Diseño e implementación de un servicio web RESTful2018

16 ABRIL

Creado por:
Andrés Ollero Morales
Víctor Arzoz Consuegra
Javier Sánchez Tirados

Introducción

Nuestro proyecto se basa en el diseño y la creación de un servicio web con una conexión a una API y a una Base de datos en SQL.
Para el diseño hemos usado las trablas proporcionadas en el enunciado de la práctica. Y para la implementación hemos usado la máquina virtual disponible en moodle.

Nuestro diseño RestFul tiene el nombre de Soscorro que es el que aparece en las URI. Todas las URI comienzan por http://localhost:8080/Soscorro/

Esta memoria se compone de 2 partes:

1. Tablas con operaciones de servicio, recursos de información, colecciones, contenedores, URI, verbos etc
2. Muestreo de la ejecución de la operaciones del cliente REST

1. Diseño RESTFUL

URI	Soscorro/api/users	
Método	GET	
Descripción	Devuelve una lista con las uris de todos los usuarios que cumplen con los del filtro de nombre, si existe	
Cadena de consulta	namePattern =	Patron de nombre para filtrar los usuarios
	offset =	Posición de la lista en la que empezar a devolver la representación del objeto
	count =	Número de elementos a representar en la lista devuelta
Devuelve	200	OK + JSON
	400	Bad Request
	500	Internal Server Error
Método	POST	
Cuerpo de petición	application/json	
Devuelve	201	CREATED + JSON
	500	Internal Server Error

URI	Soscorro/api/users/id	
Método	GET	
Descripción	Devuelve la representacion del usuario seleccionado por su id	
Devuelve	200	OK + JSON
	400	Bad Request
	404	Not Found
	500	Internal Server Error
Método	DELETE	
Descripción	Borra al usuario seleccionado por su id	
Devuelve	204	No Content
	404	Not Found
	500	Internal Server ERROR
Método	PUT	

Descripción	Actualiza el usuario seleccionado (Aunque se envíe una representación del objeto, a la hora de actualizar el usuario, se ignora el campo id y se utiliza el antiguo)	
Cuerpo de la petición	application/json	
Devuelve	200	OK
	404	Not found
	500	Internal Server Error

URI	**Soscorro/api/messages**	
Método	GET	
Descripción	Devuelve una lista con los mensajes escritos por el usuario creatorId en el muro de id, que puede filtrarse por fecha	
Cadena de consulta	offset =	Posición de la lista en la que empezar a devolver la representación del objeto
	count =	Número de elementos a representar en la lista devuelta
	startDate =	Elección de la fecha inicial para devolver los mensajes
	endDate =	Elección de la fecha final para devolver los mensajes
	creatorId =	Devuelve los mensajes con el id especificado.
	forumId =	Id del foro del usuario
	contentPattern =	Patrón de contenido de mensaje.
	selectMessagesFromIdCreator=	Booleano para elegir si seleccionar los mensajes que

		contengan o no ese id de creador
Devuelve	200	OK + JSON
	400	Bad request
	500	Internal Server Error
Método	POST	
Descripción	Publica un mensaje en el muro del usuario id	
Cuerpo de petición	application/json	
Devuelve	201	CREATED + JSON
	500	Internal Server Error

URI	**Soscorro/api/messages/messageId**	
Método	GET	
Descripción	Devuelve una representación de un mensaje identificado por messageId	
Devuelve	200	OK + JSON
	400	Bad Request
	404	Not found
	500	Internal Server Error
Método	DELETE	
Descripción	Borra el mensaje identificado por messageId	
Devuelve	204	No Content
	404	Not Found
	500	Internal Server Error
Método	PUT	
Descripción	Actualiza el mensaje identificado por messageId	
Cuerpo de la petición	application/json	
Devuelve	200	OK
	404	Not found
	500	Internal Server Error

URI	**Soscorro/api/users/id/friends**	
Método	GET	
Descripción	Devuelve la lista de amigos del usuario	
Cadena de consulta	offset =	Posición de la lista en la que empezar a devolver la representación del objeto
	count =	Número de elementos a representar en la lista devuelta
	namePattern =	Patron de nombre de usuario para filtrar
Devuelve	200	OK + JSON
	400	Bad Request
	500	Internal Server Error
Método	POST	
Descripción	Crea una relacción de amistad unidireccional	
Cadena de consulta	fromUserId =	Usuario que activa la relación de amistad
	toUserId =	Usuario que recibe la relación de amistad
Devuelve	201	CREATED + JSON
	400	Bad Request
	404	Not found
	500	Internal Server Error
Método	DELETE	
Descripción	Destruye una relación de amistad unidireccional	
Cuerpo de petición	application/json	
Devuelve	204	No content
	404	Not found
	500	Internal Server Error

