

BOOKNET - RESTFULL API

Este proyecto constituye la primera practica grupal de la asignatura Sistemas Orientados a Servicios (SOS) de la facultad ETSIINF de la Universidad Politecnica de Madrid. Este proyecto consiste en el desarrollo de un API de tipo REST para una red social llamada booknet, esta red social es un punto de encuentro entre diferentes lectores, los cuales pueden compartir los libros que han leído junto con una pequeña resena, a su vez los usuarios pueden agregarse entre sí como amigos y poder ver la actividad reciente de los mismos.

Informacion

Informacion del proyecto:

| | |
|------------|---|
| Titulación | Grado de Ingeniería Informática. Plan 09. |
| Curso | 2020/21 |
| Asignatura | Sistemas Orientados a Servicios (SOS) |
| Curso | 3º Curso |
| Semestre | 6º Semestre (Tarde) |
| Proyecto | Practica 1 - RESTFull API |

Autores

- Jesus Vallejo Collados [Matricula]
- Francisco Javier Serrano Arrese [180487]

Indice

- [BOOKNET - RESTFULL API](#)
 - [Informacion](#)
 - [Autores](#)
 - [Indice](#)
 - [Base de datos](#)
 - [Aspectos generales](#)
 - [Esquema E/R](#)
 - [Tabla USERS](#)
 - [Tabla FRIENDSHIPS](#)
 - [Tabla BOOKS](#)
 - [Tabla READ_BOOKS](#)

- [Script para crear la base de datos booknet](#)
- [Diseno de las URIs](#)
 - [Users](#)

Base de datos

Aspectos generales

Informacion de la base de datos en REST-VM:

- URI: [uri.booknet.com:3306](#)
- Nombre: [booknet](#)
- Usuario: [restuser](#)
- Contraseña: [restuser](#)

Tablas que conforman la base de datos:

- **USERS** (Tabla con los datos de los usuarios)
- **FRIENDSHIPS** (Tabla con las amistades entre usuarios)
- **BOOKS** (Tabla con la informacion de los libros)
- **READ_BOOKS** (Tabla con la informacion de las lecturas de los libros)

Esquema E/R

Esquema de Entidad Relacion empleado en el diseno de la base de datos booknet:

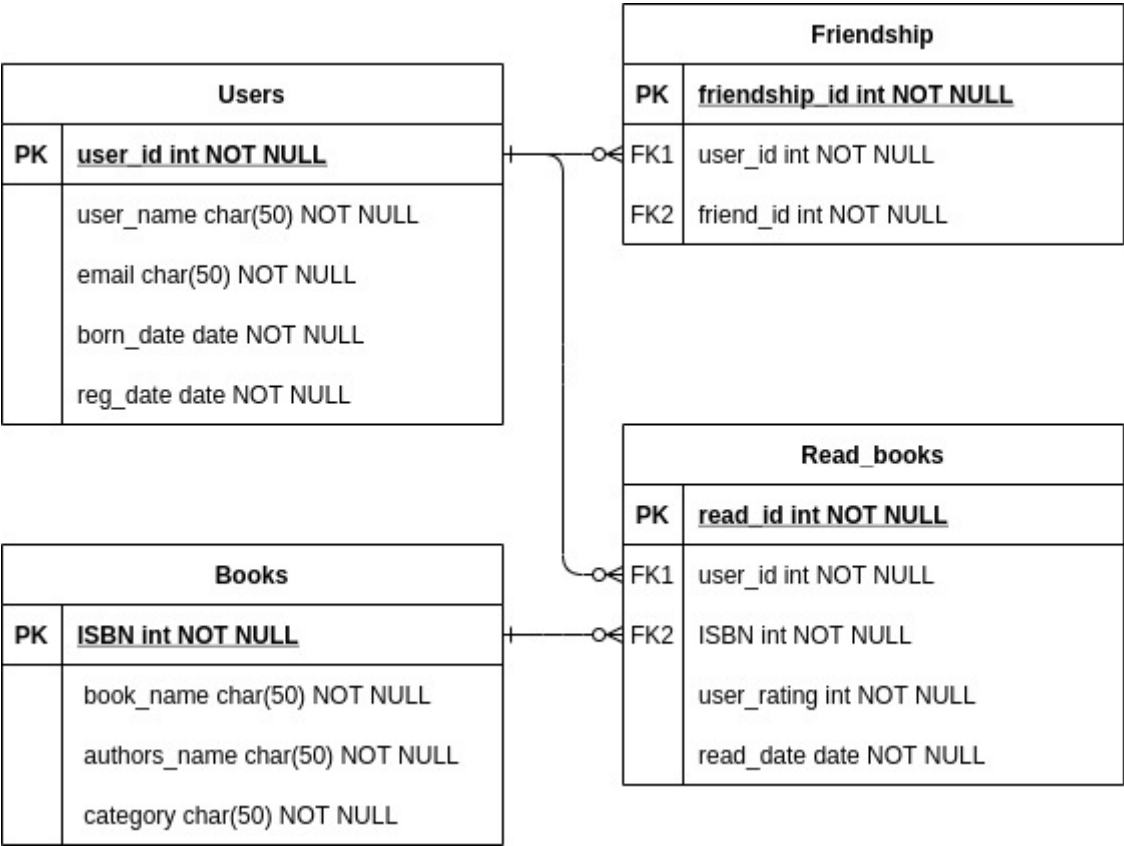


Tabla USERS

Tabla con la informacion de los usuarios de booknet.

Cada usuario cuenta con los siguientes atributos:

| user_id | user_name | email | edad |
|---------|-----------|---------|------|
| INT | VARCHAR | VARCHAR | INT |

Primary Key: user_id

Tabla FRIENDSHIPS

Tabla con la informacion de las amistades entre los usuarios

Cada amistad cuenta con los siguientes atributos:

| friendship_id | user_id | friend_id |
|---------------|---------|-----------|
| INT | INT | INT |

Primary Key: friendship_id
Foreign Key: user_id
Foreign Key: friend_id

Tabla BOOKS

Tabla con la informacion de los libros de la red booknet.

Cada libro cuenta con los siguientes atributos:

| isbn | book_name | authors_name | category |
|------|-----------|--------------|----------|
| INT | VARCHAR | VARCHAR | VARCHAR |

Primary Key: isbn

Tabla READ_BOOKS

Tabla con la informacion de una lectura de un libro realizada por un usuario

Cada lectura cuenta con los siguientes atributos:

| read_id | user_id | isbn | user_rating | read_date |
|---------|---------|------|-------------|-----------|
| INT | INT | INT | INT | INT |

Primary Key: read_id

Foreign Key: user_id

Foreign Key: isbn

Script para crear la base de datos booknet

Nota: Este script es capaz de funcionar a pesar de que la base de datos haya sido creada anteriormente.

Script booknet_db:

```
CREATE DATABASE IF NOT EXISTS `booknet` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `booknet`;
-- MySQL dump 10.13  Distrib 5.5.22, for debian-linux-gnu (i686)
--
-- Host: localhost    Database: booknet
--
-- Server version    5.5.22-0ubuntu1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
```

```

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `books`
--

DROP TABLE IF EXISTS `books`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `books` (
  `isbn` int(11) NOT NULL AUTO_INCREMENT,
  `book_name` varchar(50) NOT NULL,
  `authors_name` varchar(50) NOT NULL,
  `category` varchar(50) NOT NULL,
  PRIMARY KEY (`isbn`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `books`
--

LOCK TABLES `books` WRITE;
/*!40000 ALTER TABLE `books` DISABLE KEYS */;
INSERT INTO `books` VALUES (1,'libro1','autor1','categoria1'),
(2,'libro2','autor2','categoria1'),(3,'libro3','autor3','categoria1'),
(4,'libro4','autor4','categoria2'),(5,'libro5','autor4','categoria3');
/*!40000 ALTER TABLE `books` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `friendship`
--

DROP TABLE IF EXISTS `friendship`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `friendship` (
  `friendship_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `friend_id` int(11) NOT NULL,
  PRIMARY KEY (`friendship_id`),
  UNIQUE KEY `user_id_2` (`user_id`,`friend_id`),
  KEY `user_id` (`user_id`),
  KEY `friend_id` (`friend_id`),
  CONSTRAINT `friendship_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `friendship_ibfk_2` FOREIGN KEY (`friend_id`) REFERENCES

```

```

`users` (`user_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `friendship`
--

LOCK TABLES `friendship` WRITE;
/*!40000 ALTER TABLE `friendship` DISABLE KEYS */;
INSERT INTO `friendship` VALUES (1,1,2),(2,1,3),(3,2,1),(4,2,3),(5,3,1);
/*!40000 ALTER TABLE `friendship` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `read_books`
--

DROP TABLE IF EXISTS `read_books`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `read_books` (
  `read_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `isbn` int(11) NOT NULL,
  `user_rating` int(11) NOT NULL,
  `read_date` int(8) NOT NULL,
  PRIMARY KEY (`read_id`),
  UNIQUE KEY `read_id_2` (`user_id`,`isbn`),
  KEY `user_id` (`user_id`),
  KEY `isbn` (`isbn`),
  CONSTRAINT `read_books_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `read_books_ibfk_2` FOREIGN KEY (`isbn`) REFERENCES `books`
(`isbn`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=24 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `read_books`
--

LOCK TABLES `read_books` WRITE;
/*!40000 ALTER TABLE `read_books` DISABLE KEYS */;
INSERT INTO `read_books` VALUES (1,1,1,5,20001102),(2,1,2,7,20001106),
(3,1,3,3,19990202),(4,2,1,6,20010806),(5,2,2,2,20051203),
(6,3,4,8,20091005),(7,3,5,4,19880407);
/*!40000 ALTER TABLE `read_books` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `users`
--

```

```

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `edad` int(11) NOT NULL,
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `users`
--

LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES (1,'user1','mail1@mail.com',50),
(2,'user2','mail2@mail.com',20),(3,'user3','mail3@mail.com',22),
(4,'user4','mail4@mail.com',34);
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2021-04-21 16:01:09

```

Diseno de las URIs

Nota: El simbolo '~' representa en nuestro caso: <http://localhost8080/booknet/api>

Users

[POST] ~/users

| URI | http://localhost:8080/booknet/api/v1/users/ |
|-------------|---|
| Descripción | Permite crear un usuario, enviando una estructura JSON del tipo User. |

| URI | http://localhost:8080/booknet/api/v1/users/ |
|--------------------|---|
| Método | POST |
| Cadena de consulta | Ninguna |
| Cuerpo | POX (users/user+json) |
| Devuelve | <ul style="list-style-type: none"> • 201 : Created y cabecera Location • 406 : Not Acceptable • 415 : Unsupported Media Type |

