

Cuerpo de la petición POST: en esta petición de HTTP queremos crear un mensaje (Body)en la página personal de usuario “spiderman” (debajo se muestra la pagina personal de usuario antes de la petición)

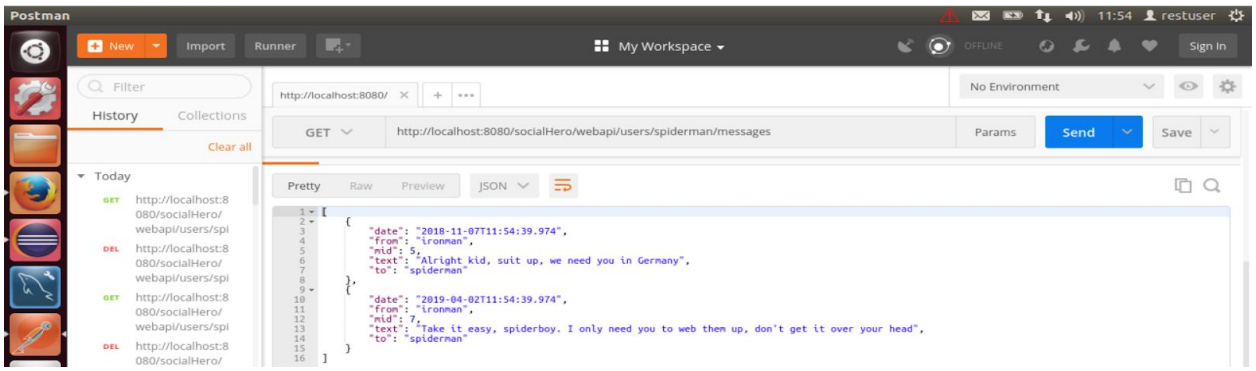


Debajo se muestra la pagina personal de usuario después de la petición (mensaje 12 añadido): 200 OK + lista de mensajes(pagina personal) de usuario:

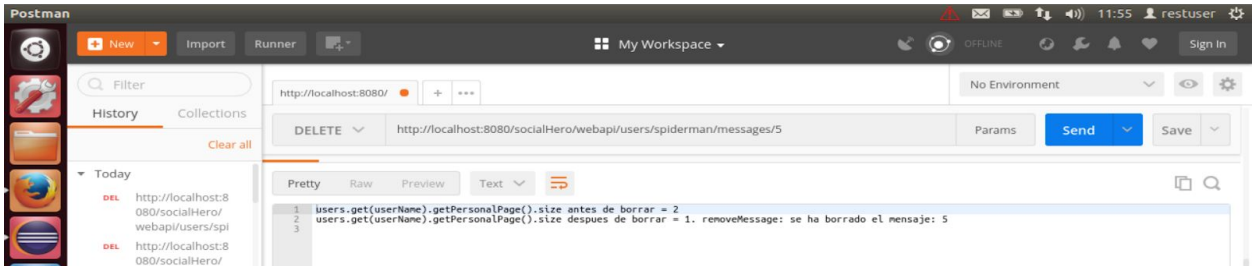


Eliminar un mensaje propio

Cuerpo de la petición DELETE: en esta petición de HTTP queremos borrar un mensaje de la página personal de usuario "spiderman" (debajo se muestra la pagina personal de usuario antes de la petición):

