

# API - SOS\_UPMSocial2017

Práctica 1 de una API RESTFull de la asignatura Sistemas Orientados a Servicios (SOS) de la Escuela Técnica Superior de Ingenieros Informáticos (ETISIINF, antigua Facultad de Informática, FI) de la Universidad Politécnica de Madrid (UPM).

## Información

---

Infomacion:	
Titulación	Grado de Ingeniería Informática. Plan 09.
Año	2016-2017
Materia	Sistemas Orientados a Servicios
Curso	3º Curso
Semestre	6º Semestre (Mañana)
Proyecto	Practica 1 - API RESTFull

## Autores

---

- Diego Fernández Peces-Barba [t110271]
- Gonzalo Montiel Penedo [t110362]

## Índice

---

[TOC]

## Diseño de la Base de Datos

---

**Nota:** Dado que la practica se basa en el diseño de una API Restful, en la implementación de un servidor y cliente para dicha API, hemos creado sólo lo necesario para que funcione correctamente.

Parametros de la base de datos:

- URI de la base de datos: `diegofpb.asuscomm.com:3306` (Alojada en una Raspberry PI 2)
- Nombre de la base de datos: `RestBBDD` .

- Nombre del usuario: `upmsocialapi`.
- Contraseña del usuario: `dieguito1`.

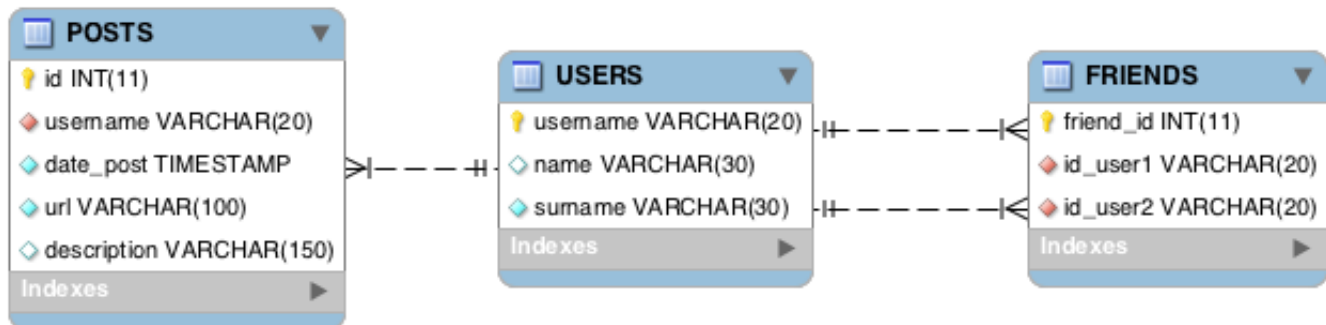
Para la base de datos disponemos de tres tablas:

- **USERS** (Tabla con datos de los usuarios).
- **POSTS** (Tabla con posts de los usuarios).
- **FRIENDS** (Tabla con amistades de los usuarios).

## Esquema E/R

El esquema de Entidad Relación diseñado para la base de datos de la práctica es el siguiente:

Entidad / Relación de la base de datos:



## Tabla USERS

Tabla donde se guardaran los usuarios de la plataforma.

Los atributos de la tabla serían los siguientes:

username	name	surname
VARCHAR	VARCHAR	VARCHAR

## Tabla POSTS

Tabla donde se guardaran los post o mensajes que creen los usuarios de la plataforma.

Los atributos de la tabla serían los siguientes:

id	username	date_post	url	description
INT	VARCHAR	TIMESTAMP	VARCHAR	VARCHAR

## Tabla FRIENDS

Nos hemos basado en el estilo de Facebook. Un amigo pide petición de amistad a otro, y directamente se genera un vínculo de amistad entre ellos, no hace falta que el otro solicite amistad al primero, ya que pertenece al vínculo de amistad.

Los atributos de la tabla serían los siguientes:

friend_id	id_user1	id_user2
INT	VARCHAR	VARCHAR

## Código SQL de creación de la base de datos

**Nota:** La base de datos no tiene por que estar vacía. El cliente funciona tanto si hay datos ya introducidos como si no.

Código:

```
CREATE SCHEMA IF NOT EXISTS `RestBBDD` DEFAULT CHARACTER SET utf8 ;
USE `RestBBDD` ;

-----
--          TABLA USERS
-----

DROP TABLE IF EXISTS `USERS`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `USERS` (
  `username` varchar(20) NOT NULL,
  `name` varchar(30) DEFAULT NULL,
  `surname` varchar(30) NOT NULL,
  PRIMARY KEY (`username`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-----
--          TABLA POSTS
-----

DROP TABLE IF EXISTS `POSTS`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `POSTS` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
```

```

`username` varchar(20) NOT NULL,
`date_post` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
`url` varchar(100) NOT NULL,
`description` varchar(150) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id` (`id`),
KEY `username` (`username`),
CONSTRAINT `POSTS_ibfk_1` FOREIGN KEY (`username`) REFERENCES `USERS` (`username`
) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=150 DEFAULT CHARSET=latin1;