URI	**Soscorro/users/id/friends/messages**
Método	GET

Descripción	Devuelve los mensajes publicados por los amigos en sus muros	
Cadena de consulta	offset =	Posición de la lista en la que empezar a devolver la representación del objeto
	count =	Número de elementos a representar en la lista devuelta
	startDate =	Elección de la fecha inicial para devolver los mensajes
	endDate =	Elección de la fecha final para devolver los mensajes
Devuelve	200	OK + JSON
	400	Bad Request
	500	Internal Server Error
URI	**Soscorro/api/users/id/profile**	
Método	GET	
Descripción	Devuelve datos básicos del usuario, último mensaje de su página, número de amigos y mensajes de sus 10 últimos amigos.	
Devuelve	200	OK + JSON
	400	Bad Request
	404	Not found
	500	Internal Server Error

Para la recoleción y uso de los datos hemos usado una base de datos en SQL:

Utilizamos 3 tablas, una para usuarios, otra para los mensajes y otra para las relaciones de amistad.

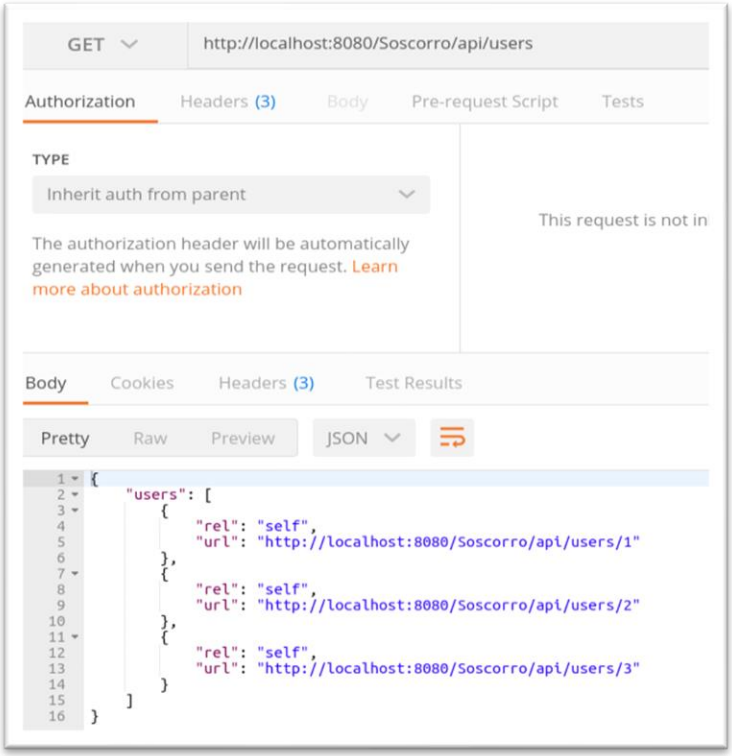
User_has_friends = tabla de amistades
Users = tabla de usuarios
Messages = tabla de mensajes

Table	Column	Type
Messages	messageId	int(11)
Messages	creatorId	int(11)
Messages	forumId	int(11)
Messages	lastModDate	date
Messages	creationDate	date
Messages	content	text
Users	userId	int(11)
Users	userName	varchar(45)
Users_has_friends	userId	int(11)
Users_has_friends	friendId	int(11)