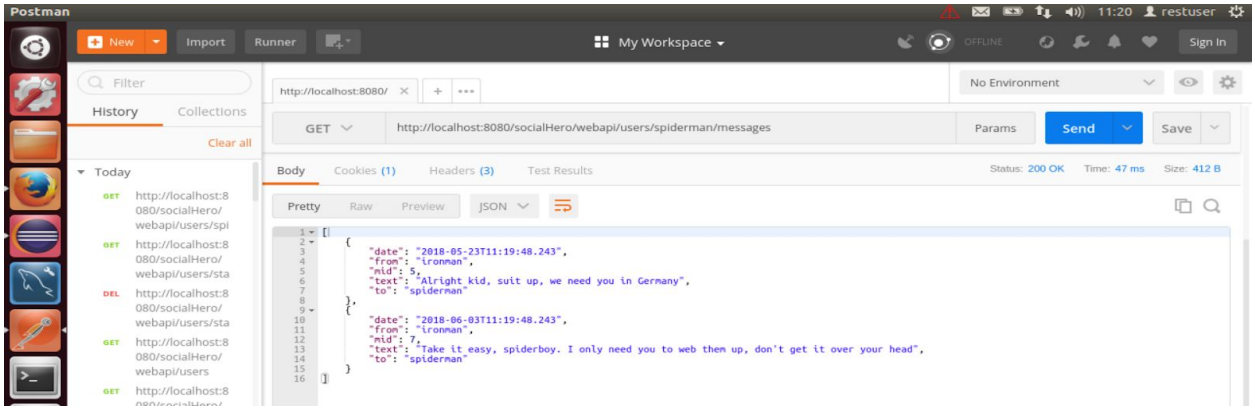


Respuesta petición: Debido a que nos hemos encontrado con problemas a la hora de inicializar o hacer permanentes los cambios en las listas (colecciones usadas como almacenamiento de información/base datos) de mensajes y de amigos, la respuesta se ha devuelto en formato TEXT_PLAIN con printlns que muestran el estado antes y despues de la página personal de usuario demostrando que efectivamente se borra el mensaje (este problema se extrapola al caso de amigos: "Actualizar/Editar Mensaje", "Eliminar amigo" y "Añadir/actualizar amigo")



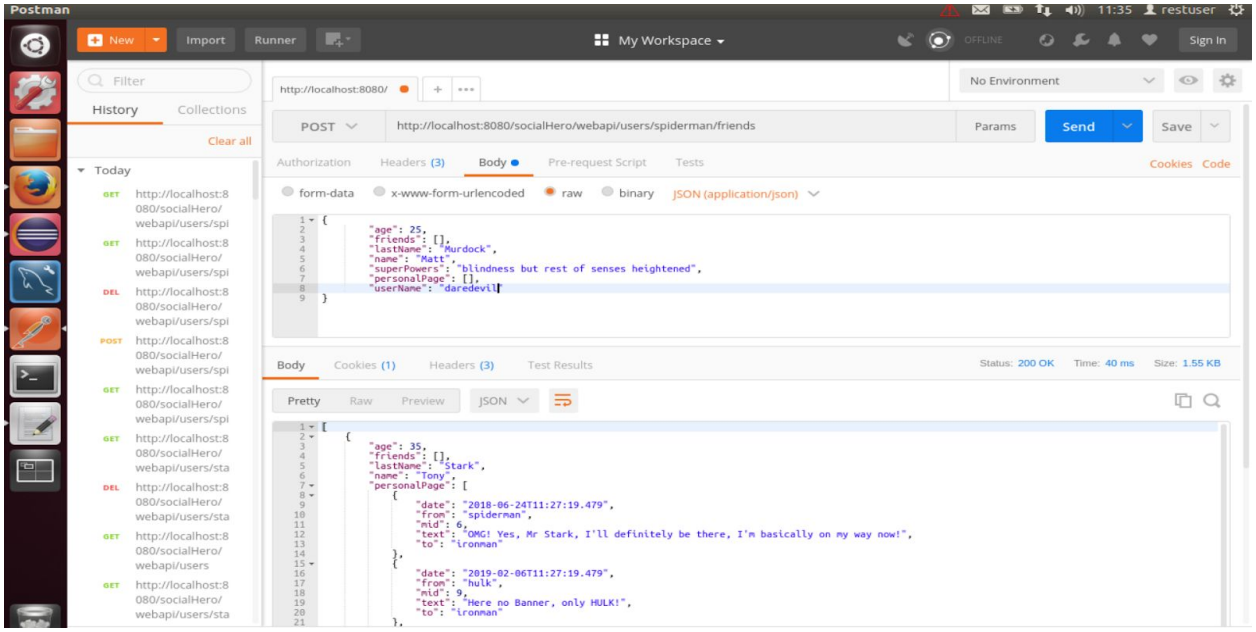
Obtener una lista de todos los mensajes de usuario en su página personal.