[GET] ~/users/{user_id}

| URI | http://localhost:8080/booknet/api/v1/users/{user_id} |
|--------------------|---|
| Descripción | Devuelve en formato JSON la informacion del usuario con user_id con primary key. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none"> • filter_by_id= búsqueda por user_id |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> • 200 : OK y POX (usuarios/usuario+json) • 404 : Not Found |

[PUT] ~/users/{user_id}

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

[DELETE] ~/users/{user_id}

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

[GET] ~/users

| URI | http://localhost:8080/UPMSocial/api/v1/users/ |
|--------------------|--|
| Descripción | Devuelve una lista de usuarios de la red que contenga en su nombre el parametro user_name. En caso de no proporcionar este parametro, se devuelve toda la red de usuarios. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none"> • filter_by_text= búsqueda por nombre |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> • 200 : OK y POX (usuarios/usuario+xml) • 404 : Not Found |

[POST] ~/users/{user_id}

| URI | http://localhost:8080/booknet/api/v1/users/{user_id} |
|--------------------|---|
| Descripción | Permite anadir un libro leído por un usuario a la red, enviando una estructura JSON del tipo read_book. |
| Método | POST |
| Cadena de consulta | Ninguna |
| Cuerpo | POX (users/user+json) |

| URI | http://localhost:8080/booknet/api/v1/users/{user_id} |
|----------|---|
| Devuelve | <ul style="list-style-type: none"> • 201 : Created y cabecera Location • 406 : Not Acceptable • 415 : Unsupported Media Type |

READ BOOKS

[DELETE] ~/users/{user_id}/read_books/{isbn}

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/read_books/{isbn} |
|--------------------|--|
| Descripción | Elimina la lectura de un libro por parte de un usuario. |
| Método | DELETE |
| Cadena de consulta | Ninguna |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> • 200 : OK • 404 : Not Found • 503 : Service Unavailable |

[PUT] ~/users/{user_id}/read_books/{isbn}

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

[GET] ~/users/{user_id}/readings

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/readings |
|--------------------|---|
| Descripción | Permite consultar los ultimos libros leídos por un usuario en función de una fecha y con los criterios de paginación adecuados. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none"> filter_by_text= búsqueda por fecha |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> 200 : OK y POX (usuarios/usuario+json) 404 : Not Found |

FRIENDS

[POST] ~/users/{user_id}/friends/

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

[DELETE] ~/users/{user_id}/friends/

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friends |
|--------------------|--|
| Descripción | Elimina amistad de un usuario con otro. |
| Método | DELETE |
| Cadena de consulta | Ninguna |
| Cuerpo | Ninguno |

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friends |
|----------|--|
| Devuelve | <ul style="list-style-type: none"> • 200 : OK • 404 : Not Found • 503 : Service Unavailable |

[GET] ~/users/{user_id}/friends

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friends |
|--------------------|---|
| Descripción | Permite consultar los amigos de un usuario en funcion de un parametro nombre, en caso de que este no se introduzca, se devolvera la lista completa de amigos de un usuario. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none"> • filter_by_text= búsqueda por nombre |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> • 200 : OK y POX (usuarios/usuario+json) • 404 : Not Found |

[GET] ~/users/{user_id}/friendsreadings

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friendsreadings |
|--------------------|---|
| Descripción | Permite consultar las lecturas de libros por parte de los amigos de un usuario en funcion de la fecha de lectura. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none"> • filter_by_text= búsqueda por fecha |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none"> • 200 : OK y POX (usuarios/usuario+json) • 404 : Not Found |

[GET] ~/users/{user_id}/friendsrecomendations

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friendsrecomendations |
|-----|--|
|-----|--|

| URI | http://localhost:8080/UPMSocial/api/v1/users/{user_id}/friendsrecomendations |
|--------------------|---|
| Descripción | Permite consultar las recomendaciones de libros por parte de los amigos de un usuario en funcion de la fecha de lectura, el rating y el nombre del autor. |
| Método | GET |
| Cadena de consulta | <ul style="list-style-type: none">• filter_by_text= búsqueda por nombre |
| Cuerpo | Ninguno |
| Devuelve | <ul style="list-style-type: none">• 200 : OK y POX (usuarios/usuario+xml)• 404 : Not Found |

Testing de la API

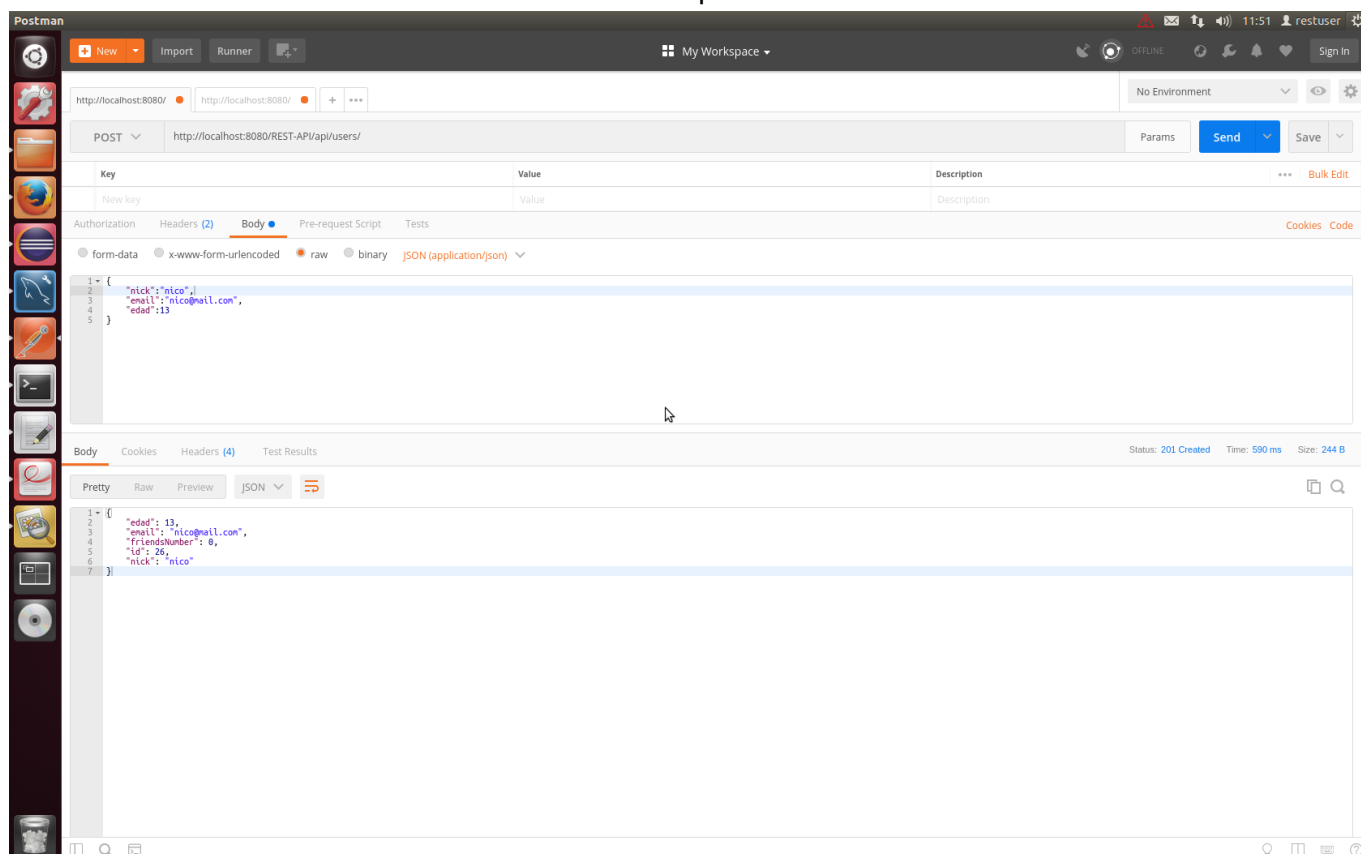
Para realizar el testing de la API RESTfull desde un cliente, se utilizo la herramienta [Postman](#). Para ello, identificabamos nuestra URI base como <http://localhost:8080/booknet/api/> y seleccionabamos el tipo de llamada que queriamos testear (PUT, POST, DELETE o GET) en nuestro caso. Seleccionando los parametros de entrada requeridos para cada llamada y con el correcto tratamiento en caso de no recibir alguno de estos parametros. Tras completar con exito el desarrollo de la API se procedio al desarrollo de un cliente java detallado mas adelante.