```

```

-----
--          TABLA FRIENDS
-----
DROP TABLE IF EXISTS `FRIENDS`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `FRIENDS` (
  `friend_id` int(11) NOT NULL AUTO_INCREMENT,
  `id_user1` varchar(20) NOT NULL,
  `id_user2` varchar(20) NOT NULL,
  PRIMARY KEY (`friend_id`),
  UNIQUE KEY `friend_id` (`friend_id`),
  KEY `id_user1` (`id_user1`),
  KEY `id_user2` (`id_user2`),
  CONSTRAINT `FRIENDS_ibfk_1` FOREIGN KEY (`id_user1`) REFERENCES `USERS` (`userna
me`),
  CONSTRAINT `FRIENDS_ibfk_2` FOREIGN KEY (`id_user2`) REFERENCES `USERS` (`userna
me`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=latin1;

```

## Diseño de las URIs

---

### Users

[GET] ~/users

URI	http://localhost:8080/UPMSocial/api/v1/users/
Descripción	Devuelve una lista de usuarios de toda la red. Si se le incluye parámetro <i>filterbytext</i> , solo devuelve los que coincidan con ese requisito.
Método	GET
Cadena de consulta	<ul style="list-style-type: none"> <li><i>filterbytext</i>= búsqueda por nombre</li> </ul>
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK y POX (usuarios/usuario+xml)</li> <li>404 : Not Found</li> </ul>

## [POST] ~/users

URI	http://localhost:8080/UPMSocial/api/v1/users/
Descripción	Permite crear un usuario, enviando una estructura XML del tipo User.
Método	POST
Cadena de consulta	Ninguna
Cuerpo	POX (users/user+xml)
Devuelve	<ul style="list-style-type: none"> <li>201 : Created y cabecera Location</li> <li>406 : Not Acceptable</li> <li>415 : Unsupported Media Type</li> </ul>

## [GET] ~/users/{username}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}
Descripción	Devuelve la estructura XML de un usuario, mediante el uso del parametro de la ruta.
Método	GET
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK y POX (usuarios/usuario+xml)</li> <li>404 : Not Found</li> </ul>

### [PUT] ~/users/{username}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}
Descripción	Modifica los atributos de un usuario
Método	PUT
Cadena de consulta	Ninguna
Cuerpo	POX (users/user+xml)
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>201 : Created y cabecera Location *(1)</li> <li>406 : Not Acceptable</li> <li>415 : Unsupported Media Type</li> </ul>

- (1): En caso de no existir, el usuario se crea.

### [DELETE] ~/users/{username}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}
Descripción	Elimina un un usuario los atributos de un usuario
Método	DELETE
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> <li>503 : Service Unavailable</li> </ul>

## Friends

### [GET] ~/users/{username}/friends

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/friends
Descripción	Obtiene los amigos de un usuario
Método	GET
Cadena de consulta	<ul style="list-style-type: none"> <li>start= Desde número</li> <li>end= Hasta número</li> <li>filterbyname= Búsqueda por nombre</li> </ul>
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> </ul>

### [POST] ~/users/{username}/friends/{username\_2}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/friends/{username_2}
Descripción	Crea una amistad de un usuario con otro
Método	POST
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>201 : Created y cabecera Location</li> <li>302 : Found</li> <li>404 : Not Found</li> </ul>

## [DELETE] ~/users/{username}/friends/{username\_2}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/friends/{username_2}
Descripción	Elimina una relaciond de amistad entre dos usuarios
Método	DELETE
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> <li>503 : Service Unavailable</li> </ul>

## Posts

### [GET] ~/users/{username}/posts



URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts
Descripción	Obtiene los posts de un usuario
Método	GET
Cadena de consulta	<ul style="list-style-type: none"> <li>• start= Desde número</li> <li>• end= Hasta número</li> <li>• from= Desde fecha Hasta fecha (YYYY-MM-DD)</li> <li>• to= Hasta fecha (YYYY-MM-DD)</li> <li>• texttosearch= Búsqueda de texto en la descripción de los posts.</li> </ul>
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>• 200 : OK</li> <li>• 404 : Not Found</li> </ul>

### [POST] ~/users/{username}/posts

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts
Descripción	Publica un posts para un usuario
Método	POST
Cadena de consulta	Ninguna
Cuerpo	POX (posts/post+xml)
Devuelve	<ul style="list-style-type: none"> <li>• 201 : Created y cabecera Location</li> <li>• 404 : Not found</li> <li>• 406 : Not acceptable</li> <li>• 415 : Unsupported Media Type</li> </ul>

### [GET] ~/users/{username}/posts/count

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts/count
Descripción	Devuelve el número de posts de un usuario
Método	GET
Cadena de consulta	<ul style="list-style-type: none"> <li>from= Desde fecha Hasta fecha (YYYY-MM-DD)</li> <li>to= Hasta fecha (YYYY-MM-DD)</li> </ul>
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> </ul>