Cuerpo de la petición GET: en esta petición de HTTP queremos obtener lista de mensajes del usuario "spiderman" (debajo se muestra la página personal de usuario): 200 OK + lista mensajes

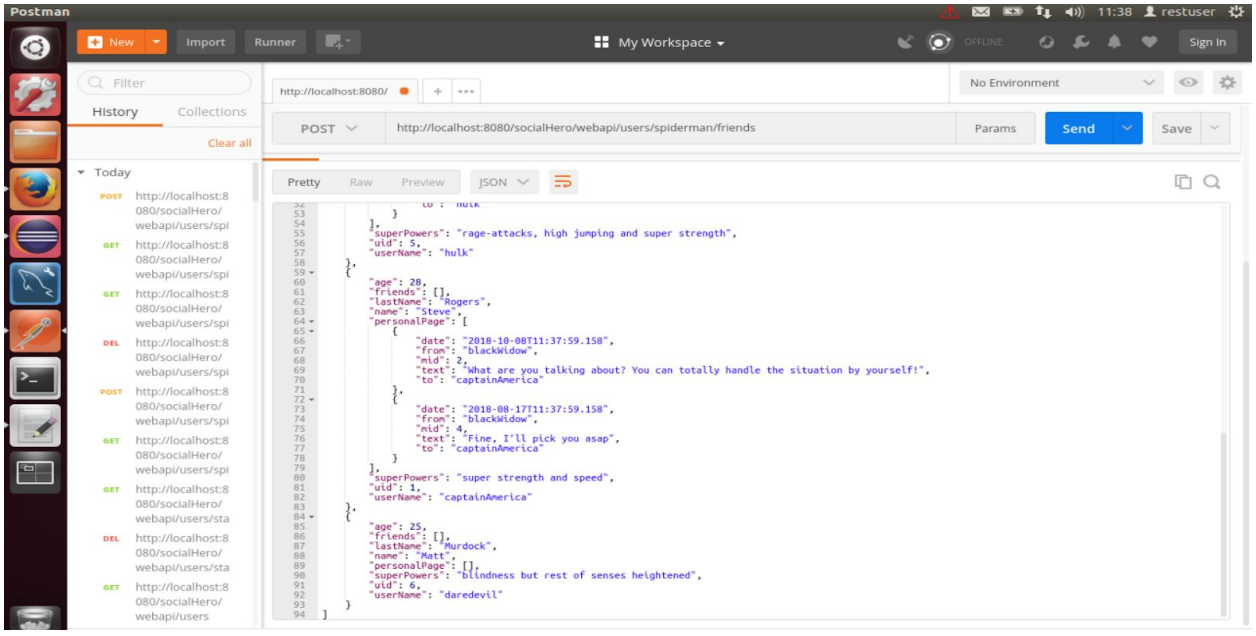


Añadir un amigo a lista de amigos de usuario

Cuerpo de la petición POST: en esta petición de HTTP queremos añadir usuario "daredevil" a lista de amigos del usuario "spiderman" (debajo se muestra la página personal de usuario):

