SISTEMAS DISTRIBUIDOS 2018/2019

Manuel Moreno Martinez
mmoreno2223@alumno.uned.es

TABLA DE CONTENIDO

Entorno

Primero vamos a listar el entorno en el cual se ha realizado la practica

IDE

Se ha utilizado Eclipse 4.11

Java

Se ha utilizado JDK 11 como se indicaba en la guía de la práctica.

```
faiya@faiya-MS-7A39:~$ java --version
openjdk 11.0.3 2019-04-16
OpenJDK Runtime Environment (build 11.0.3+7-Ubuntu-1ubuntu219.04.1)
OpenJDK 64-Bit Server VM (build 11.0.3+7-Ubuntu-1ubuntu219.04.1, mixed mode, sharing)
```

Sistema operativo

El sistema operativo en el cual se ha realizado la práctica es ubuntu 19.04 , disco.

```
faiya@faiya-MS-7A39:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 19.04
Release:        19.04
Codename:       disco
```

Planteamiento

En este apartado voy a comentar como he planteado diferentes partes de la práctica, y diferentes esquemas que he hecho para visualizar su funcionamiento.

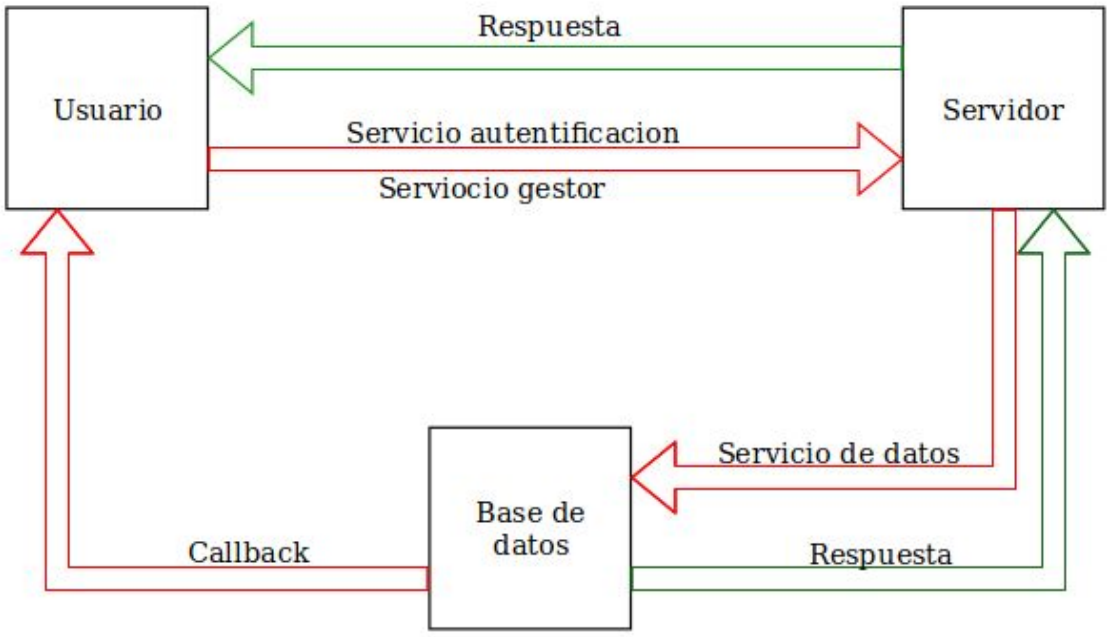
Usuario

Base de datos

La base de datos es una parte esencial de

Servicios

La comunicación entre las tres partes del programa es en general Usuario>Servidor>Basededatos, exceptuando los callback que ocurren cuando el usuario se logea, también se usa el callback para mandar a un usuario los mensajes cuando esta conectado.



Excepciones

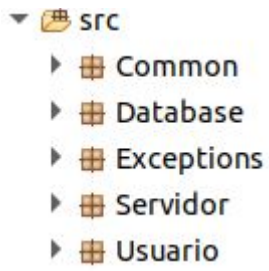
Las excepciones son una parte fundamental para asegurar el correcto funcionamiento del programa y su uso ininterrumpido. Se han creado diferentes excepciones básicas para diferentes posibles problemas que pueden surgir en el programa.

Algunas de estas excepciones se han rehusado con usos parecidos en diferentes funciones por lo que una excepción no tiene porque ser el mismo error sino un cúmulo de errores causados por un motivo parecido.

Un ejemplo de esto puede ser UserRegistered que se ha usado cuando ya hay un usuario registrado con ese nick (Al intentar registrar) o cuando no hay un usuario con ese nick (Al intentar hacer follow).

Estructura

La estructura del programa es la siguiente.



Se han creado 5 packages

Common

Es el package que contiene las clases comunes que se comparten entre las tres principales.

Contiene todas las interfaces que se han usado para hacer por ejemplo las implementaciones de los servicios.

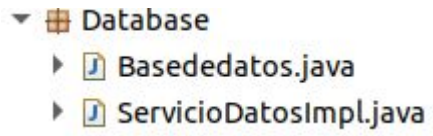
además contiene una interfaz extra que es ServidorInterface, que simplemente contiene 3 variables que son el nombre, máquina y puerto que va a utilizar el servidor.

```
public interface ServidorInterface extends Remote{
    public static final String NOMBRE_SERVICIO = "ServidorTwitter";
    public static final String MAQUINA = "localhost";
    public static final int PUERTO = 8822;

}
```

Además de las interfaces tenemos dos clases serializables , Trino y User. Trino es la clase proporcionada por el equipo docente y User es una clase que encapsula los datos de los usuarios.

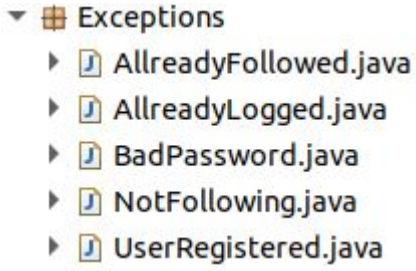
Database

En la base de datos simplemente tenemos la clase main de la base de datos y la implementación del servicio de datos que permite la comunicación con la base de datos.



Exceptions

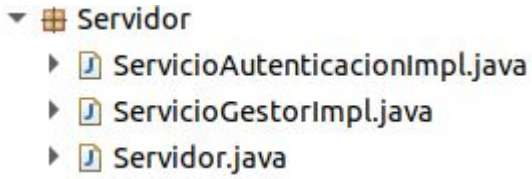
En el paquete exceptions nos encontramos todas las excepciones propias que se utilizan.



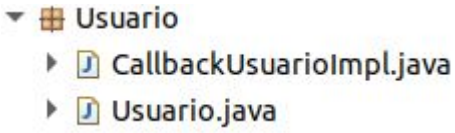
Cada una de estas excepciones simplemente es una extensión de la clase Exception de forma muy básica.

```
package Exceptions;

public class AllreadyLogged extends Exception{
    private static final long serialVersionUID = 1L;

    public AllreadyLogged(String s)
    {
        super(s);
    }
}
```

Servidor



Usuario



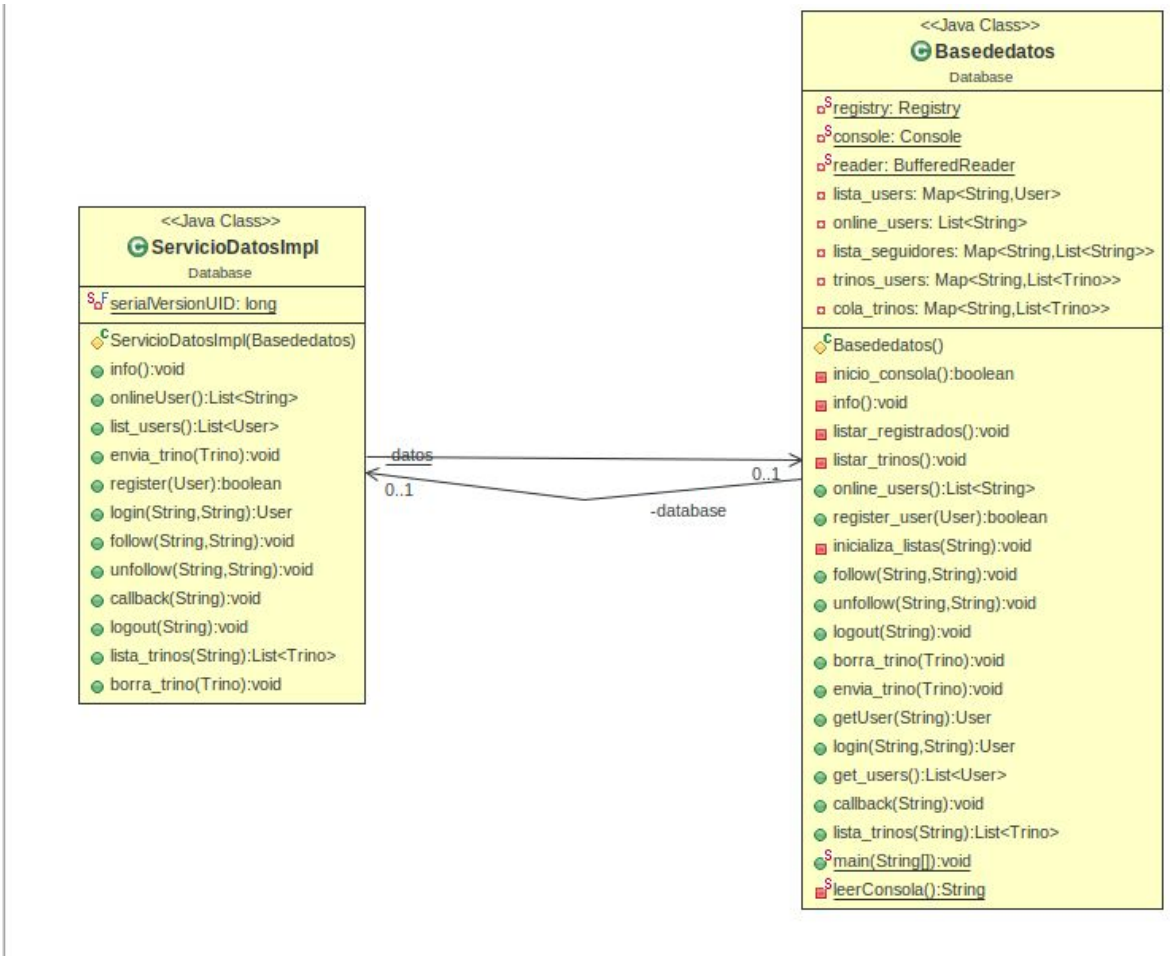
Callback

El callback, es la herramienta que permite que cuando un usuario se conecte, reciba los tweets que tiene pendientes de los usuarios a lo que sigue. Además el callback se utiliza para mandar los tweets a los usuarios conectados, ya que permite la conexión Basededatos->Usuario.

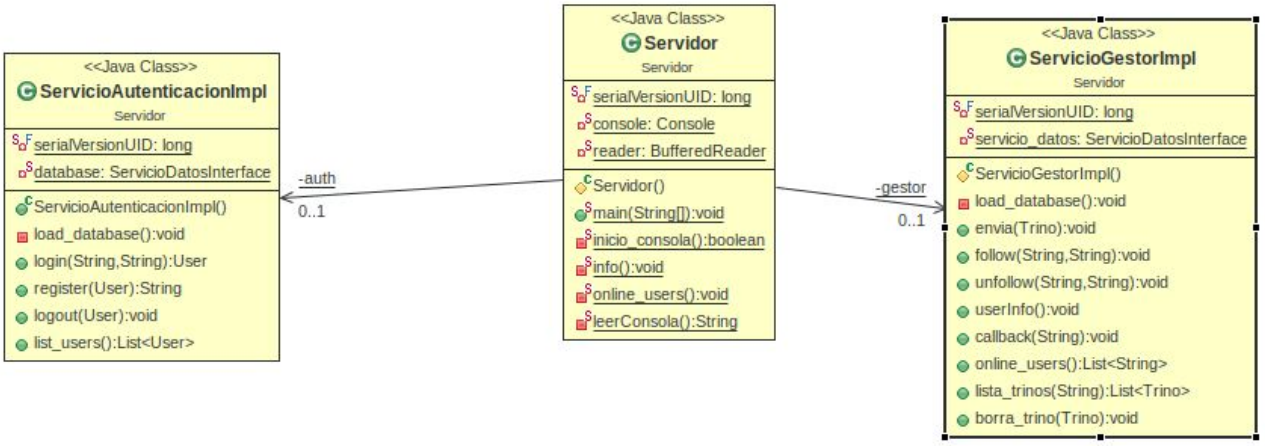
Diagramas

En este apartado vamos a ver los diagramas de clases de cada parte del código y sus relaciones.

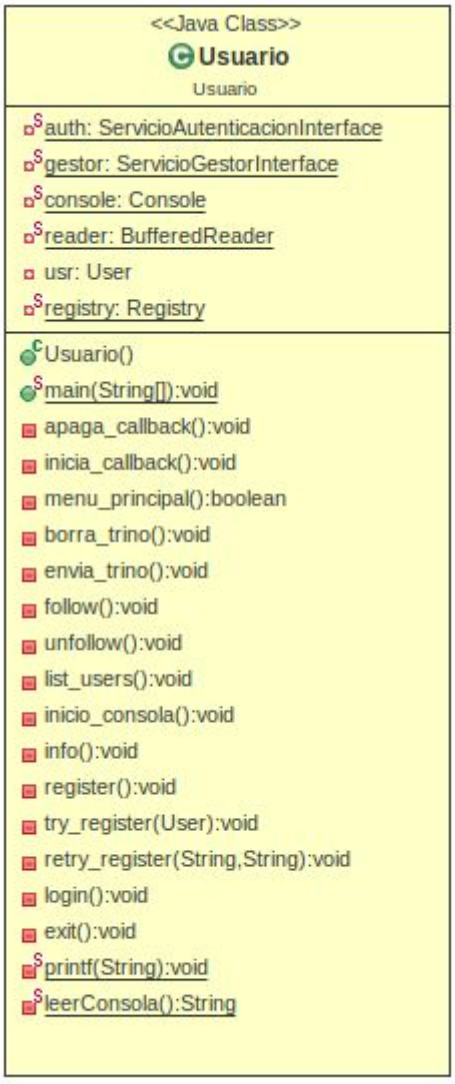
Base de datos

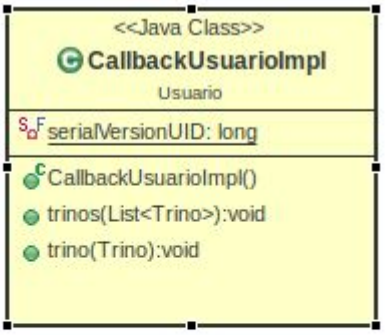


Servidor


<<Java Class>>
ServicioAutenticacionImpl
Servidor
serialVersionUID: long
database: ServicioDatosInterface
ServicioAutenticacionImpl()
load_database():void
login(String,String):User
register(User):String
logout(User):void
list_users():List<User>
-auth
0..1
<<Java Class>>
Servidor
Servidor
serialVersionUID: long
console: Console
reader: BufferedReader
Servidor()
main(String[]):void
inicio_consola():boolean
info():void
online_users():void
leerConsola():String
-gestor
0..1
<<Java Class>>
ServicioGestorImpl
Servidor
serialVersionUID: long
servicio_datos: ServicioDatosInterface
ServicioGestorImpl()
load_database():void
envia(Trino):void
follow(String,String):void
unfollow(String,String):void
userInfo():void
callback(String):void
online_users():List<String>
lista_trinos(String):List<Trino>
borra_trino(Trino):void

Usuario





Código y Funcionamiento

User

Esta parte de la memoria va a explicar el código y funcionamiento desde el punto de vista de un usuario utilizando el programa.

Menu de inicio

```
Elija la operación:
1.- Registrar un nuevo usuario.
2.- Hacer login.
3.- Salir
```

Es lo primero que nos vamos a encontrar cuando un usuario accede al programa.
Solamente nos permite acceder al programa como un usuario, crear uno nuevo o salirnos.

Register

Para registrarse simplemente seleccionamos la opción 1.

Después introducimos nuestro nombre, un nick y una contraseña y ya habíamos registrado a nuestro usuario.

Es posible que aparezcan excepciones que se han visto anteriormente, como intentar registrar un usuario con un nick que ya existe.

```
Elija la operación:
1.- Registrar un nuevo usuario.
2.- Hacer login.
3.- Salir
1
Introduzca su nombre:
manuel
Introduzca su nick:
faiya
Introduzca su contraseña:
123
usuario registrado con exito
```

Login

Prácticamente igual que el anterior, aunque solo hay que introducir el nick y la contraseña para poder logear, también puede producir excepciones.

Exit

Simplemente finaliza el programa.

Menu principal

```
Elija la operación:
1.- Información del Usuario.
2.- Enviar Trino.
3.- Listar Usuarios del Sistema.
4.- Seguir a
5.- Dejar de seguir a.
6.- Borrar trino a los usuarios que todavía no lo han recibido
7.- Logout.
```

En este menú tenemos las 7 opciones que se nos pedía en el enunciado.

1. Informacion

   Eligiendo la opción uno, se nos muestra por pantalla la url de los servicios que tiene. En este caso podemos ver que solo esta el callback de el usuario registrado con nick faiya.

   ```
   1
   Servicios rmi del cliente:
   rmi://localhost:8821/Callback/faiya
   ```

2. Enviar trino

   La segunda opción nos permite enviar un trino a los usuarios que nos siguen.

   ```
   2
   Introduzca el mensaje que quiere enviar:
   Trino de prueba
   ```

   Después de enviarlo les aparecerá a los seguidores en tiempo real con este formato.

   ```
   > faiya# Trino de prueba
   ```

   Faiya es el nick del usuario que ha enviado el trino y "Trino de prueba" es el contenido.

3. Listar usuarios

   Simplemente nos muestra los nicks de los usuarios registrados

   ```
   3
   @tt, @faiya
   ```

4. Seguir a

   Nos permite seguir a un usuario introduciendo su nick. Nos puede dar errores si intentamos seguir un nick que no está registrado.

```
4
Introduzca el nick del usuario al que quiere seguir:
test
No existe un usuario con nick: "test"

Intentelo de nuevo
```

Es posible seguirte a ti mismo lo cual puede resultar raro, pero como no venía especificado he tomado esa decisión. Si no te sigues a ti mismo no te llegan tus propios trinos.

5. Dejar de seguir

   Simplemente deja de seguir a la persona que pongas el nick.

6. Borrar trino

   Este es el método que puede ser más complicado.
   te permite elegir un trino de todos los que has enviado.
   Después borra ese trino de la cola de pendientes de los usuarios que te siguen.

```
6
Por favor eliga uno de los siguientes trinos:
0.- Test 1
1.- Test 2
1
Se ha borrado con exito el trino Test 2
```

7. Logout

   Te desconecta del servidor apaga tu callback y sale.

Admin

Servidor

El menú que nos encontramos en el servidor es el siguiente. Tiene las tres opciones que se pedían en la guia.

```
Iniciado servidor twitter
Elija la operación:
1.- Información del Servidor.
2.- Listar Usuarios Logeados.
3.- Salir.
```

```
1
Servicios rmi del servidor:
ServicioAutenticacion    :       rmi://localhost:8822/Auth_service
ServicioGestor           :       rmi://localhost:8822/gestor_service
```

La primera opción nos da la información del servidor.

```
2

[Info] Los usuarios conectados son los siguientes:
@faiya
```

La segunda opción nos muestra los usuarios conectados (no los registrados).
Y la tercera apaga los servicios del servidor cierra su registro y sale.

Database

El terminal de la base de datos es el único que puede ser algo más distinto.
Lo primero que vemos es el menu igual que los demás.

```
Iniciado base de datos
Elija la operación:
1.- Información de la Base de Datos.
2.- Listar Usuarios Registrados.
3.- Listar Trinos
4.- Salir.
```

Pero sin tener que hacer nada, es en este terminal en el cual vamos a ir viendo información de lo que está pasando.
Como por ejemplo

```
[Info] Se ha registrado con exito al usuario: @faiya
[Info] trino recibido
[Info] Common.Trino@Trino de prueba|faiya|1563024967795|
[Info] Se ha registrado con exito al usuario: @tt
[Info] Usuario @tt ha seguido a @faiya correctamente.
[Info] trino recibido
[Info] Common.Trino@Trino de prueba|faiya|1563025040387|
[Error] Se ha intentado seguir a un usuario que no existe
[Info] Intentando borrar trino Common.Trino@Trino de prueba|faiya|1563025040387|
 de @faiya para sus seguidores
[Info] Cola de trinos del seguidor @tt:[]
[Info] El trino no esta en la lista.
[Info] trino recibido
[Info] Common.Trino@Test 1|tt|1563025467818|
[Info] trino recibido
[Info] Common.Trino@Test 2|tt|1563025472322|
[Info] Intentando borrar trino Common.Trino@Test 2|tt|1563025472322| de @tt para
```

Como se puede ver nuestra información y errores que van ocurriendo cuando los usuarios utilizan el programa. Esta parte se podría haber implementado en la zona del servidor pero por comodidad se ha dejado en la parte de la base de datos.

```
1
Servicios rmi de la base de datos:
rmi://localhost:8823/Database
```

La opción 1 nos permite ver los servicios de la base de datos, con su url.

```
2

@tt, @faiya
```

La opción 2 nos muestra los usuarios registrados

```
3

[Info] Mostrando trinos.
Trinos de @tt:
        >Test 1
        >Test 2
Trinos de @faiya:
        >Trino de prueba
        >Trino de prueba
```

La opción 3 nos muestra los trinos de todos los usuarios.

La opción 4 apaga los servicios y sale.

Ejecución

Para ejecutarlo, se han creado 4 bash scripts, Database.sh, Servidor.sh, Usuario.sh y Start.sh.
Se recomienda utilizar Start.sh que crea la base de datos, el servidor y dos usuarios.
Si se ejecutan por separado es necesario ejecutar la base de datos primero , después el servidor y luego los usuarios.

Si se ejecuta en otro orden, el servidor y el usuario esperaran lo que necesitan haciendo un retry cada 5 segundos un máximo de 10 veces.
Después de esas 10 veces saltará un error no controlado.

```
[Error] No se ha podido conectar al servidor, reintentando en 5 segundos . . . . .
[Info] Intento 1 de 10.
[Error] No se ha podido conectar al servidor, reintentando en 5 segundos . . . . .
[Info] Intento 2 de 10.
[Error] No se ha podido conectar al servidor, reintentando en 5 segundos . . . . .
[Info] Intento 3 de 10.
[Error] No se ha podido conectar al servidor, reintentando en 5 segundos . . . . .
[Info] Intento 4 de 10.
[Error] No se ha podido conectar al servidor, reintentando en 5 segundos . .
```

Mejoras

Base de datos

La base de datos implementada con estructuras de java debería de sustituirse por una base de datos relacional como puede ser MySql o PostgresQL incluso una base de datos mejor aun por un DBaaS (Database-as-a-service) que es una base de datos en la nube, que nos permita redundancia y escalabilidad.

Seguridad

El programa básicamente tiene una seguridad nula, la única seguridad que hay es un pequeño control de excepciones para evitar que la aleatoriedad de los usuarios. Para hacer un programa decente habría que poner una seguridad en condiciones, prohibiendo por ejemplo el acceso a la base de datos si no es desde el servidor y limitando mucho como se puede acceder al servidor.
Además las contraseñas deberían de cifrarse para evitar el robo de cuentas.