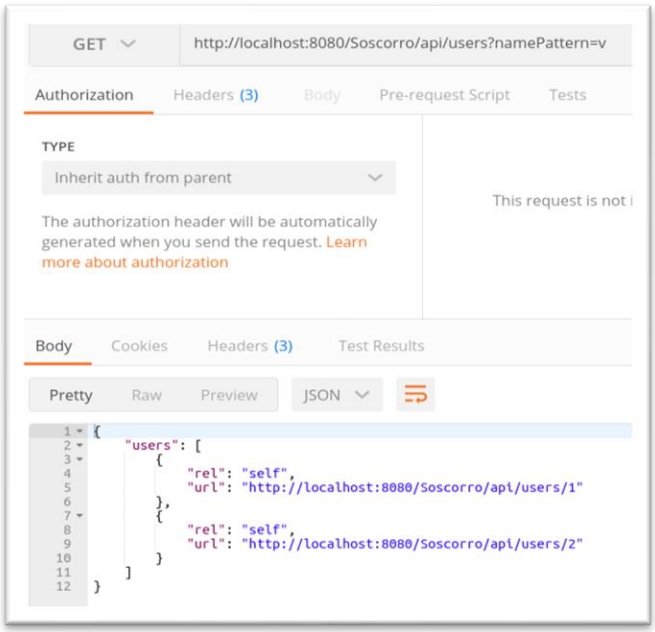
Las tres tablas están relacionadas por el userId, aunque en el caso de los mensajes se llama creatorId.
La tabla Users tiene una relación n:m con la tabla amigos, y users tiene una relación n:m con mensajes.

2. EJECUCIÓN DE OPERACIONES

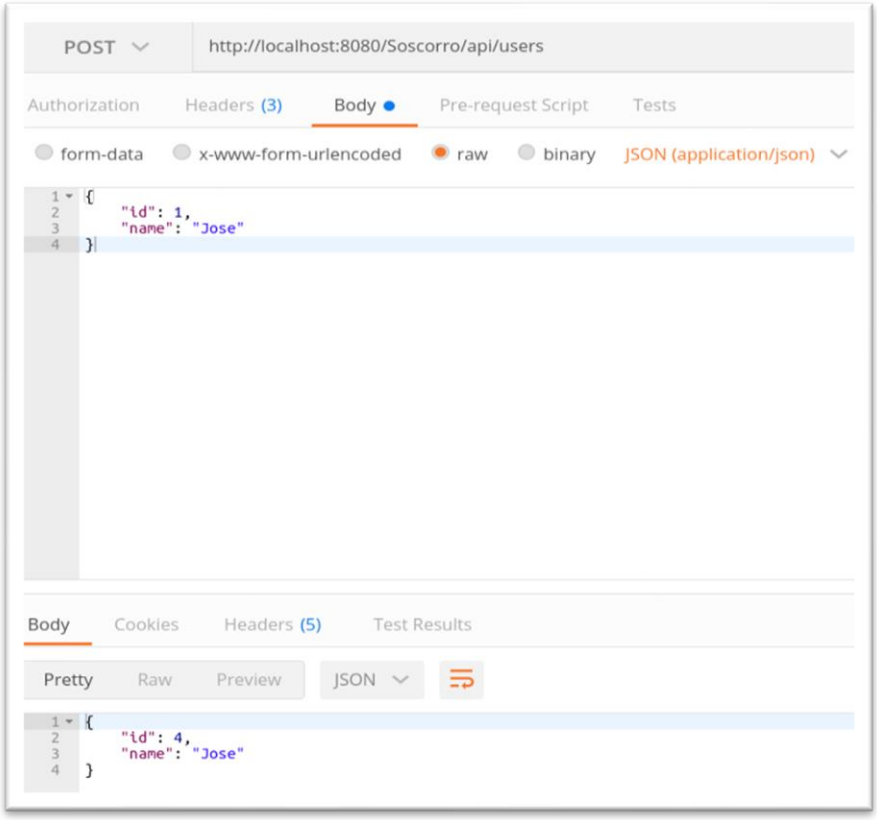
GET para sacar todos los usuarios disponibles en la red social.



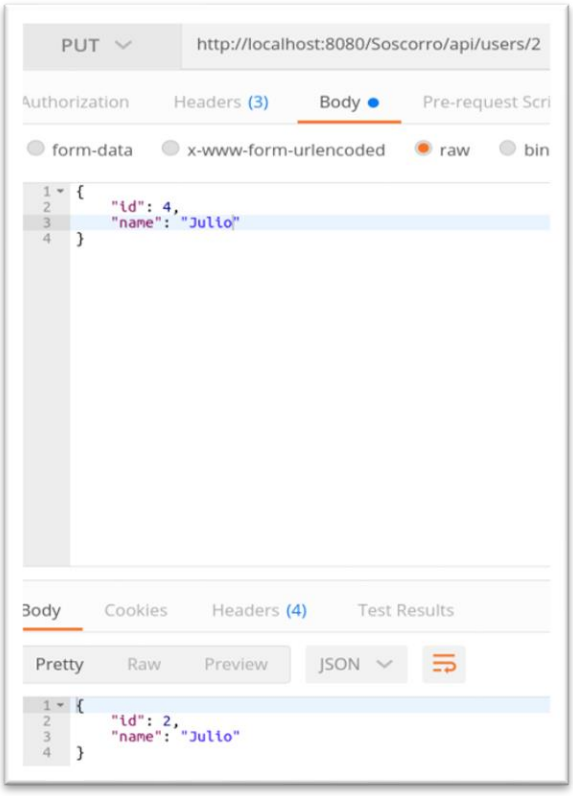
Mismo GET pero con patrón de nombre, buscando una v, tendemos dos nombres compatibles ahora mismo Javier y Victor.