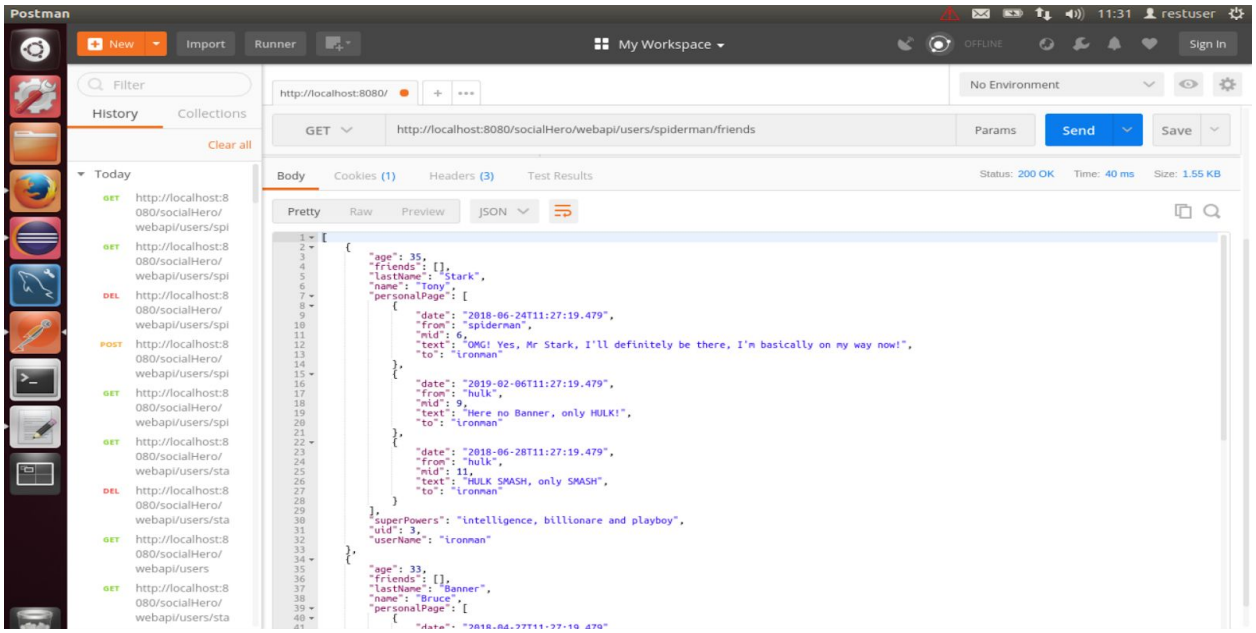


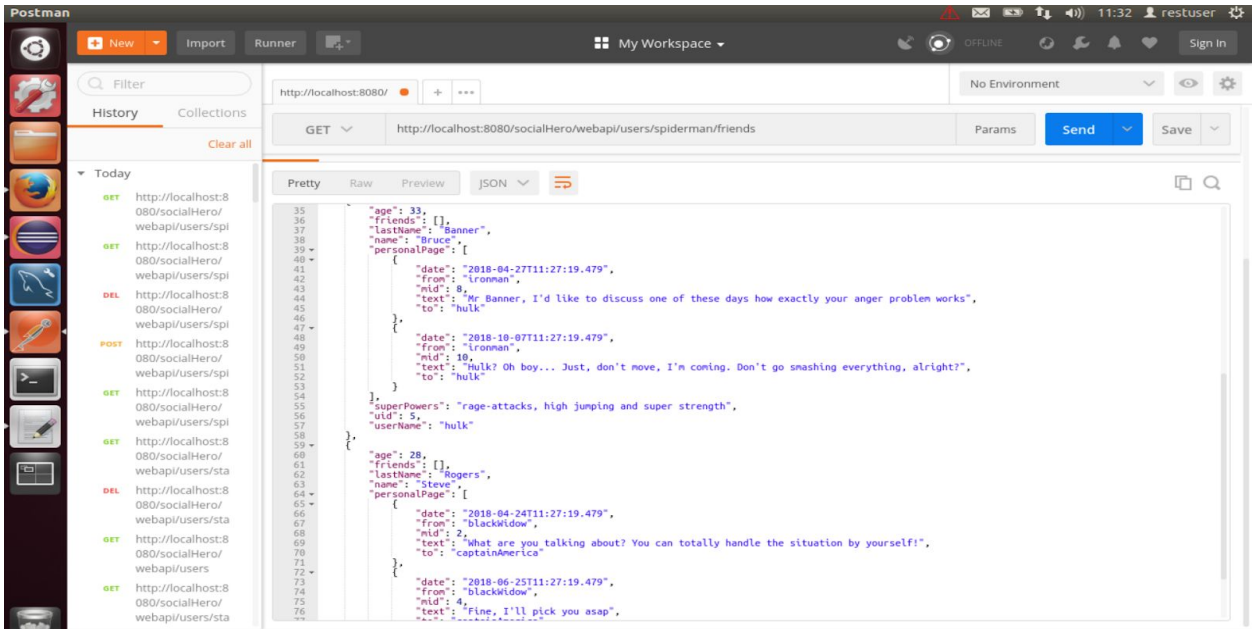
Respuesta petición: se devuelve la lista de amigos del usuario “spiderman” y nos fijamos que ha añadido dicho usuario a la lista. NOTA: como se mencionó más arriba, los cambios no son permanentes en las listas ni de mensajes ni de amigos pues hemos encontrado problemas a la hora de inicializar las listas con datos por defecto para la realización de las pruebas. No obstante el método en sí funciona como se refleja en la lista devuelta por el método