Pruebas Postman

USERS

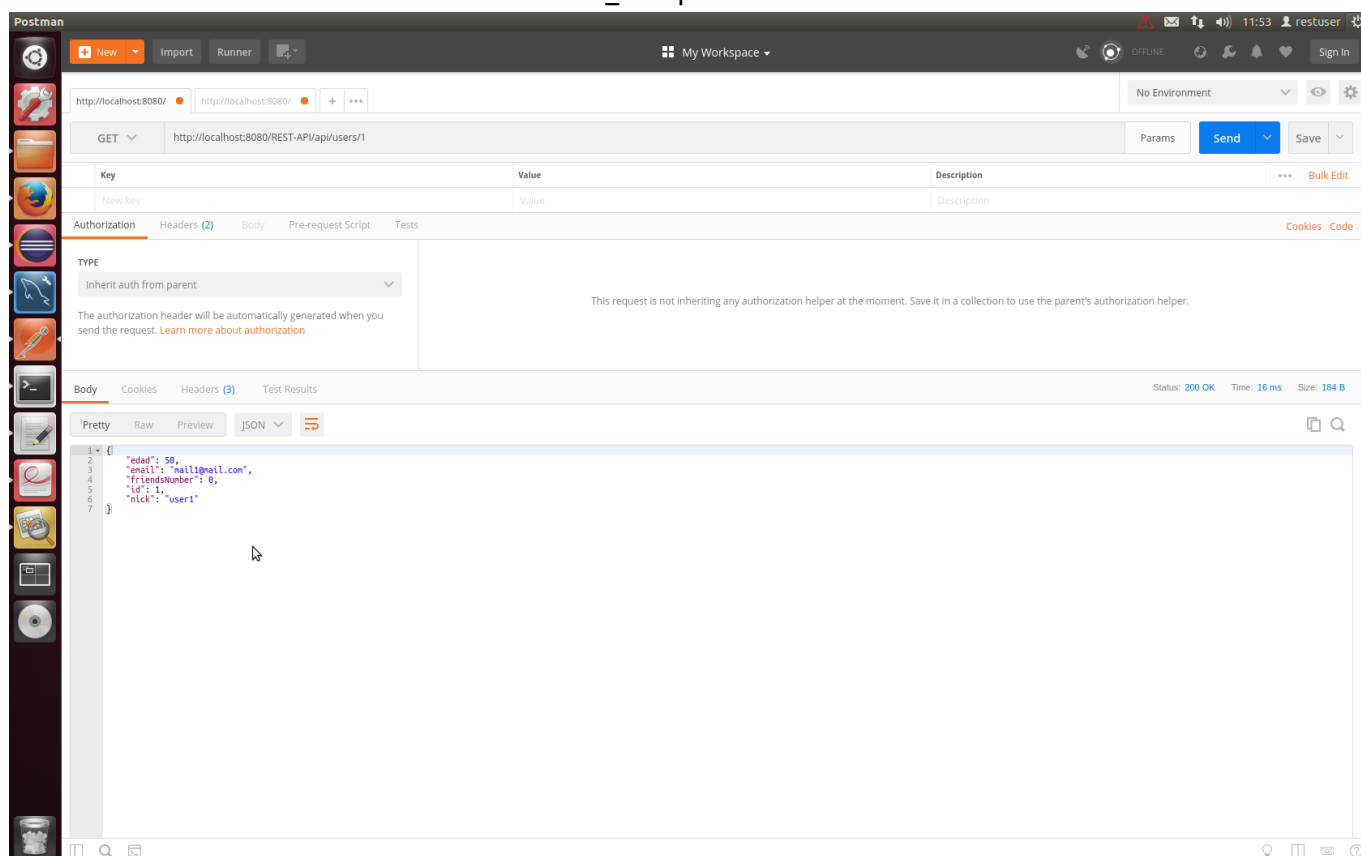
[POST] ~/users

Permite crear un usuario enviando una estructura de tipo User.



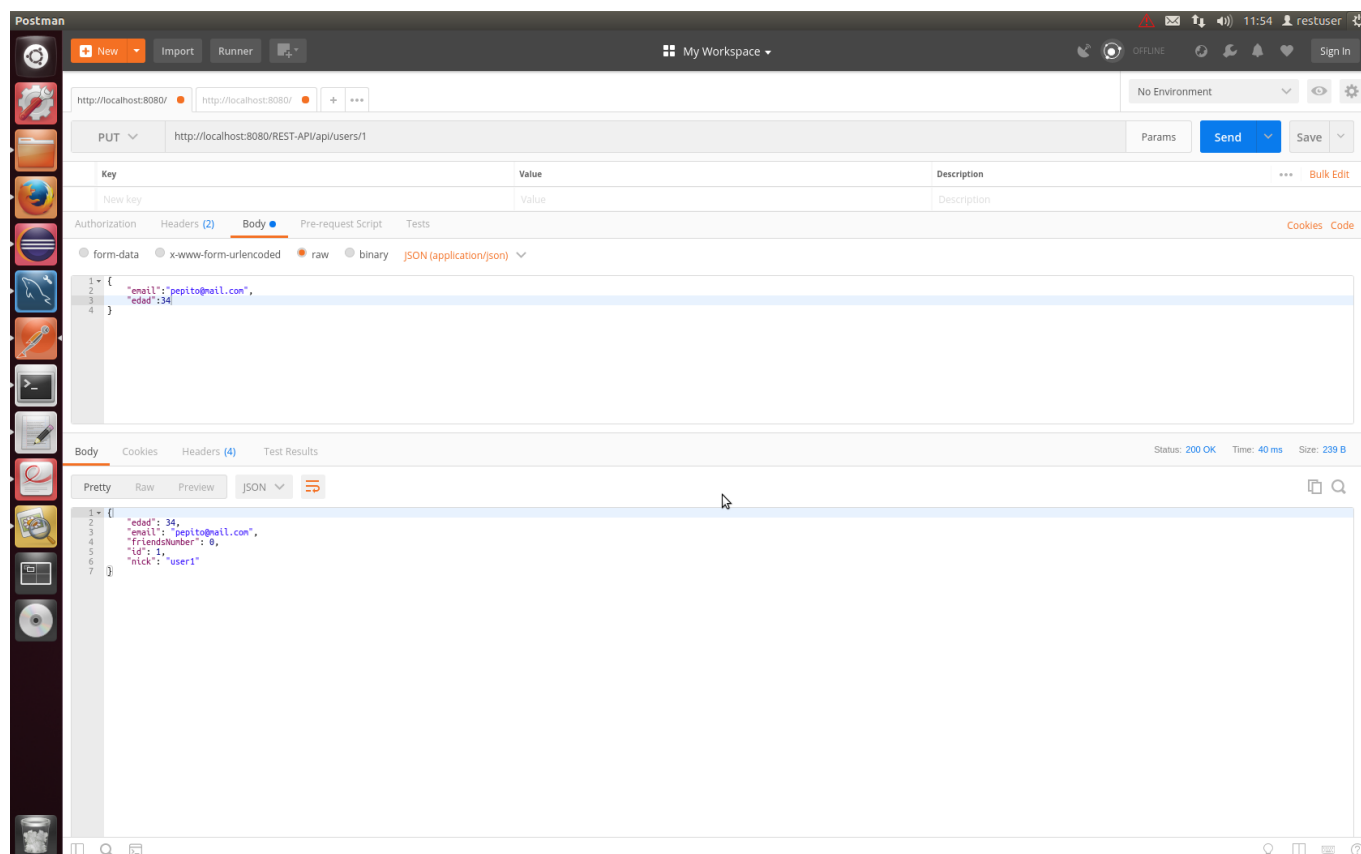
[GET] ~/users/{user_id}

Devuelve la informacion del usuario con el user_id especificado.



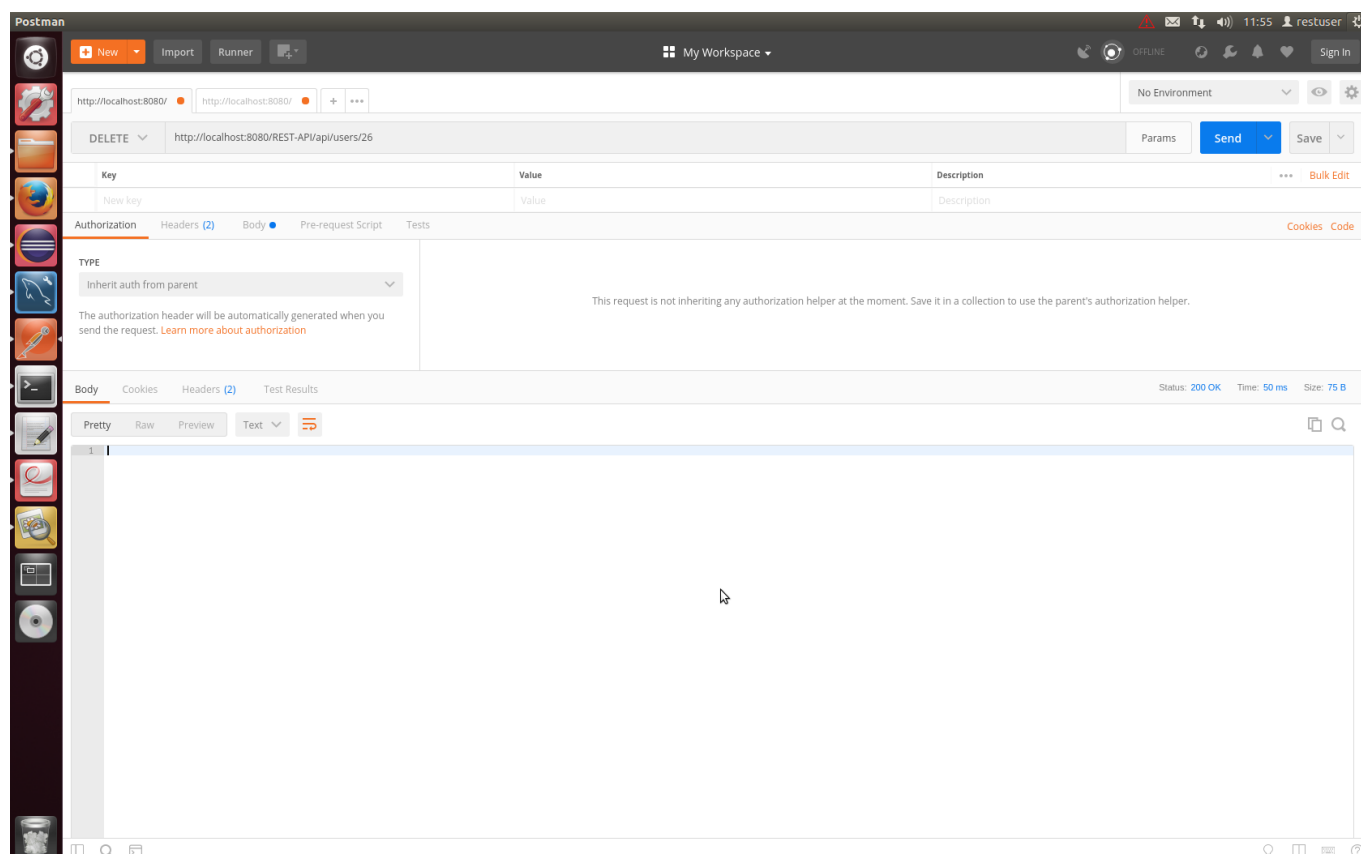
[PUT] ~/users/{user_id}

Modifica los atributos de un usuario.



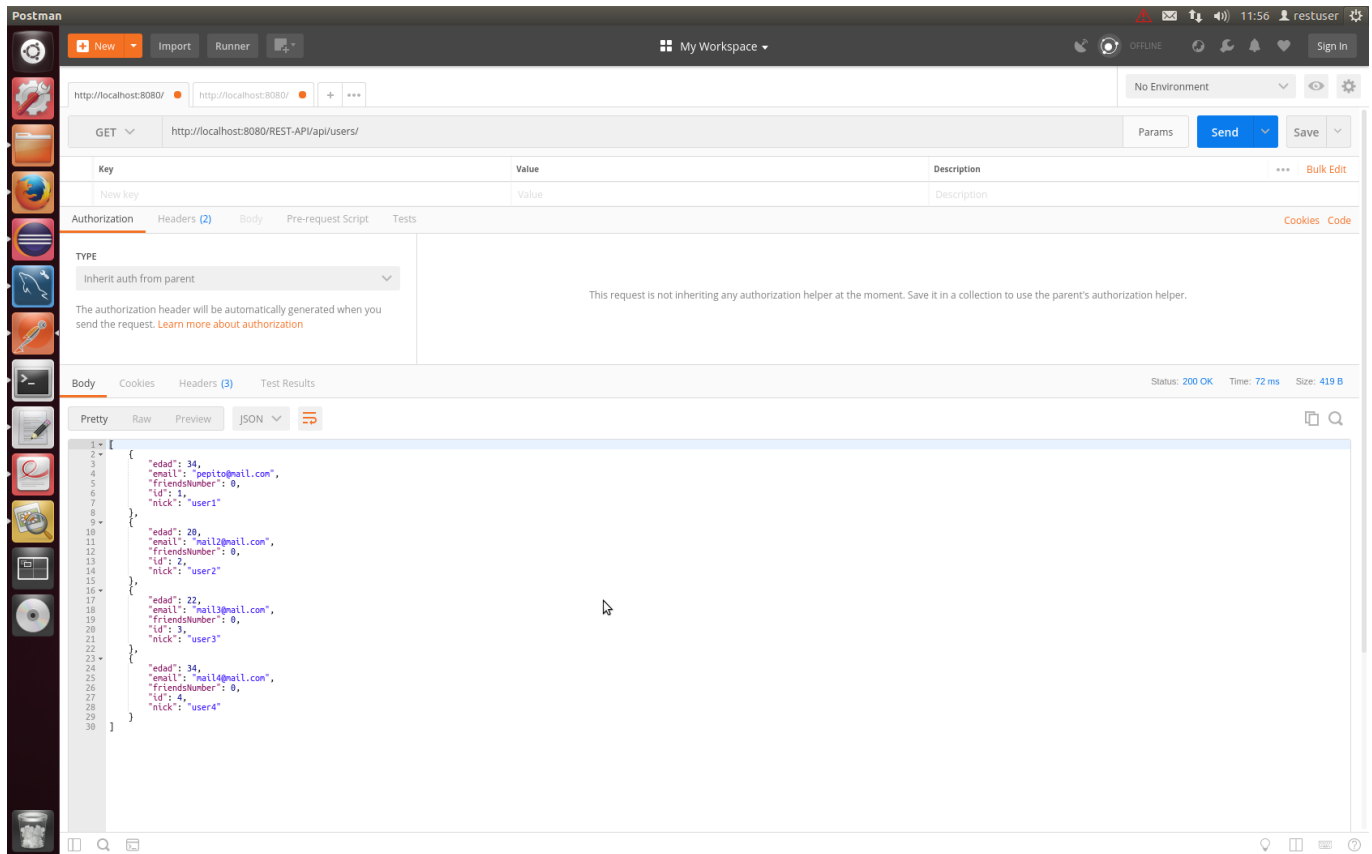
[DELETE] ~/users/{user_id}

Elimina a un usuario de la red.



[GET] ~/users

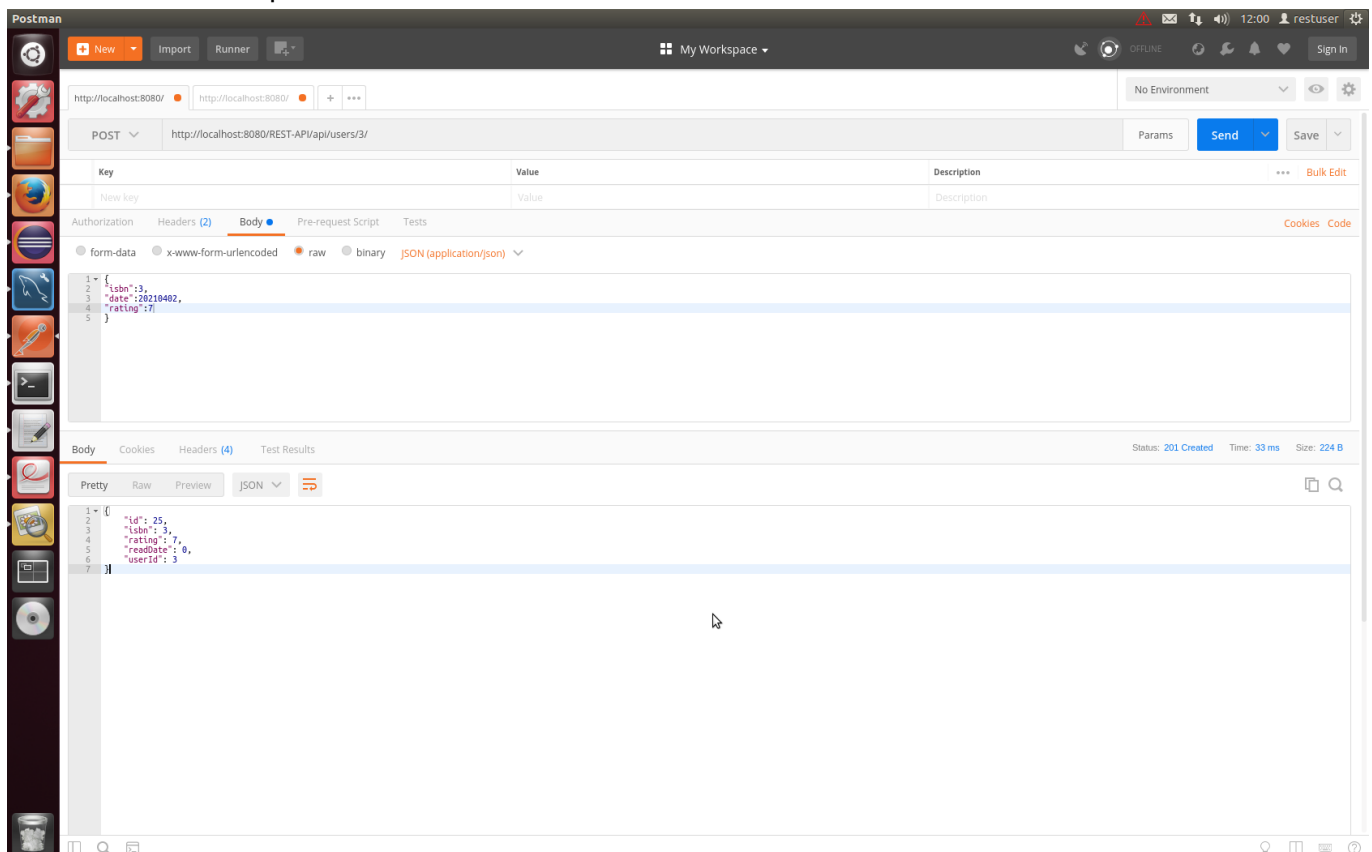
Devuelve los usuarios de la red que contenga en su nombre el parametro user_name.



READ BOOKS

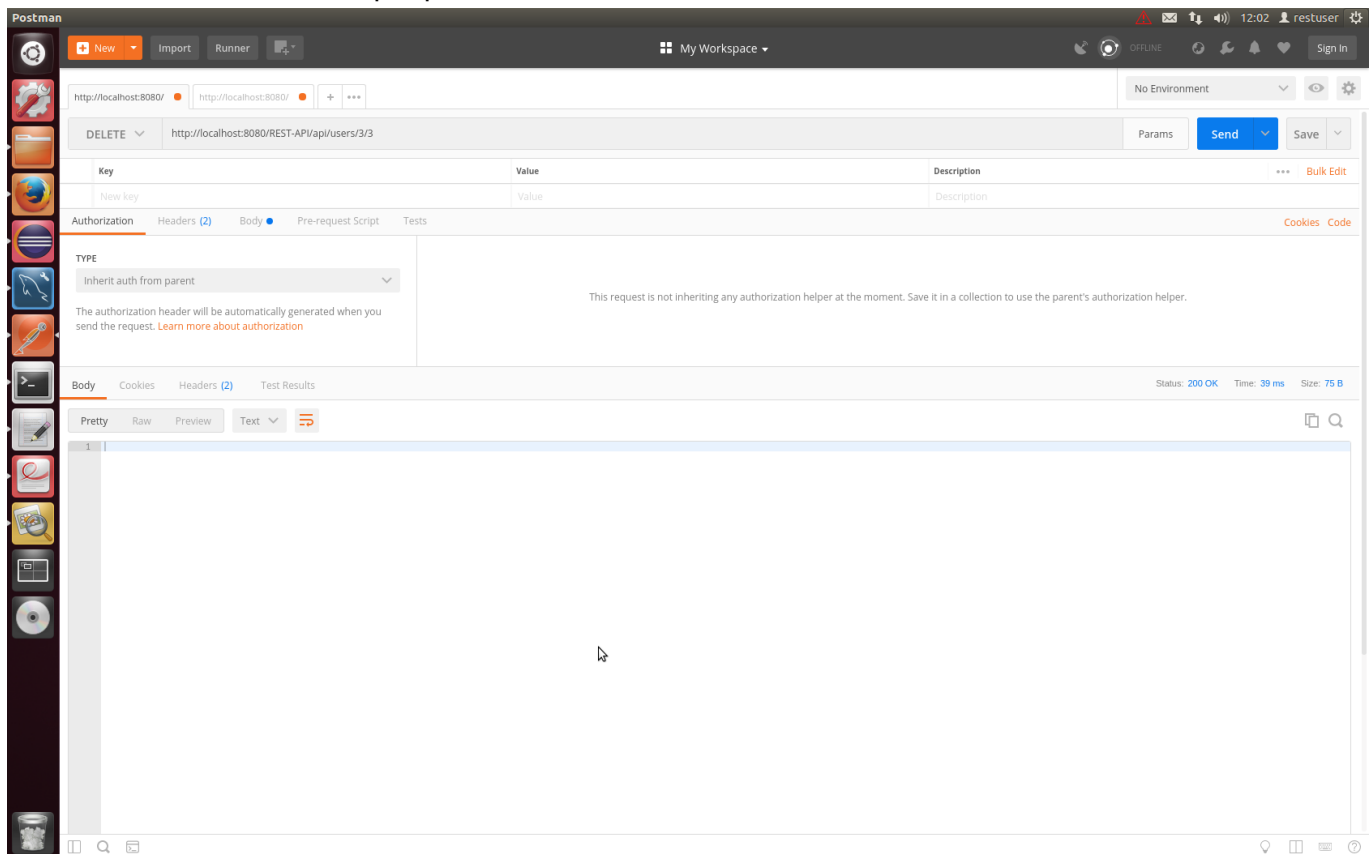
[POST] `~/users/{user_id}`

Anade un libro leído por un usuario a la red.



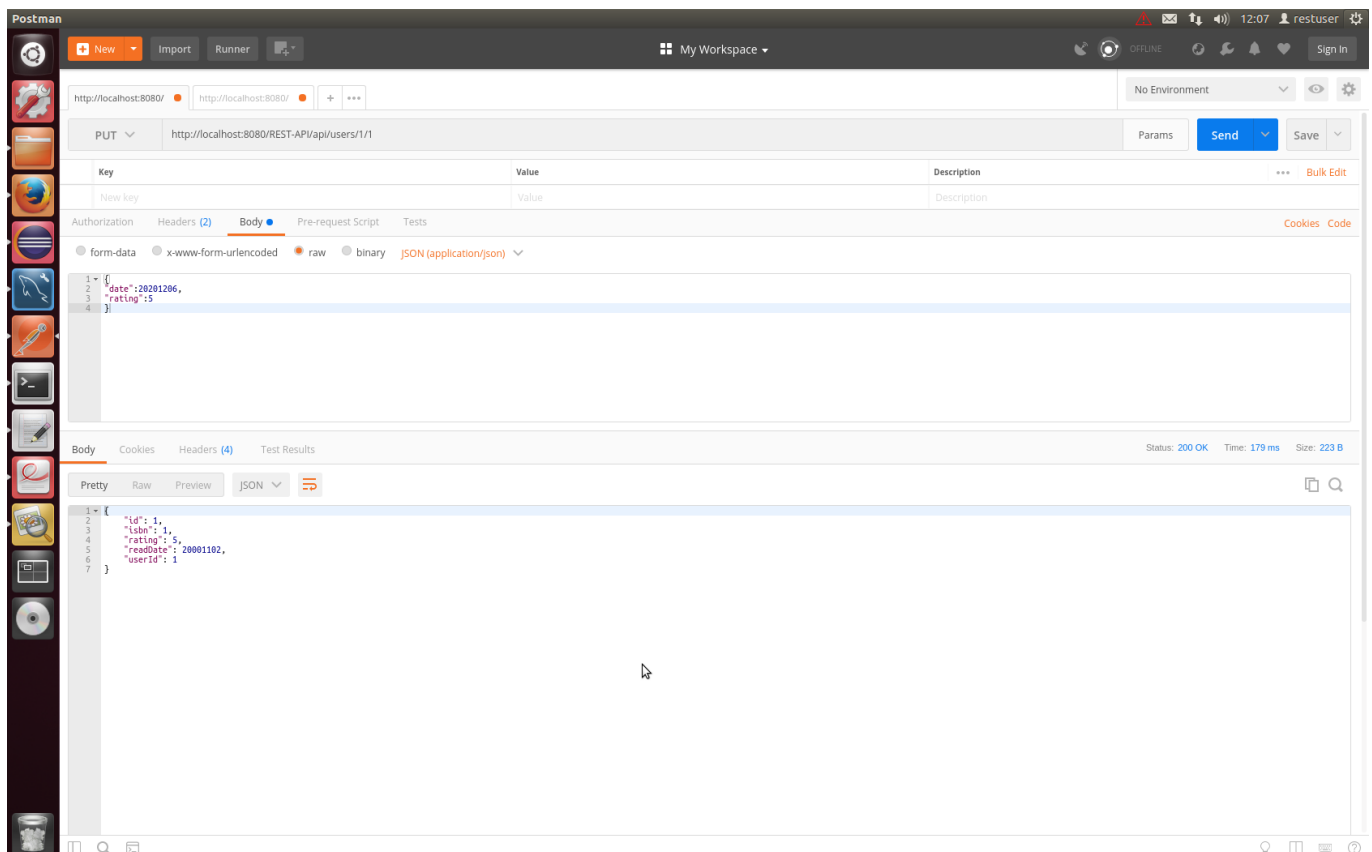
[DELETE] ~/users/{user_id}/read_books/{isbn}

Elimina la lectura de un libro por parte de un usuario.



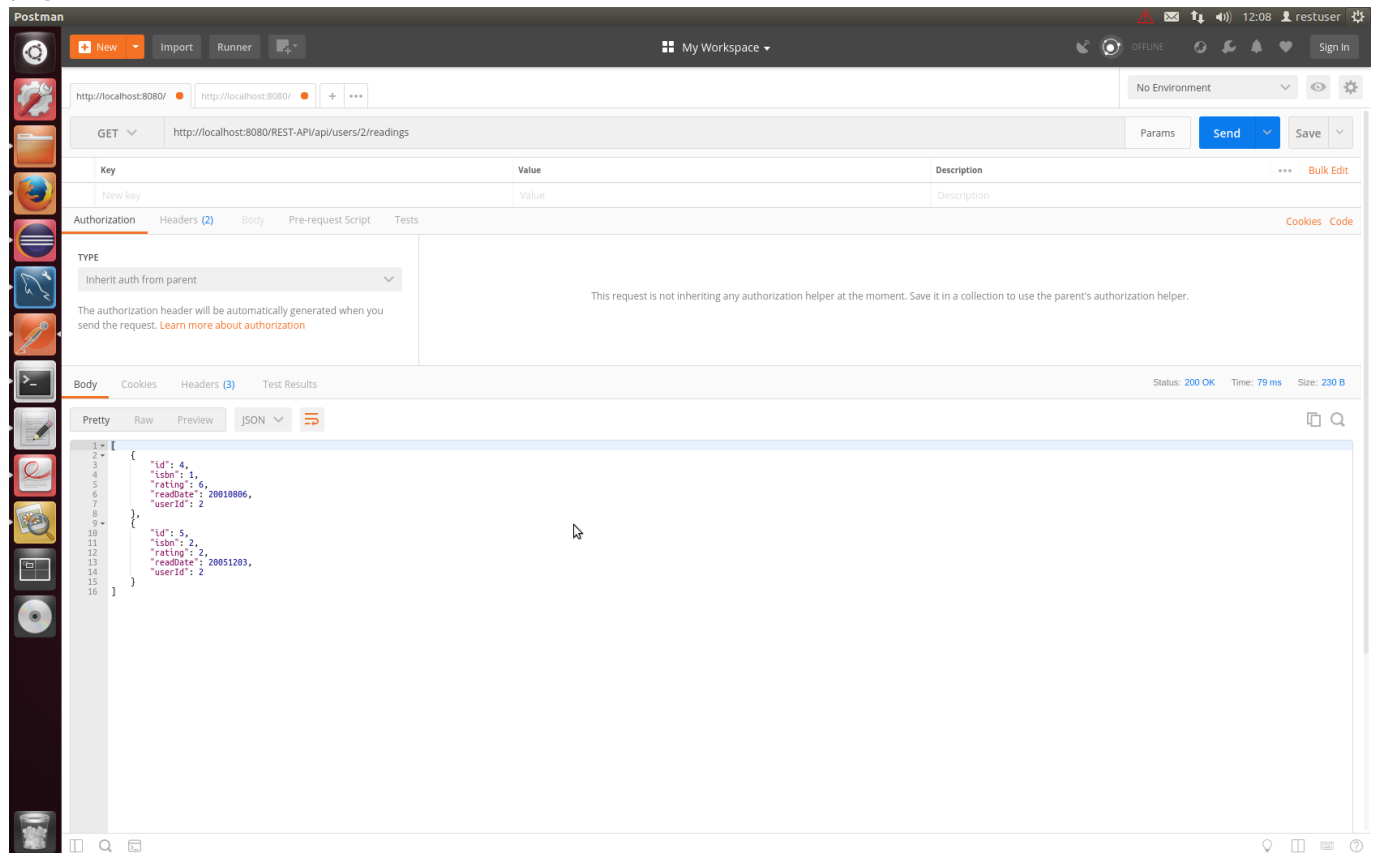
[PUT] ~/users/{user_id}/read_books/{isbn}

Modifica los atributos de la lectura de un libro.



[GET] ~/users/{user_id}/readings

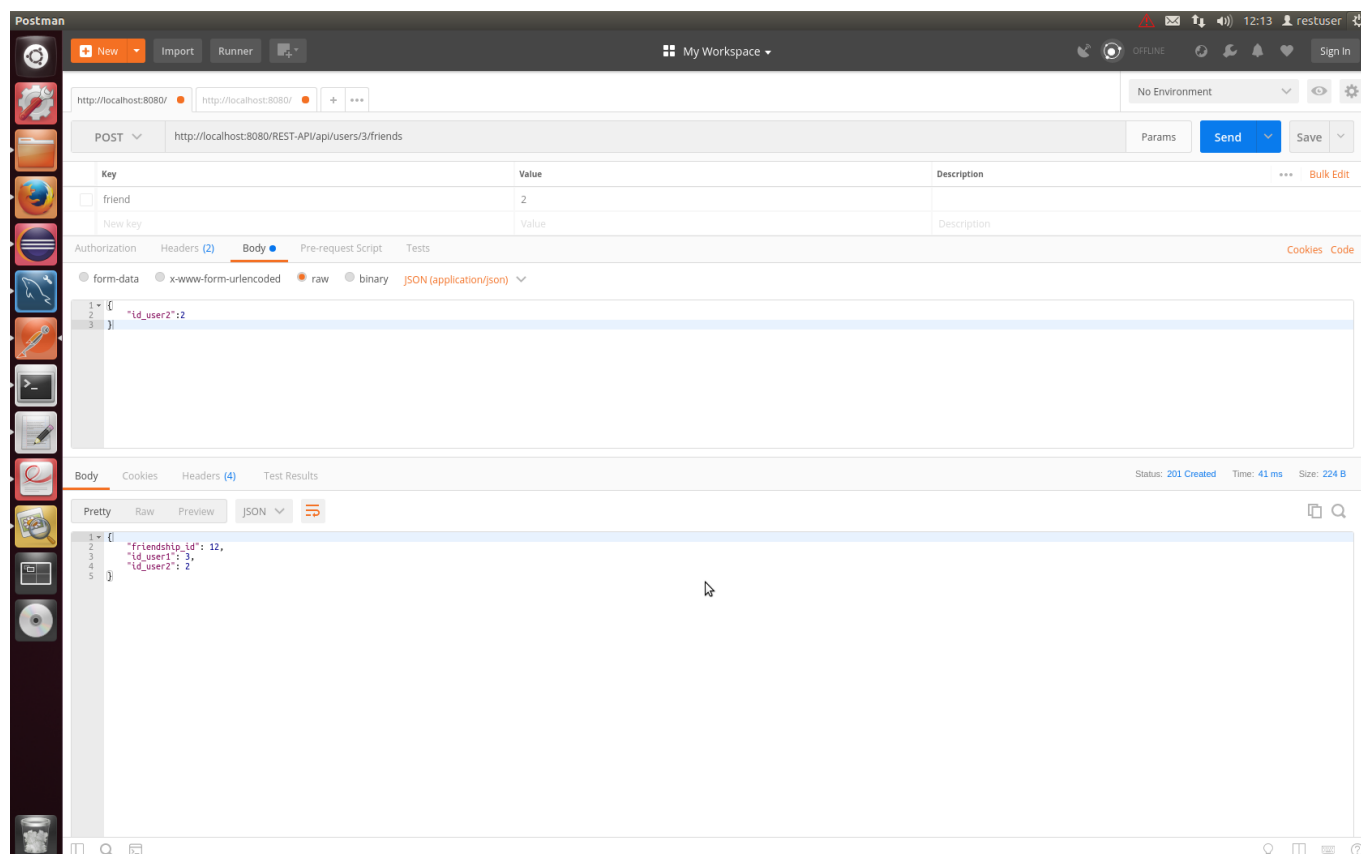
Devuelve los ultimos libros leidos por un usuario en funcion de una fecha y respetando los criterios de paginacion.