### [GET] ~/users/{username}/posts/{post\_id}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts/{post_id}
Descripción	Devuelve el contenido de un único post por su id.
Método	GET
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> </ul>

### [PUT] ~/users/{username}/posts/{post\_id}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts/{post_id}
Descripción	Modifica el contenido de un post.
Método	PUT
Cadena de consulta	Ninguna
Cuerpo	POX (posts/post+xml)
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> <li>406 : Not Acceptable</li> </ul>

### [DELETE] ~/users/{username}/posts/{post\_id}

URI	http://localhost:8080/UPMSocial/api/v1/users/{username}/posts/{post_id}
Descripción	Elimina un post de un usuario por un ID.
Método	DELETE
Cadena de consulta	Ninguna
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> <li>503 : Service Unavailable</li> </ul>

### [GET] ~/users/{user\_id}/friends/posts

URI	http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friends/posts
Descripción	Obtiene los posts de los amigos de un usuario.
Método	GET
Cadena de consulta	<ul style="list-style-type: none"> <li>start= Desde número</li> <li>end= Hasta número</li> <li>from= Desde fecha Hasta fecha (YYYY-MM-DD)</li> <li>to= Hasta fecha (YYYY-MM-DD)</li> <li>texttosearch= Búsqueda de texto en la descripción de los posts.</li> </ul>
Cuerpo	Ninguno
Devuelve	<ul style="list-style-type: none"> <li>200 : OK</li> <li>404 : Not Found</li> </ul>

## Pruebas API RESTfull

La API RESTfull se testeo mediante la aplicación POSTMAN. Se realizaban pruebas de cada metodo por separado. Al finalizar toda la API, se lanzó el cliente.

### Pruebas mediante Postman

#### USERS

[GET] ~/users (Obtener todos los usuarios de la BBDD)

GET

http://localhost:8080/UPMSocial/api/v1/users

Params

Body

Cookies

Headers (5)

Tests

Status: 200 OK

Pretty

Raw

Preview

XML

1

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

2

<users>

3

<user>

4

<name>Antonio</name>

5

<surname>Recio</surname>

6

<username>arecio\_test</username>

7

</user>

8

<user>

9

<name>CacaPepe</name>

10

<surname>Caca</surname>

11

<username>caca</username>

12

</user>

13

<user>

</user>

18

<user>

</user>

23

<user>

</user>

28

<user>

</user>

33

<user>

</user>

38

<user>

</user>

43

<user>

</user>

48

<user>

</user>

53

<user>

54

<name>User2</name>

55

<surname>Test2</surname>

56

<username>user\_test2</username>

57

</user>

58

</users>

[GET] ~/users?filterbytext=antonio (Obtener usuarios que se llamen, apelliden o cuyo nombre de usuario sea antonio)

GET

http://localhost:8080/UPMSocial/api/v1/users?filter\_by\_text=antonio

Params

Key	Value
filter_by_text	antonio
New key	value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (5)

Tests

Status: 200 OK

Pretty

Raw

Preview

XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <users>
3   <user>
4     <name>Antonio</name>
5     <surname>Recio</surname>
6     <username>arecio_test</username>
7   </user>
8 </users>
```

[POST] ~/users (Crear un usuario nuevo mediante modelo USER-XML)

POST ▾

http://localhost:8080/UPMSocial/api/v1/users

Params

Authorization

Headers (1)

Body ●

Pre-request Script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

XML (application/xml) ▾

1 ▾

<user>

2     <name>Rebeca</name>

3     <surname>Ortiz</surname>

4     <username>rortiz\_test</username>

5 </user>

Body

Cookies

Headers (5)

Tests

Status: 201 Created

Content-Length → 0

Date → Wed, 03 May 2017 09:26:37 GMT

Location → http://localhost:8080/UPMSocial/api/v1/users/rortiz\_test

Server → GlassFish Server Open Source Edition 4.1.1

X-Powered-By → Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1.1 Java/Oracle Corporation/1.8)

[GET] ~/users/{username} (Obtener un usuario especifico mediante su nombre de usuario)

GET http://localhost:8080/UPMSocial/api/v1/users/rortiz\_test Params

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (6) Tests Status: 200 OK

Pretty Raw Preview XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <user>
3   <name>Rebeca</name>
4   <surname>Ortiz</surname>
5   <username>rortiz_test</username>
6 </user>
```

[PUT] ~/users/{username} (Modificar el nombre o apellido de un usuario mediante modelo USER-XML)

PUT http://localhost:8080/UPMSocial/api/v1/users/rortiz\_test Params

Authorization Headers (1) Body Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary XML (application/xml)

```
1 <user>
2   <name>Ahora me llamo Patricia</name>
3   <surname>Anselma</surname>
4   <username>rortiz_test</username>
5 </user>
```

Body Cookies Headers (4) Tests Status: 200 OK

[DELETE] ~/users/{username} (Eliminar el perfil del usuario de la red. A su vez, borra posibles amistades y posts publicados)

DELETE

http://localhost:8080/UPMSocial/api/v1/users/rortiz\_test

Params

Authorization

Headers (1)

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (4)

Tests

Status: 200 OK

## FRIENDS

[GET] ~/users/{username}/friends (Obtiene todos los amigos de un usuario)

GET

http://localhost:8080/UPMSocial/api/v1/users/user\_test/friends/

Params

Body

Cookies

Headers (5)

Tests

Status: 200 OK

Pretty

Raw

Preview

XML

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <friendships>
3    <friendship friendshipId="43">
4      <id_user1>user_test</id_user1>
5      <id_user2>user_test2</id_user2>
6    </friendship>
7    <friendship friendshipId="44">
8      <id_user1>user_test</id_user1>
9      <id_user2>arecio_test</id_user2>
10   </friendship>
11 </friendships>

```

[GET] ~/users/{username}/friends?filterbyname=arecio (Obtiene todos los amigos de un usuario, cuyo nombre de usuario sea arecio)



GET	http://localhost:8080/UPMSocial/api/v1/users/user_test/friends?filter_by_name=arecio	Params
-----	--	--------

---

Body	Cookies	Headers (5)	Tests	Status: 200 OK
------	---------	-------------	-------	----------------

---

Pretty	Raw	Preview	XML	
--------	-----	---------	-----	--

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <friendships>
3   <friendship friendshipId="44">
4     <id_user1>user_test</id_user1>
5     <id_user2>arecio_test</id_user2>
6   </friendship>
7 </friendships>

```

**[POST] ~/users/{username}/friends/{username\_2}** (Crea una relación de amistad entre dos nombres de usuario)

POST	http://localhost:8080/UPMSocial/api/v1/users/user_test/friends/areci...	Params
------	---	--------

---

Body	Cookies	Headers (5)	Tests	Status: 201 Created
------	---------	-------------	-------	---------------------

---

**Content-Length** → 0

**Date** → Wed, 03 May 2017 09:45:17 GMT

**Location** → http://localhost:8080/UPMSocial/api/v1/users/user\_test/friends/arecio\_test

**Server** → GlassFish Server Open Source Edition 4.1.1

**X-Powered-By** → Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1.1 Java/Oracle Corporation/1.8)

**[DELETE] ~/users/{username}/friends/{username\_2}** (Elimina una relacion de amistad entre dos nombres de usuario)

DELETE	http://localhost:8080/UPMSocial/api/v1/users/user_test/friends/arecio_test	Params
--------	--	--------

---


Body	Cookies	Headers (4)	Tests	Status: 200 OK
------	---------	-------------	-------	----------------

## POSTS

**[GET] ~/users/{username}/posts** (Obtiene todos los posts de un usuario)

GET ▼ http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts Params

Body Cookies Headers (5) Tests Status: 200 OK

Pretty Raw Preview XML ▼ 

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <posts>
3   <post id="162">
4     <date_post>2017-04-27</date_post>
5     <description>Google Website</description>
6     <url>http://google.es</url>
7     <username>user_test</username>
8   </post>
9   <post id="165">
10    <date_post>2017-05-03</date_post>
11    <description>SOS</description>
12    <url>http://sos.damepaciencia.es</url>
13    <username>user_test</username>
14  </post>
15 </posts>

```

## [POST] ~/users/{username}/posts (Crea un post para un usuario)

POST ▼ http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts Params

Authorization Headers (1) Body ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary XML (application/xml) ▼

```

1 <post>
2   <date_post>2017-05-03</date_post>
3   <description>SOS</description>
4   <url>http://sos.damepaciencia.es</url>
5   <username>user_test</username>
6 </post>

```

Body Cookies Headers (5) Tests Status: 201 Created

**Content-Length** → 0

**Date** → Wed, 03 May 2017 09:59:18 GMT

**Location** → http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts/165

**Server** → GlassFish Server Open Source Edition 4.1.1

**X-Powered-By** → Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1.1 Java/Oracle Corporation/1.8)

**[GET] ~/users/{username}/posts/count (Obtiene el numero de posts de un usuario)**

GET

http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts/count

Params

Body

Cookies

Headers (6)

Tests

Status: 200 OK

Pretty

Raw

Preview

2

**[GET] ~/users/{username}/posts/{post\_id} (Obtiene un post de un determinado usuario mediante el id del post)**

GET

http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts/165

Params

Body

Cookies

Headers (5)

Tests

Status: 200 OK

Pretty


Raw


Preview






XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <post id="165">
3   <date_post>2017-05-03</date_post>
4   <description>SOS</description>
5   <url>http://sos.damepaciencia.es</url>
6   <username>user_test</username>
7 </post>
```

**[PUT] ~/users/{username}/posts/{post\_id} (Modifica el post de un usuario mediante el id del post y el modelo POST-XML)**

PUT  http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts/165 Params

Authorization Headers (1) **Body**  Pre-request Script Tests

 form-data  x-www-form-urlencoded  raw  binary XML (application/xml) 

```
1 <post>
2   <date_post>2017-05-03</date_post>
3   <description>SOS Modificado</description>
4   <url>http://sos.eslaleche.es</url>
5   <username>user_test</username>
6 </post>
```

Body Cookies Headers (4) Tests Status: 200 OK

**[DELETE] ~/users/{username}/posts/{post\_id}** (Elimina un post de un usuario mediante el id del post.)

DELETE  http://localhost:8080/UPMSocial/api/v1/users/user\_test/posts/165 Params

Body Cookies **Headers (4)** Tests Status: 200 OK

**Content-Length** → 0

**Date** → Wed, 03 May 2017 10:07:53 GMT

**Server** → GlassFish Server Open Source Edition 4.1.1

**X-Powered-By** → Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.1.1 Java/Oracle Corporation/1.8)

**[GET] ~/users/{user\_id}/friends/posts** (Obtiene los posts de los amigos de un usuario)

GET

http://localhost:8080/UPMSocial/api/v1/users/garra/friends/posts

Params

Body

Cookies

Headers (5)

Tests

Status: 200 OK

Pretty

Raw

Preview

XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <posts>
3   <post id="23">
4     <date_post>2017-04-18</date_post>
5     <description>hehe</description>
6     <url>http://pepe.com</url>
7     <username>rober</username>
8   </post>
9   <post id="24">
10    <date_post>2017-03-18</date_post>
11    <description>hehe</description>
12    <url>http://pepe.com</url>
13    <username>rober</username>
14  </post>
15  <post id="26">
16    <date_post>2017-01-20</date_post>
17    <description>hehe</description>
18    <url>http://pepe.com</url>
19    <username>rober</username>
20  </post>
21  <post id="25">
22    <date_post>2016-03-18</date_post>
23    <description>hehe</description>
24    <url>http://pepe.com</url>
25    <username>rober</username>
26  </post>
27 </posts>
```

## Pruebas mediante Cliente JAVA

Operaciones.

```
// Peticion 0 [DELETE] => Borrar usuarios de la red social para comenzar de 0.
    System.out.println("Peticion 0 [DELETE] Dando de baja los perfiles de prueba anteriores (user_test y user_test2) y demas...");
    Response del_res = target.path("api/v1/users/user_test")
        .request()
        .delete();

    System.out.println("Estado: " + del_res.getStatus());
    del_res.close();

    del_res = target.path("api/v1/users/user_test2")
        .request()
        .delete();
    System.out.println("Estado: " + del_res.getStatus());
    del_res.close();

    del_res = target.path("api/v1/users/arecio_test")
        .request()
        .delete();
    System.out.println("Estado: " + del_res.getStatus());
    del_res.close();

    del_res = target.path("api/v1/users/henry_test")
        .request()
        .delete();
    System.out.println("Estado: " + del_res.getStatus());
    del_res.close();

    del_res = target.path("api/v1/users/elcuqui_test")
        .request()
        .delete();
    System.out.println("Estado: " + del_res.getStatus());
    del_res.close();
```

Peticion 0 [DELETE] Dando de baja los perfiles de prueba anteriores (*usertest* y *usertest2*) y demas...  
 Estado: 200 Estado: 200 Estado: 200 Estado: 200 Estado: 200

```
// Peticion 1 [POST] => Crear Usuario
    System.out.println("\nPeticion 1 [POST] crear 2 usuarios de test y 3 mas para otras pruebas...");
    User user_test = new User("User","Test","user_test");
    User user_test2 = new User("User2","Test2","user_test2");
    User user_test3 = new User("Antonio","Recio","arecio_test");
    User user_test4 = new User("Enrique","Pastor","henry_test");
    User user_test5 = new User("Amador","Rivas","elcuqui_test");
```

```
Response response = target.path("api/v1/users")
    .request()
    .post(Entity.xml(user_test));

System.out.println("Estado: " + response.getStatus());
if (response.getStatus()==201){
    System.out.println("Location: " + response.getLocation()+"\n");
}
response.close();

response = target.path("api/v1/users")
    .request()
    .post(Entity.xml(user_test2));

System.out.println("Estado: " + response.getStatus());
if (response.getStatus()==201){
    System.out.println("Location: " + response.getLocation()+"\n");
}
response.close();

response = target.path("api/v1/users")
    .request()
    .post(Entity.xml(user_test3));

System.out.println("Estado: " + response.getStatus());
if (response.getStatus()==201){
    System.out.println("Location: " + response.getLocation()+"\n");
}
response.close();

response = target.path("api/v1/users")
    .request()
    .post(Entity.xml(user_test4));

System.out.println("Estado: " + response.getStatus());
if (response.getStatus()==201){
    System.out.println("Location: " + response.getLocation()+"\n");
}
response.close();

response = target.path("api/v1/users")
    .request()
    .post(Entity.xml(user_test5));
```

```

System.out.println("Estado: " + response.getStatus());
if (response.getStatus()==201){
    System.out.println("Location: " + response.getLocation()+"\n");
}
response.close();

```

Peticion 1 [POST] crear 2 usuarios de test y 3 mas para otras pruebas... Estado: 201 Location: http://localhost:8080/UPMSocial/api/v1/users/user\_test

Estado: 201 Location: http://localhost:8080/UPMSocial/api/v1/users/user\_test2

Estado: 201 Location: http://localhost:8080/UPMSocial/api/v1/users/arecio\_test

Estado: 201 Location: http://localhost:8080/UPMSocial/api/v1/users/henry\_test

Estado: 201 Location: http://localhost:8080/UPMSocial/api/v1/users/elcuqui\_test

```

// Peticion 2 [GET] => Obtener dicho usuario.
    System.out.println("\nPeticion 2 (1/2) [GET] Obtenemos los datos del primer usuario creado ahora mismo...");
    User user_test_verify = target.path("api/v1/users/user_test")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(User.class);

    System.out.println( "Name: "+user_test_verify.getName()+"\n"+
        "Surname: "+user_test_verify.getSurname()+"\n"+
        "Username: "+user_test_verify.getUsername()+"\n");

    System.out.println("\nPeticion 2 (2/2) [GET] Obtenemos los datos del segundo usuario creado ahora mismo...");
    User user_test_verify2 = target.path("api/v1/users/user_test2")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(User.class);

    System.out.println( "Name: "+user_test_verify2.getName()+"\n"+
        "Surname: "+user_test_verify2.getSurname()+"\n"+
        "Username: "+user_test_verify2.getUsername()+"\n");

```

Peticion 2 (1/2) [GET] Obtenemos los datos del primer usuario creado ahora mismo... Name: User Surname: Test Username: user\_test

Peticion 2 (2/2) [GET] Obtenemos los datos del segundo usuario creado ahora mismo... Name: User2 Surname: Test2 Username: user\_test2

```

// Peticion 3 [POST] => Crear un post de dicho usuario.

```



```

        System.out.println("\nPeticion 3 (1/3) [POST] => Crear un post del primer
usuario...");

        Post post_test1 = new Post();
        post_test1.setDate_post("2017-04-27");
        post_test1.setDescription("Google Website");
        post_test1.setUrl("http://google.es");
        post_test1.setUsername("user_test");

        response = target.path("api/v1/users/user_test/posts")
                .request()
                .post(Entity.xml(post_test1));

        System.out.println("Estado: " + response.getStatus());

        if (response.getStatus()==201){
            System.out.println("Location: " + response.getLocation());
        }

        response.close();

        System.out.println("\nPeticion 3 (2/3) [POST] => Crear segundo post del pr
imer usuario...");

        Post post_test3 = new Post();
        post_test3.setDate_post("2017-04-27");
        post_test3.setDescription("Tuentis Website");
        post_test3.setUrl("http://tuenti.es");
        post_test3.setUsername("user_test");

        response = target.path("api/v1/users/user_test/posts")
                .request()
                .post(Entity.xml(post_test3));

        System.out.println("Estado: " + response.getStatus());

        if (response.getStatus()==201){
            System.out.println("Location: " + response.getLocation());
        }

        URI location_post_2 = response.getLocation();

        response.close();

```

```

        System.out.println("\nPeticion 3 (3/3) [POST] => Crear un post del segundo
usuario...");

        Post post_test2 = new Post();
        post_test2.setDate_post("2017-04-27");
        post_test2.setDescription("Facebook Website");
        post_test2.setUrl("http://facebook.es");
        post_test2.setUsername("user_test2");

        response = target.path("api/v1/users/user_test2/posts")
                .request()
                .post(Entity.xml(post_test2));

        System.out.println("Estado: " + response.getStatus());

        if (response.getStatus()==201){
            System.out.println("Location: " + response.getLocation());
        }

        URI location_post_1 = response.getLocation();

        response.close();

        Post post_test4 = new Post();
        post_test4.setDate_post("2017-05-02");
        post_test4.setDescription("Mariscos Recio");
        post_test4.setUrl("http://MariscosRecio.es");
        post_test4.setUsername("arecio_test");

        response = target.path("api/v1/users/arecio_test/posts")
                .request()
                .post(Entity.xml(post_test4));

        System.out.println("Estado: " + response.getStatus());

        if (response.getStatus()==201){
            System.out.println("Location: " + response.getLocation());
        }

        response.close();

```

Peticion 3 (1/3) [POST] => Crear un post del primer usuario... Estado: 201 Location:  
[http://localhost:8080/UPMSocial/api/v1/users/user\\_test/posts/41](http://localhost:8080/UPMSocial/api/v1/users/user_test/posts/41)

Peticion 3 (2/3) [POST] => Crear segundo post del primer usuario... Estado: 201 Location:  
[http://localhost:8080/UPMSocial/api/v1/users/user\\_test/posts/42](http://localhost:8080/UPMSocial/api/v1/users/user_test/posts/42)

Peticion 3 (3/3) [POST] => Crear un post del segundo usuario... Estado: 201 Location:  
[http://localhost:8080/UPMSocial/api/v1/users/user\\_test2/posts/43](http://localhost:8080/UPMSocial/api/v1/users/user_test2/posts/43) Estado: 201 Location:  
<http://localhost:8080/UPMSocial/api/v1/users/areciotest/posts/44>

```
// Peticion 4 [GET] => Obtener posts de un usuario (user_test).
    System.out.println("\nPeticion 4 [GET] Obtenemos los posts de un usuario (
user_test)...");

    System.out.println(target.path("api/v1/users/user_test/posts")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 4 [GET] Obtenemos los posts de un usuario (user\_test)...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><posts><post id="41"><date_
post>2017-04-27</date_post><description>Google Website</description><url>http://go
ogle.es</url><username>user_test</username></post><post id="42"><date_post>2017-04
-27</date_post><description>Tuentis Website</description><url>http://tuenti.es</ur
l><username>user_test</username></post></posts>
```

```
// Peticion 5 [PUT] => Modificar un post de un usuario.
    System.out.println("\nPeticion 5 [PUT] => Modificar el post del primer usu
ario...");

    Post post_put_1 = new Post();
    post_put_1.setDate_post("2017-05-01");
    post_put_1.setDescription("La que se avecina");
    post_put_1.setUrl("http://lqsa.es");
    post_put_1.setUsername("user_test2");

    String[] segments = location_post_1.getPath().split("/");
    String idStr = segments[segments.length-1];
    int id = Integer.parseInt(idStr);

    response = target.path("api/v1/users/user_test2/posts/"+id)
        .request()
        .put(Entity.xml(post_put_1));

    System.out.println("Estado: " + response.getStatus());

    response.close();
```

Peticion 5 [PUT] => Modificar el post del primer usuario... Estado: 200

```
// Peticion 6 [GET] => Obtener número posts de un usuario en un periodo (user_test).

    System.out.println("\nPeticion 6 [GET] Obtener número posts de un usuario en un periodo (user_test)...");

    System.out.println("Número de posts: " + target.path("api/v1/users/user_test/posts/count")
        .queryParams("from", "2017-04-20")
        .queryParams("to", "2017-05-20")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 6 [GET] Obtener número posts de un usuario en un periodo (user\_test)... Número de posts: 2

```
// Peticion 7 [DELETE] => Eliminar un post de un usuario.

    String[] segments2 = location_post_2.getPath().split("/");
    String idStr2 = segments2[segments2.length-1];
    int id2= Integer.parseInt(idStr2);

    System.out.println("\nPeticion 7 [DELETE] => Eliminar un post de un usuario...");
    response = target.path("api/v1/users/user_test/posts/"+id2)
        .request()
        .delete();

    System.out.println("Estado: " + response.getStatus());
    response.close();
```

Peticion 7 [DELETE] => Eliminar un post de un usuario... Estado: 200

```
// Peticion 8 [POST] => Agregar un amigo
    System.out.println("\nPeticion 8 [POST] => Agregar un amigo...");

    Friendship friendship_1 = new Friendship();
    friendship_1.setId_user1("user_test");
    friendship_1.setId_user2("user_test2");

    response = target.path("api/v1/users/user_test/friends/user_test2")
        .request()
        .post(Entity.xml(friendship_1));

    System.out.println("Estado: " + response.getStatus());

    Friendship friendship_2 = new Friendship();
    friendship_2.setId_user1("arecio_test");
    friendship_2.setId_user2("henry_test");

    response = target.path("api/v1/users/arecio_test/friends/henry_test")
        .request()
        .post(Entity.xml(friendship_2));

    System.out.println("Estado: " + response.getStatus());
```

Peticion 8 [POST] => Agregar un amigo... Estado: 201 Estado: 201

```
// Peticion 9 [GET] => Obtener lista de todos amigos
    System.out.println("\nPeticion 9 [GET] => Obtener lista de amigos...");

    System.out.println(target.path("api/v1/users/user_test/friends")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 9 [GET] => Obtener lista de amigos...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><friendships><friendship fr
iendshipId="22"><id_user1>user_test</id_user1><id_user2>user_test2</id_user2></fri
endship></friendships>
```

```
// Peticion 10 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo inexistente
System.out.println("\nPeticion 10 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo inexistente (debe de dar 404)...");

try {
    System.out.println(target.path("api/v1/users/user_test/friends")
        .queryParams("filter_by_name", "pepe")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
} catch (Exception error) {

    System.out.println(error);

}
}
```

Peticion 10 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo inexistente (debe de dar 404)... javax.ws.rs.NotFoundException: HTTP 404 Not Found

```
// Peticion 11 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo existente
System.out.println("\nPeticion 11 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo existente (user)...");

System.out.println(target.path("api/v1/users/user_test/friends")
    .queryParams("filter_by_name", "user")
    .request()
    .accept(MediaType.APPLICATION_XML)
    .get(String.class));
```

Peticion 11 [GET] => Obtener lista de todos amigos con filtro de nombre para amigo existente (user)...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><friendships><friendship friendshipId="22"><id_user1>user_test</id_user1><id_user2>user_test2</id_user2></friendship></friendships>
```

```
// Peticion 12 [PUT] => Modificar datos del perfil de un usuario
    System.out.println("\nPeticion 12 [PUT] => Modificar datos del perfil de un usuario...");

    User user_put_1 = new User();
    user_put_1.setUsername("user_test");
    user_put_1.setName("New user Name");
    user_put_1.setSurname("This is my new surname");

    response = target.path("api/v1/users/user_test/")
        .request()
        .put(Entity.xml(user_put_1));

    System.out.println("Estado: " + response.getStatus());

    response.close();
```

Peticion 12 [PUT] => Modificar datos del perfil de un usuario... Estado: 200

```
// Peticion 13 [GET] => Obtener lista de todos los usuarios
    System.out.println("\nPeticion 13 [GET] => Obtener lista de todos los usuarios...");

    System.out.println(target.path("api/v1/users")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 13 [GET] => Obtener lista de todos los usuarios...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><users><user><name>Antonio</name><surname>Recio</surname><username>arecio_test</username></user><user><name>CacaPepe</name><surname>Caca</surname><username>caca</username></user><user><name>Diego</name><surname>Fernández</surname><username>diegofpb</username></user><user><name>Diego</name><surname>Fernández</surname><username>diegofpb2</username></user><user><name>Amador</name><surname>Rivas</surname><username>elcuqui_test</username></user><user><name>emiluci</name><surname>ger</surname><username>emi</username></user><user><name>garrar</name><surname>Fernández</surname><username>garra</username></user><user><name>Enrique</name><surname>Pastor</surname><username>henry_test</username></user><user><name>rober daniel</name><surname>sirius</surname><username>rober</username></user><user><name>New user Name</name><surname>This is my new surname</surname><username>user_test</username></user><user><name>User2</name><surname>Test2</surname><username>user_test2</username></user></users>
```

```
// Peticion 14 [GET] => Buscar posibles amigos "Buscar a antonio"
    System.out.println("\nPeticion 14 [GET] => Buscar posibles amigos \"Buscar a antonio\"...");

    System.out.println(target.path("api/v1/users")
        .queryParams("filter_by_name", "antonio")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 14 [GET] => Buscar posibles amigos "Buscar a antonio"...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><users><user><name>Antonio</name><surname>Recio</surname><username>arecio_test</username></user><user><name>CacaPepe</name><surname>Caca</surname><username>caca</username></user><user><name>Diego</name><surname>Fernández</surname><username>diegofpb</username></user><user><name>Diego</name><surname>Fernández</surname><username>diegofpb2</username></user><user><name>Amador</name><surname>Rivas</surname><username>elcuqui_test</username></user><user><name>emiluci</name><surname>ger</surname><username>emi</username></user><user><name>garrar</name><surname>Fernández</surname><username>garra</username></user><user><name>Enrique</name><surname>Pastor</surname><username>henry_test</username></user><user><name>rober daniel</name><surname>sirius</surname><username>rober</username></user><user><name>New user Name</name><surname>This is my new surname</surname><username>user_test</username></user><user><name>User2</name><surname>Test2</surname><username>user_test2</username></user></users>
```

```
// Peticion 15 [GET] => Obtener la lista de posts publicados por amigos que contienen un determinado texto
    System.out.println("\nPeticion 15 [GET] => Obtener la lista de posts publicados por amigos (de user_test) que contienen un determinado texto...");

    System.out.println(target.path("api/v1/users/user_test/friends/posts")
        .queryParams("filter_by_text", "avecina")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 15 [GET] => Obtener la lista de posts publicados por amigos (de user\_test) que contienen un determinado texto...

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><posts><post id="43"><date_post>2017-05-01</date_post><description>La que se avecina</description><url>http://lqsa.es</url><username>user_test2</username></post></posts>
```



```
// Peticion 16 [GET] => Obtener la lista de posts publicados por amigos.
    System.out.println("\nPeticion 16 [GET] => Obtener la lista de posts publicados por amigos.");

    System.out.println(target.path("api/v1/users/henry_test/friends/posts")
        .request()
        .accept(MediaType.APPLICATION_XML)
        .get(String.class));
```

Peticion 16 [GET] => Obtener la lista de posts publicados por amigos.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><posts><post id="44"><date_post>2017-05-02</date_post><description>Mariscos Recio</description><url>http://MariscosRecio.es</url><username>arecio_test</username></post></posts>
```

```
// Peticion 17 [DELETE] => Borrar un amigo
    System.out.println("\nPeticion 17 [DELETE] => Borrar un amigo...");
    response = target.path("api/v1/users/user_test/friends/user_test2")
        .request()
        .delete();

    System.out.println("Estado: " + response.getStatus());
    response.close();
```

Peticion 17 [DELETE] => Borrar un amigo... Estado: 200