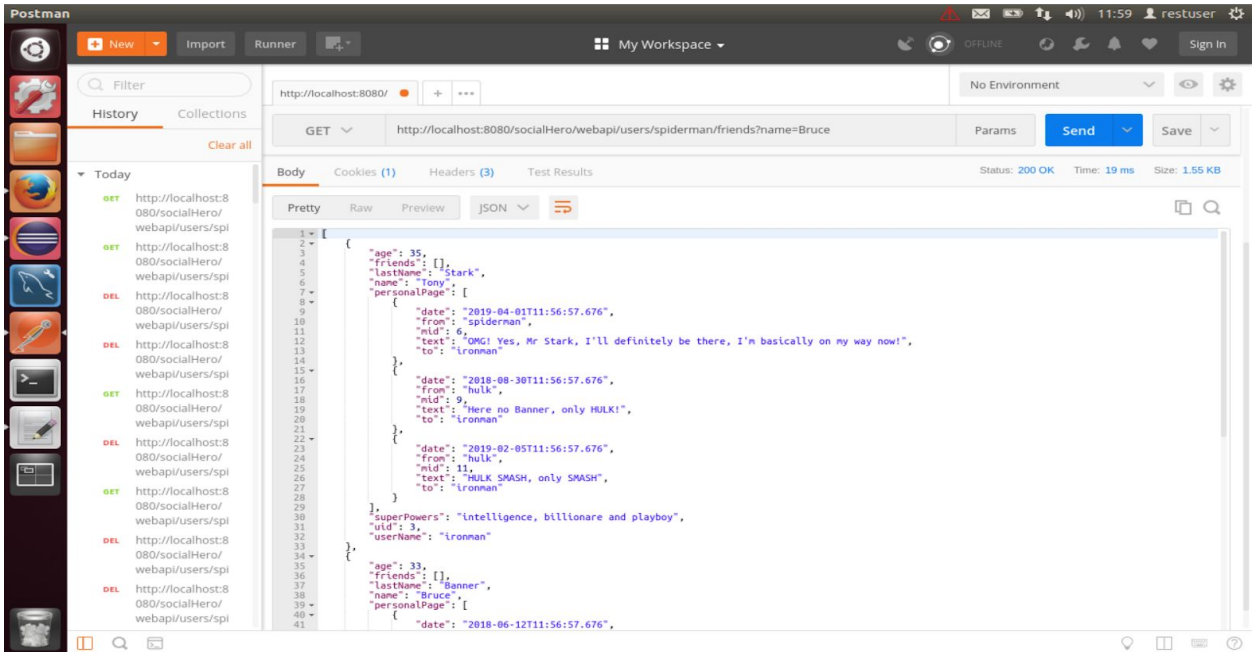
Obtener lista de todos los amigos de usuario. Debe permitirse ser filtrada por patrón de nombre

Cuerpo de la petición GET: en esta petición de HTTP queremos obtener lista de usuarios amigos del usuario “spiderman”: 200 OK + lista de amigos de usuario