FRIENDS

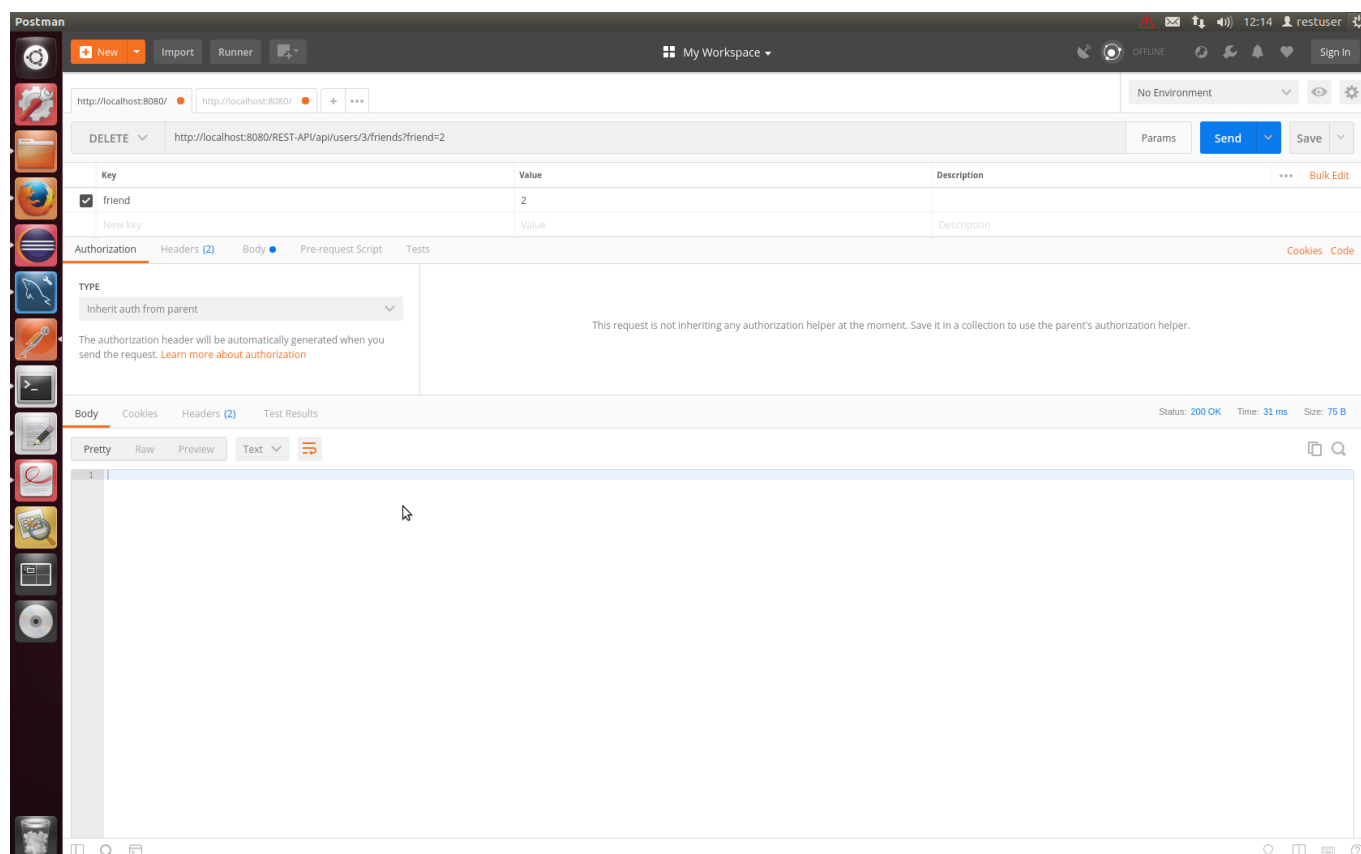
[POST] ~/users/{user_id}/friends/

Crea una amistad de un usuario con otro.



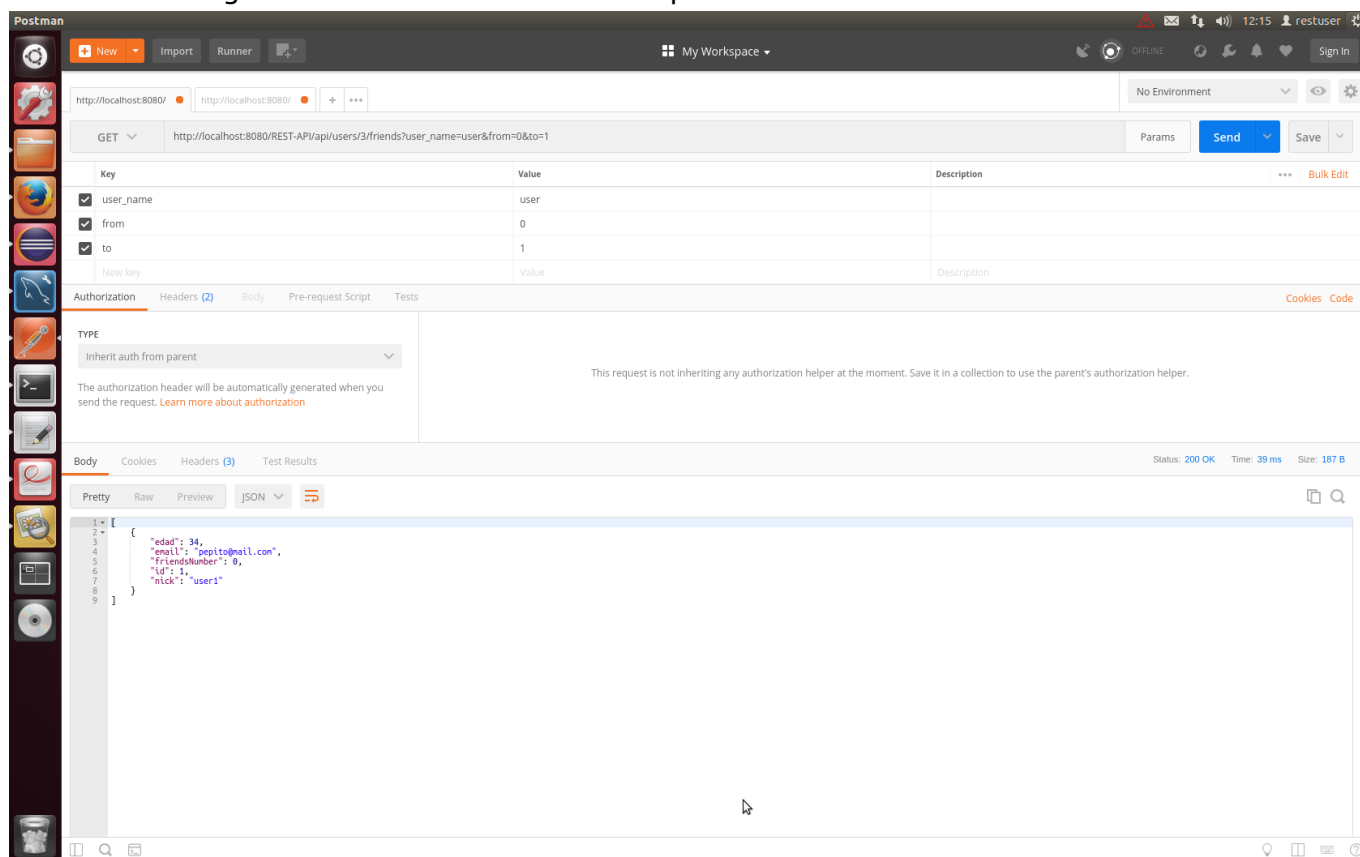
[DELETE] ~/users/{user_id}/friends/

Elimina la amistad de un usuario con otro.



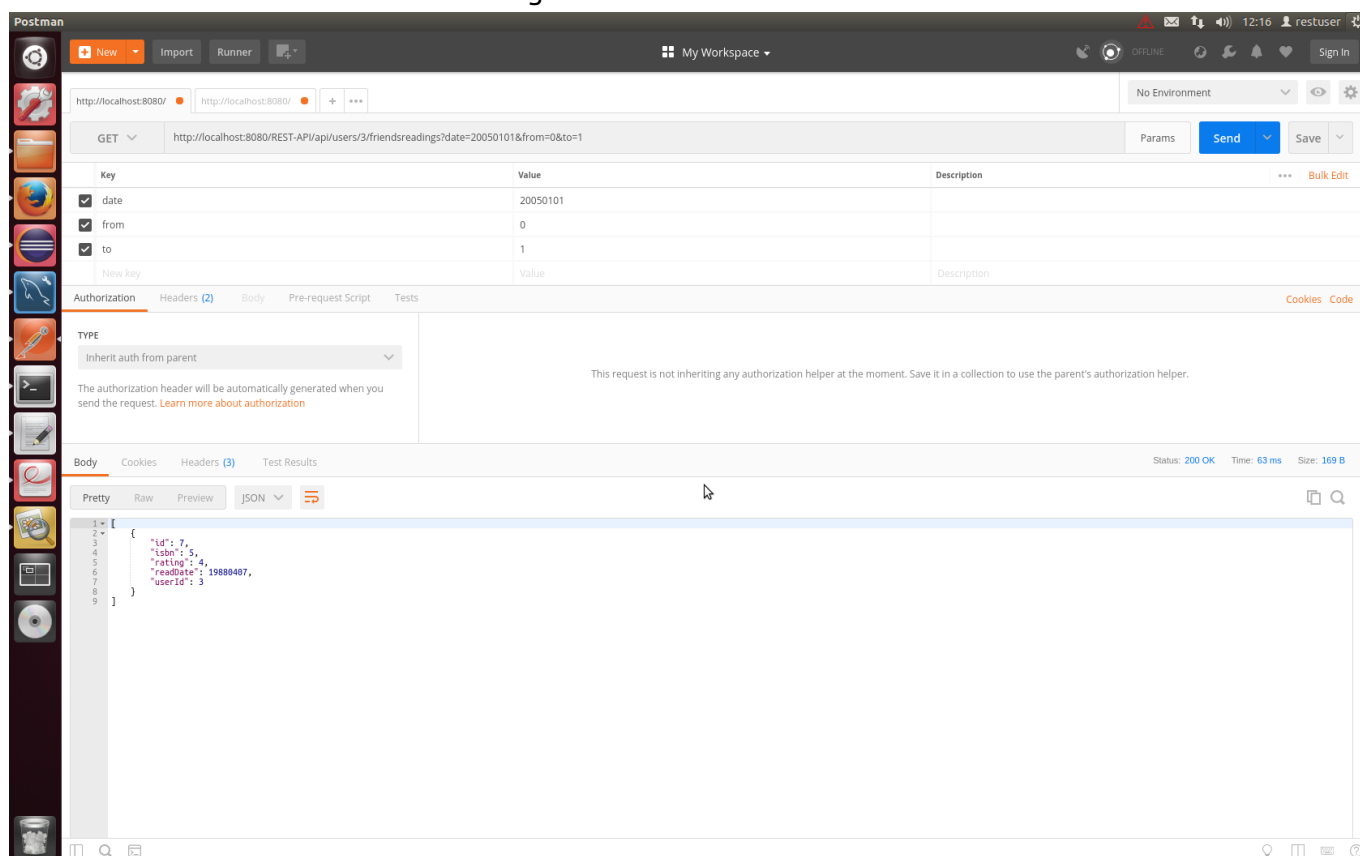
[GET] ~/users/{user_id}/friends

Devuelve los amigos de un usuario en funcion de un parametro nombre.



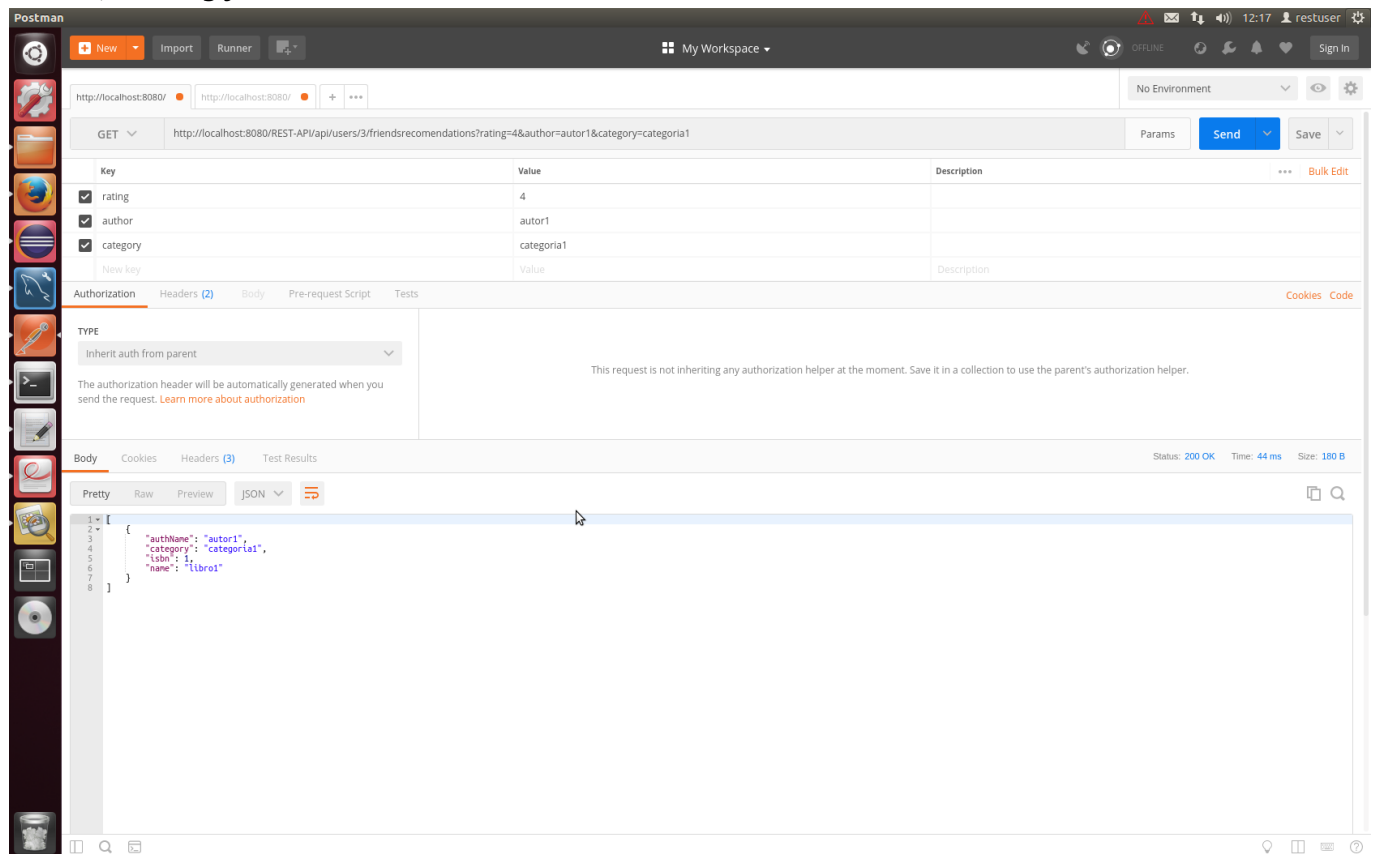
[GET] `~/users/{user_id}/friendsreadings`

Devuelve las lecturas de libros de los amigos de un usuario en funcion de la fecha de lectura.



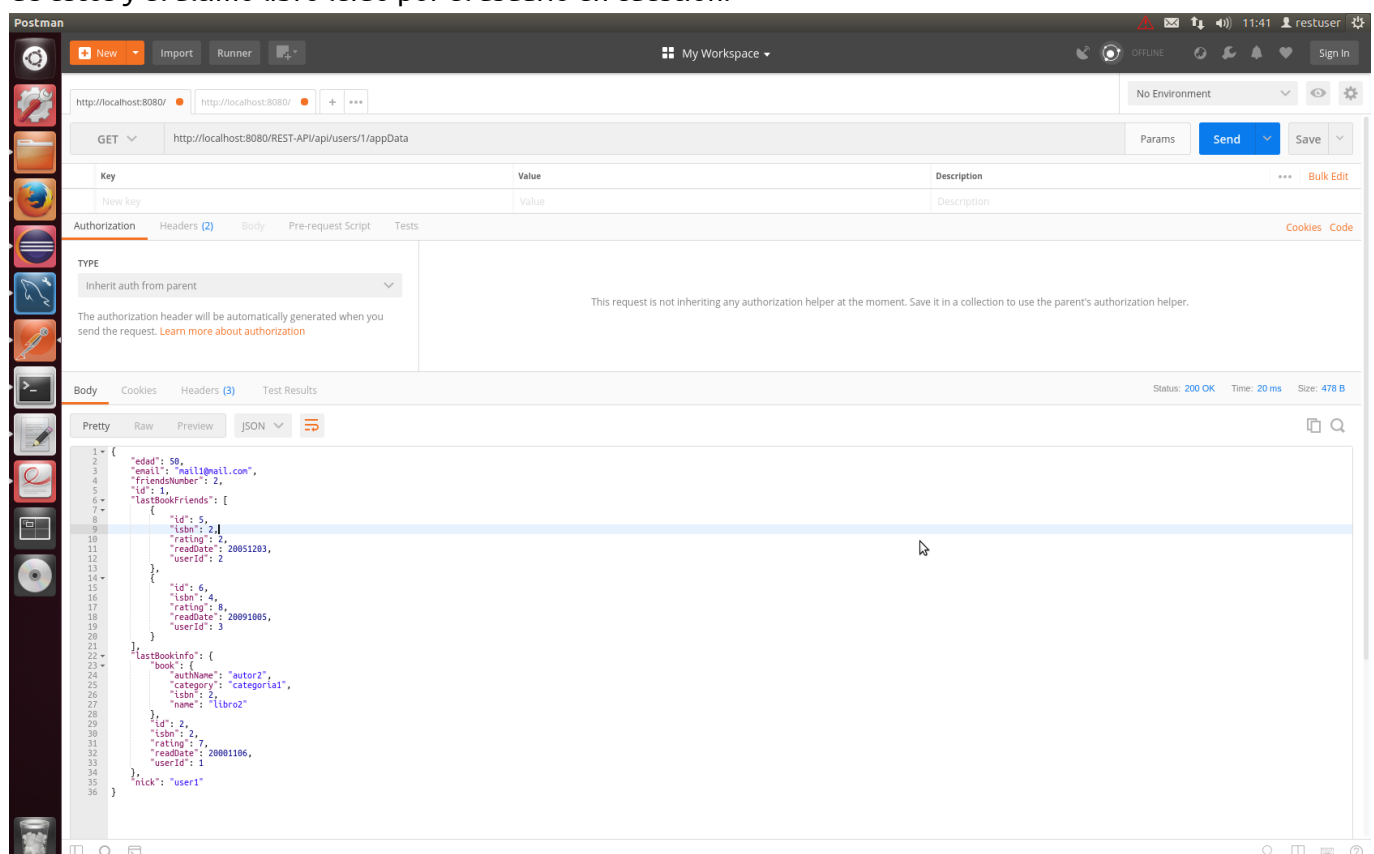
[GET] `~/users/{user_id}/friendsrecomendations`

Devuelve las recomendaciones de libros por parte de los amigos de un usuario en funcion de la fecha de lectura, el rating y el nombre del autor.



[GET] ~users/{user_id}

Devuelve la informacion completa de un usuario, incluidos los ultimos libros leidos por sus amigos, numero de estos y el ultimo libro leído por el usuario en cuestion.



Pruebas Cliente JAVA

USERS

Crear un usuario

```
User u = new client.User();
u.setNick("pepe");
u.setEdad(19);
u.setEmail("pepe@mail.com");
r =
target.path("api").path("users").request().accept(MediaType.APPLICATION_JSON).post(Entity.json(u), Response.class);
System.out.println("CREAR USER:"+r.getStatus());
System.out.println("Location: " +
r.getHeaders().get("Location").get(0).toString());
```

CREAR USER:201 Location: http://localhost:8080/booknet/api/users/1

Ver datos de un usuario

```
System.out.println("DATOS DE UN
USER:"+target.path("api").path("users/1").request()
.accept(MediaType.APPLICATION_JSON).get(String.class));
```

DATOS DE UN USER: {"edad":19,"email":"pepito@mail.com","friendsNumber":0,"id":1,"nick":"pepe"}

Actualizar datos de un usuario

```
User u2 = new client.User();
u2.setEdad(25);
u2.setEmail("pepe23@mail.com");
System.out.println("ACTUALIZAR DATOS DE UN
USER:"+target.path("api").path("users/1").request()
.accept(MediaType.APPLICATION_JSON).put(Entity.json(u2), Response.class).getStatus());
```

ACTUALIZAR DATOS DE UN USER:200

Borrar usuario

```
r = target.path("api").path("users/4").request().delete();
System.out.println("BORRAR DATOS DE UN USER:"+r.getStatus());
```

BORRAR DATOS DE UN USER:200

Obtener usuarios en la red

```
System.out.println("LISTADO
USUARIOS:"+target.path("api").path("users/").request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO USUARIOS: [{"edad":19,"email":"pepito@mail.com","friendsNumber":0,"id":1,"nick":"pepe"}, {"edad":20,"email":"mail2@mail.com","friendsNumber":0,"id":2,"nick":"user2"}]

Obtener usuarios por nombre

```
System.out.println("LISTADO USERS POR
NOMBRE:"+target.path("api").path("users/").queryParams("user_name",
"user").request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO USERS POR NOMBRE:

[{"edad":20,"email":"mail2@mail.com","friendsNumber":0,"id":2,"nick":"user2"}]

Add lectura por parte de un usuario con calificacion

```
ReadBook readBook = new ReadBook();
readBook.setIsbn(3);
readBook.setRating(8);
readBook.setReadDate(20201102); //yyyymmdd
Response r1 =
target.path("api").path("users/3").request().accept(MediaType.APPLICATION_J
SON).post(Entity.json(readBook), Response.class);
System.out.println("CREAR LECTURA:"+r1.getStatus());
System.out.println("Location: " +
r1.getHeaders().get("Location").get(0).toString());
```

CREAR LECTURA:201 Location: http://localhost:8080/booknet/api/users/read_books/1

Eliminar la lectura de un libro por parte de un usuario

```
r = target.path("api").path("users/3/3").request().delete();
System.out.println("BORRAR LECTURA DE USUARIO:"+r.getStatus());
```

BORRAR LECTURA DE USUARIO:200

Editar la lectura de un libro por parte de un usuario

```
ReadBook readBook2 = new ReadBook();
readBook2.setRating(8);
readBook2.setReadDate(20210730);//yyyymmdd
System.out.println("ACTUALIZAR DATOS DE UNA
LECTURA:"+target.path("api").path("users/1/2").request()

.accept(MediaType.APPLICATION_JSON).put(Entity.json(readBook2), Response.class).getStatus());
```

ACTUALIZAR DATOS DE UNA LECTURA:200

Consultar los libros leídos

```
System.out.println("LISTADO
LIBROS:"+target.path("api").path("users/1/readings").request()
.accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS: [{"id":4,"isbn":1,"rating":6,"readDate":20010806,"userId":2},
{"id":5,"isbn":2,"rating":2,"readDate":20051203,"userId":2}]

Consultar libros antes de cierta fecha

```
System.out.println("LISTADO LIBROS LIMITE
FECHA:"+target.path("api").path("users/1/readings").queryParams("date",20210
101).request()
.accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS LIMITE FECHA: [{"id":4,"isbn":1,"rating":6,"readDate":20010806,"userId":2}]

Consultar libros antes de cierta fecha y con paginación específica

```
System.out.println("LISTADO LIBROS LIMITE FECHA Y
CANTIDAD:"+target.path("api").path("users/1/readings").queryParams("date",20
210101).queryParams("from",1).queryParams("to",2).request()
.accept(MediaType.APPLICATION_JSON).get(String.class));
```


Nota: Al haber limitado las posiciones de 1 a 2, el resultado es el mismo que en el apartado anterior.

LISTADO LIBROS LIMITE FECHA Y CANTIDAD:

```
[{"id":4,"isbn":1,"rating":6,"readDate":20010806,"userId":2}]
```

Add amigo

```
Friendship f = new client.Friendship();
f.setId_user2(2);
r =
target.path("api").path("users/3/friends").request().accept(MediaType.APPLICATION_JSON).post(Entity.json(f), Response.class);
System.out.println("ANADIR AMIGO:"+r.getStatus());
System.out.println("Location: " +
r.getHeaders().get("Location").get(0).toString());
```

ANADIR AMIGO:2001 Location: http://localhost:8080/booknet/api/users/3/friends/12

Borrar amistad

```
r =
target.path("api").path("users/3/friends").queryParams("friend",1).request().accept(MediaType.APPLICATION_JSON).delete();
System.out.println("BORRAR DATOS DE UN USER:"+r.getStatus());
```

BORRAR DATOS DE UN USER:200

Consultar amigos

```
System.out.println("LISTADO AMIGOS:"+target.path("api").path("users/1/friends").request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO AMIGOS: [{"edad":34,"email":"pepito@mail.com","friendsNumber":0,"id":1,"nick":"user1"}]

Consultar amigos con cierto patron de nombre

```
System.out.println("LISTADO AMIGOS LIMITE NOMBRE:"+target.path("api").path("users/1/friends").queryParams("friend_name","user2").request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

Nota: Su unico amigo tiene nombre user1 y al introducir el filtro friend_name = user2, no encuentra ningun amigo que satisface la condicion LISTADO DE AMIGOS LIMITE NOMBRE: []

Consultar amigos con cierto patron de nombre y paginacion especifica

```
System.out.println("LISTADO AMIGOS LIMITE NOMBRE Y  
CANTIDAD:"+target.path("api").path("users/1/friends").queryParams("friend_name", "user").queryParams("from", 1).queryParams("to", 2).request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO AMIGOS LIMITE NOMBRE Y CANTIDAD:

[{"edad":34,"email":"pepito@mail.com","friendsNumber":0,"id":1,"nick":"user1"}]

Consulta libros leidos por amigos

```
System.out.println("LISTADO LIBROS  
AMIGOS:"+target.path("api").path("users/1/friendsreadings").request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS AMIGOS: [{"id":7,"isbn":5,"rating":4,"readDate":19880407,"userId":3}]

Consulta libros leidos por amigos previos a una fecha

```
System.out.println("LISTADO LIBROS AMIGOS LIMITE  
FECHA:"+target.path("api").path("users/1/friendsreadings").queryParams("date", 20201202).request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS AMIGOS LIMITE FECHA: [{"id":7,"isbn":5,"rating":4,"readDate":19880407,"userId":3}]

Consulta libros leidos por amigos previos a una fecha y paginacion especifica

```
System.out.println("LISTADO LIBROS AMIGOS LIMITE FECHA Y  
CANTIDAD:"+target.path("api").path("users/1/friendsreadings").queryParams("date", 20200102).queryParams("from", 1).queryParams("to", 2).request().accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS AMIGOS LIMITE FECHA Y CANTIDAD:

[{"id":7,"isbn":5,"rating":4,"readDate":19880407,"userId":3}]

Consulta libros recomendados

```
System.out.println("LISTADO LIBROS
RATING:"+target.path("api").path("users/1/friendsrecomendations").queryPara
m("rating",4).request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS RATING: [{"authName":"autor1","category":"categoria1","isbn":1,"name":"libro1"}]

Consulta libros recomendados por amigos filtrando por rating y autor

```
System.out.println("LISTADO LIBROS RATING LIMITE
AUTOR:"+target.path("api").path("users/1/friendsrecomendations").queryParam
("rating",2).queryParam("author","autor2").request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS RATING LIMITE AUTOR: [{"id":7,"isbn":5,"rating":4,"readDate":19880407,"userId":3}]

Consulta libros recomendados filtrando autores y categoria

```
System.out.println("LISTADO LIBROS AMIGOS LIMITE AUTOR Y
CATEGORIA:"+target.path("api").path("users/1/friendsrecomendations").queryP
aram("rating",4).queryParam("author","autor").queryParam("category","catego
ria1").request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO LIBROS AMIGOS LIMITE AUTOR Y CATEGORIA:

[{"id":7,"isbn":5,"rating":4,"readDate":19880407,"userId":3}]

Consulta datos completos de un usuario por la aplicacion movil

```
System.out.println("LISTADO
APPDATA:"+target.path("api").path("users/1/appData").request()
    .accept(MediaType.APPLICATION_JSON).get(String.class));
```

LISTADO APPDATA: {"edad":50,"email":"mail1@mail.com","friendsNumber":2,"id":1,"lastBookFriends": [{"id":5,"isbn":2,"rating":2,"readDate":20051203,"userId":2}, {"id":6,"isbn":4,"rating":8,"readDate":20091005,"userId":3}], "lastBookInfo":{"book": {"authName":"autor2","category":"categoria1","isbn":2,"name":"libro2"},"id":2,"isbn":2,"rating":7,"readDate":20001106,"userId":1},"nick":"user1"}