POST para añadir nuevo usuario a la red social.



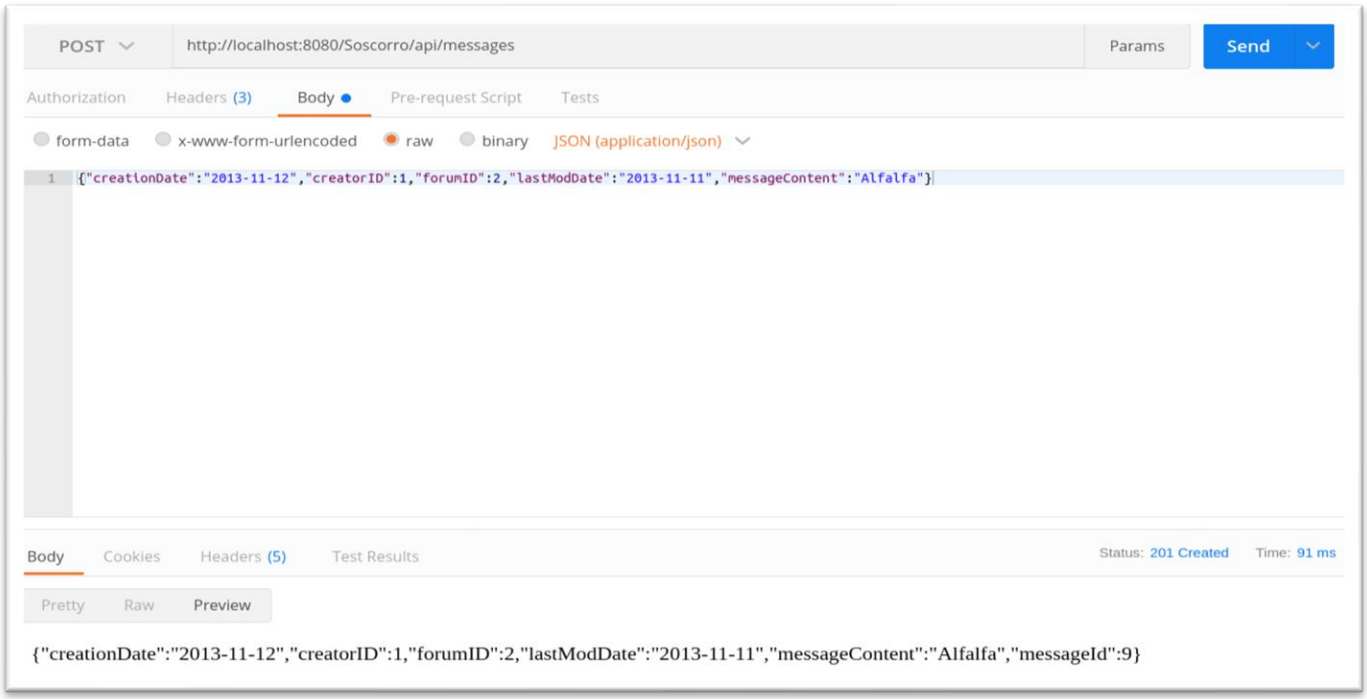
PUT para cambiar nombre de usuario.



DELETE para borra el usuario.

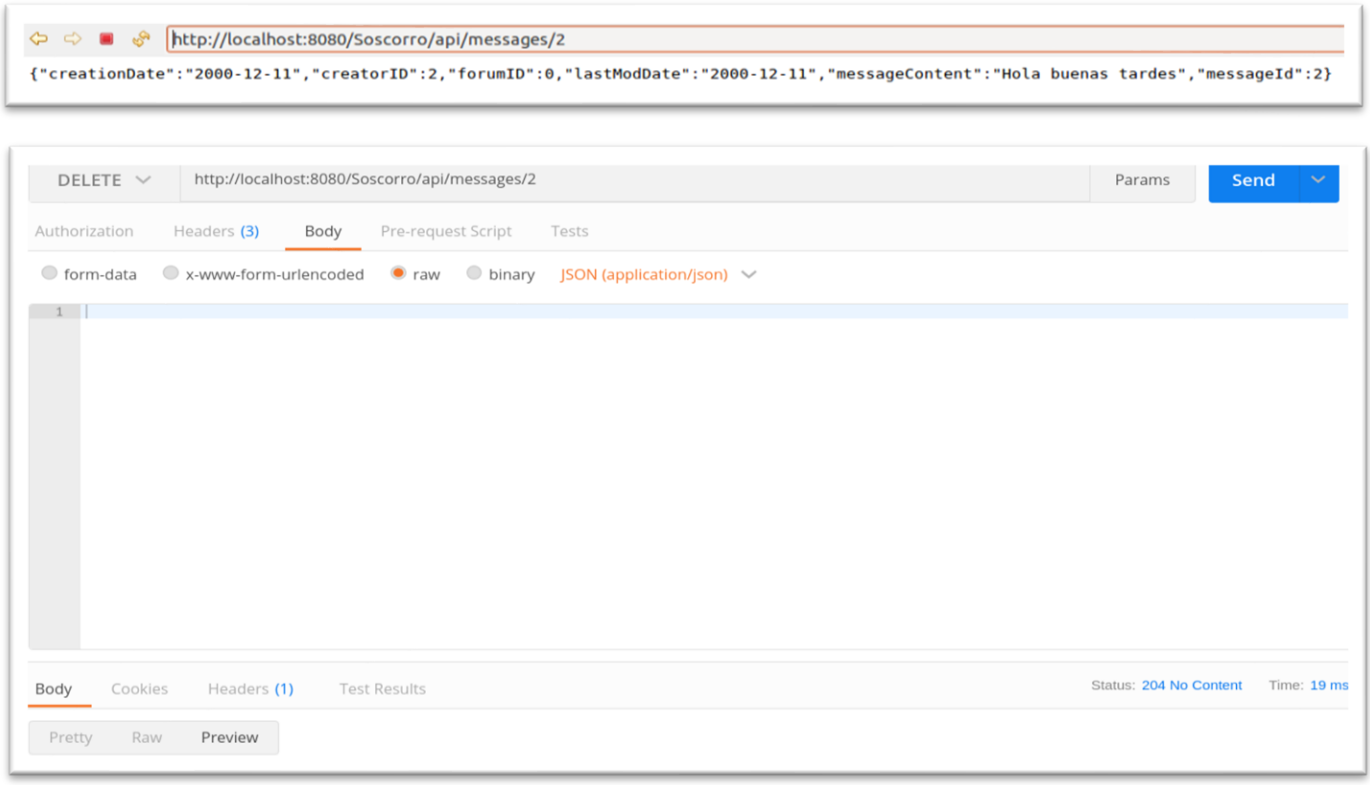
Tenemos 3 uris y al borrarlo solo 2.

POST para añadir el mensaje al su página personal.



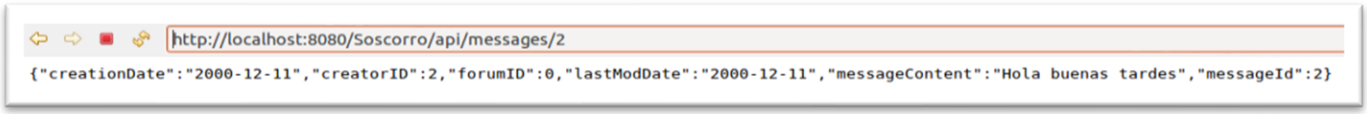
DELETE para eliminar mensaje de su página personal.

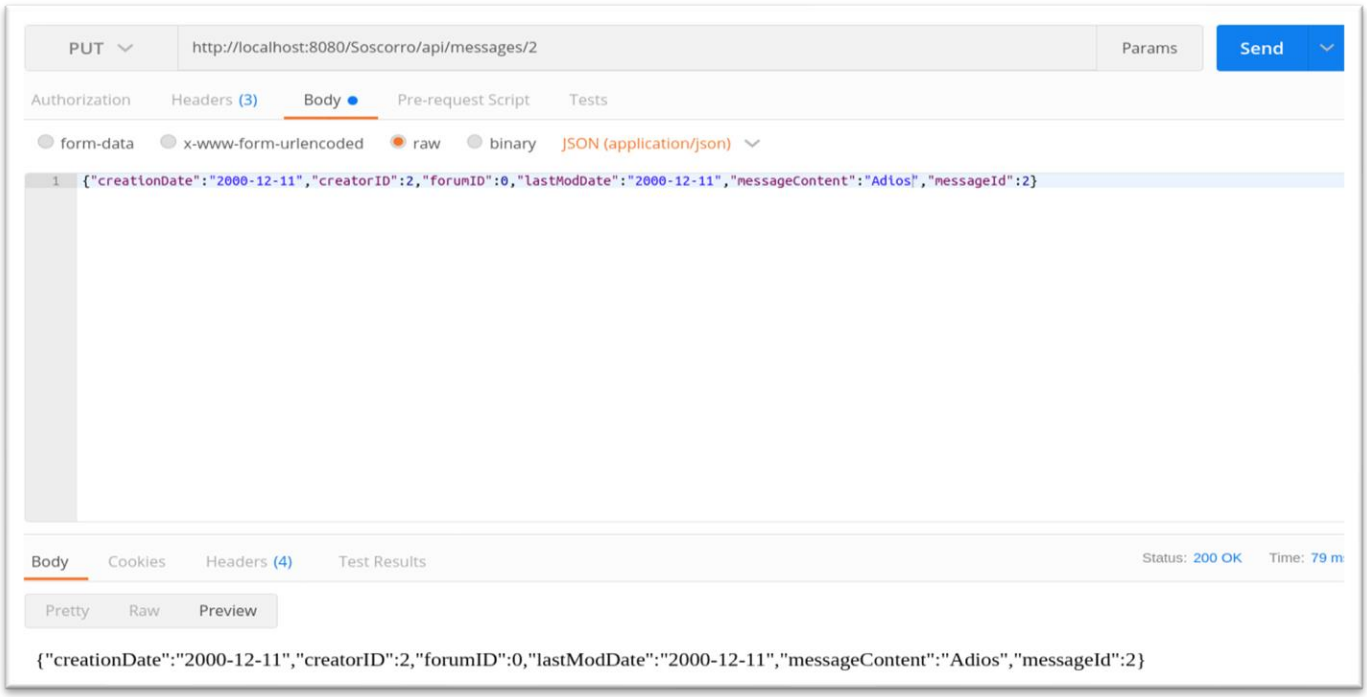
Al devolvernos 204 NoContent sabemos que el mensaje se ha eliminado.



PUT para editar mensaje de su página personal.

El contenido del mensaje es “Buenas tardes” y se ha cambiado por “Adios”.





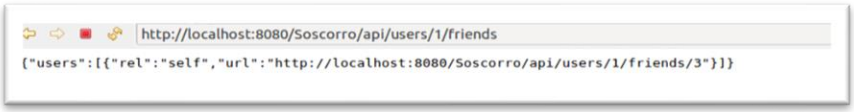
GET para obtener todos los mensajes de un usuario.

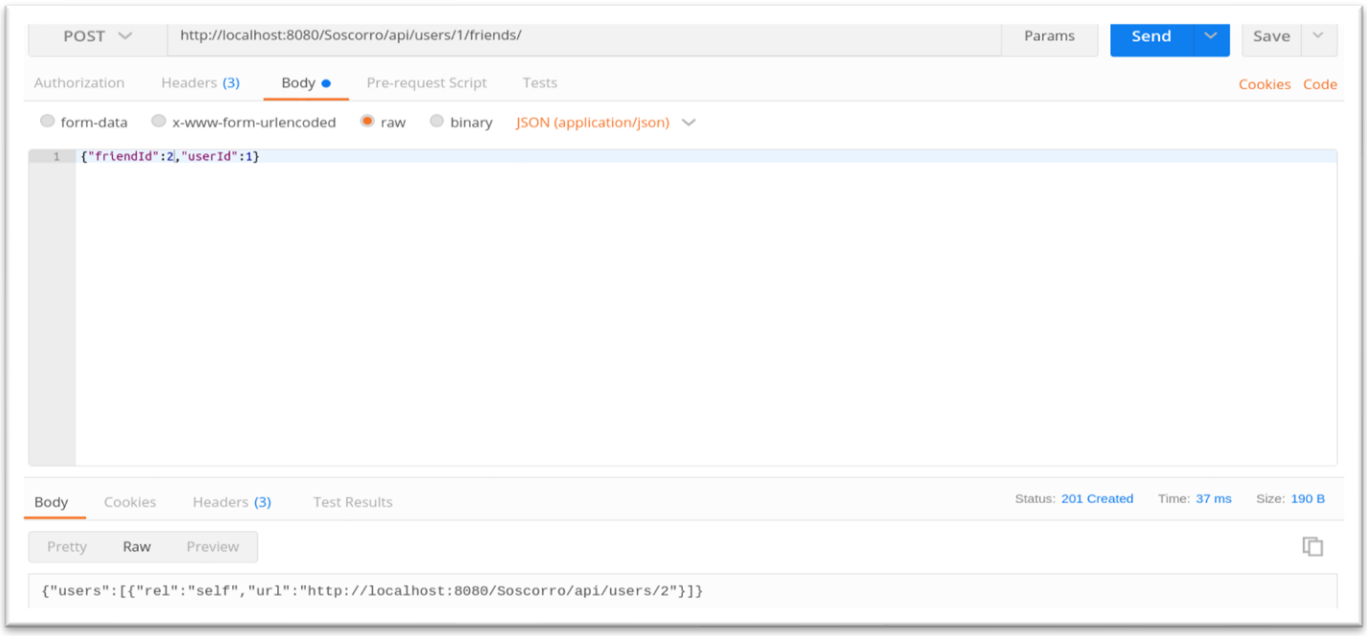
Mismo GET pero con búsqueda por fecha / cantidad.

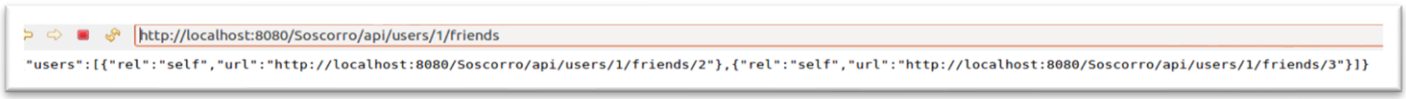
Hemos puesto de ejemplo el primer mensaje pero ahora solo dos tiene fecha mayor a 2010

POST para añadir un amigo.

Mostramos los amigos que había antes y como después hay uno más.

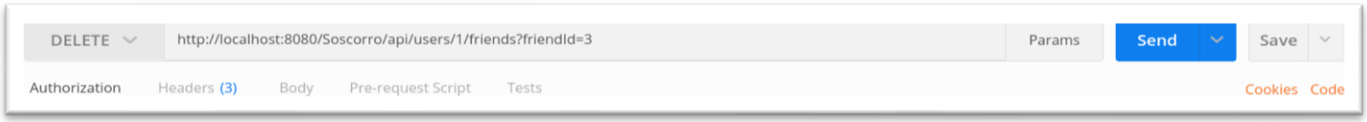






DELETE para borrar un amigo.

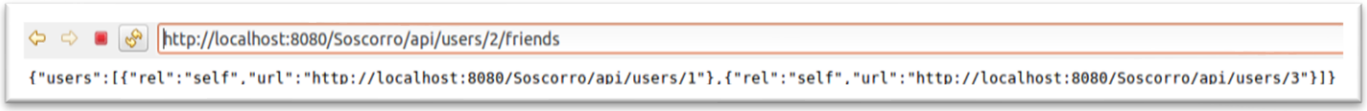
Borramos el amigo número 3.

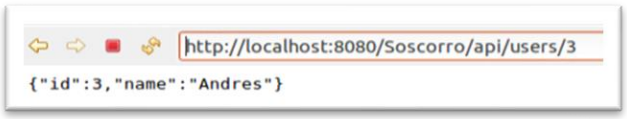


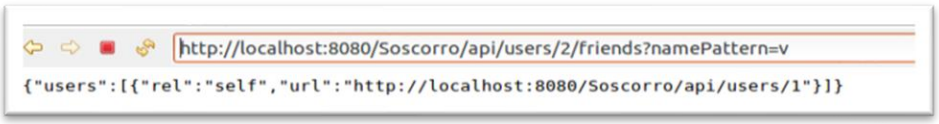
GET para obtener todos los amigos.

GET pero para obtener por patrón de nombre de amigos.

Andrés no tiene v, entonces no sale al volverlo a buscar. (El otro nombre es Javi)

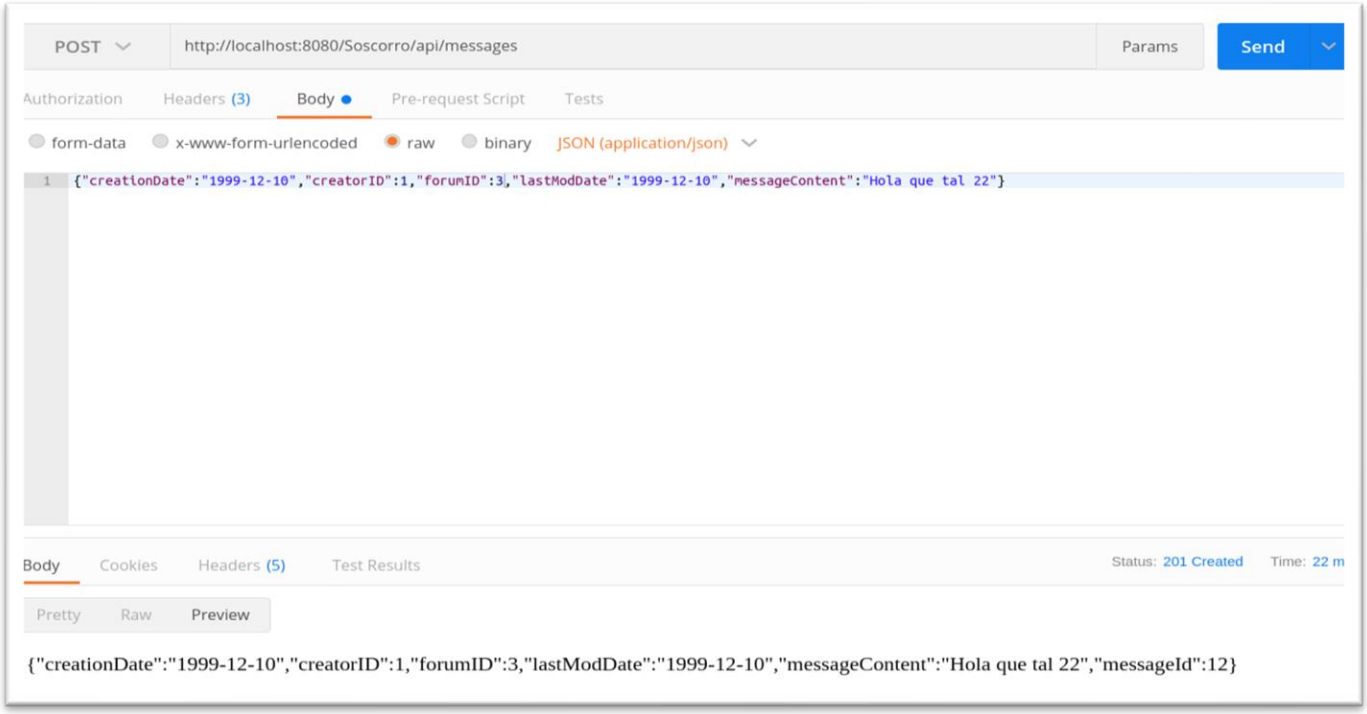






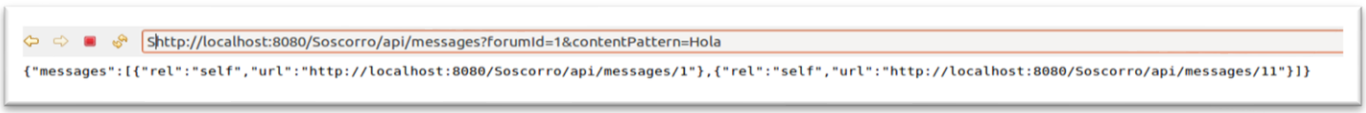
POST para enviar un mensaje a otro usuario en su página personal.

Podemos ver como puedo crear arriba en el forum1 que sería mi usuario y aquí en el forum 3.



GET buscar en todos los mensajes de nuestros amigos por contenido.

Con los demás podemos ver que tenemos más de 2 mensajes, pero con el filtro de Hola solo hay dos.



GET profile

Con este GET recibimos datos de usuario, últimos 10 mensajes de sus amigos y último mensaje escrito por el usuario.