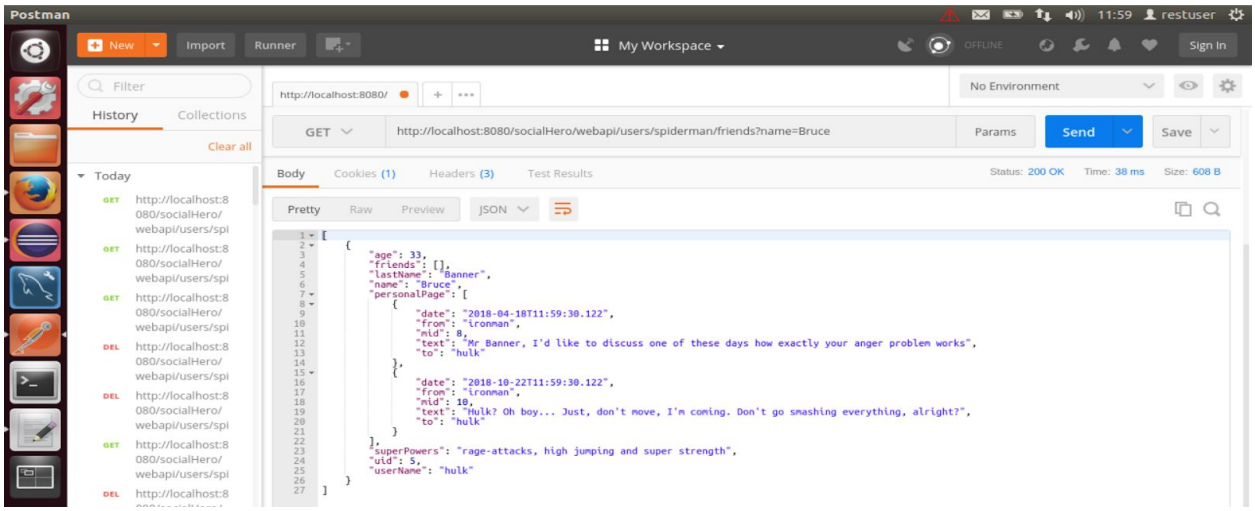




CASO DE SER FILTRADA POR PATRÓN DE NOMBRE:
Antes de la petición GET con **@**queryparam”name” vemos abajo la lista de amigos (parcial) de usuario “spiderman”:

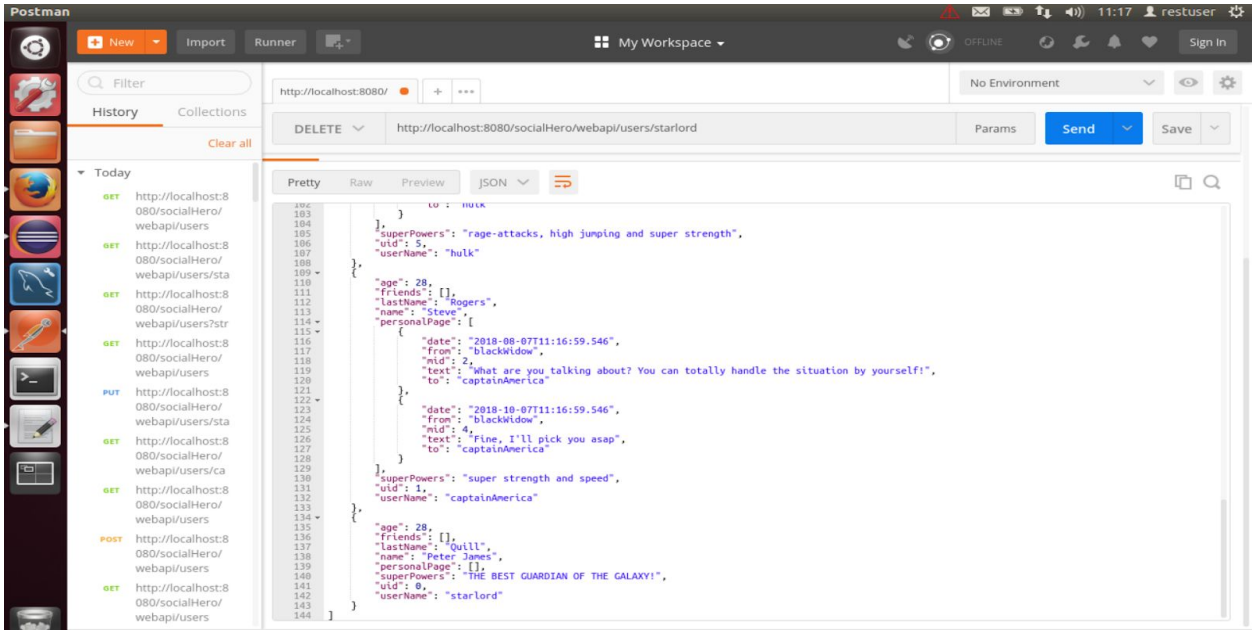


Respuesta de la petición: devuelve filtrado por el nombre “Bruce” los amigos que tengan ese nombre: 200 OK + lista amigos de usuario filtrada

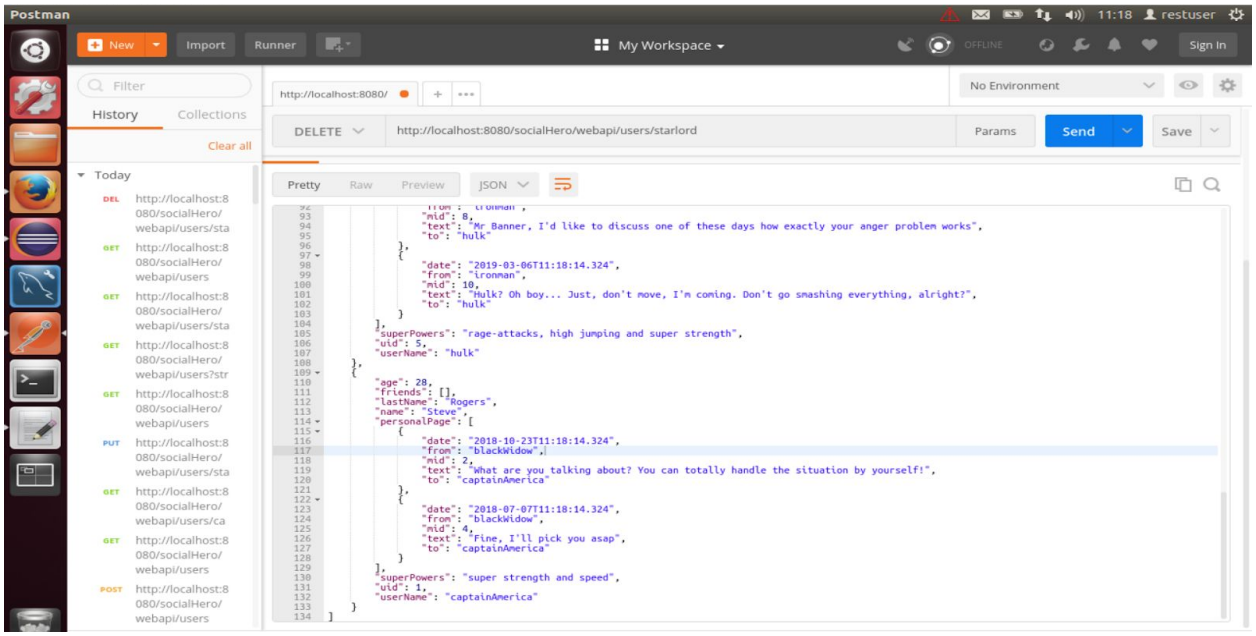


Borrar nuestro perfil de la red social

Cuerpo de la petición DELETE: en esta petición de HTTP queremos borrar el usuario “starlord” de la lista de usuarios de la red:



Respuesta: 200 OK + lista de usuarios de la red social sin usuario “starlord”



Obtener lista de ultimos mensajes de las paginas de usuario

Cuerpo de la petición DELETE: en esta petición de HTTP queremos borrar el usuario “starlord” de la lista de usuarios de la red: