

Sistema básico de microblogging usando Java RMI

Índice

1. - Introducción.....	3
1.1 - ¿Qué versión de Java hemos usado?.....	3
1.2 - ¿Qué IDE hemos utilizado?.....	3
1.3 - Módulos del proyecto.....	4
2. - Estructuración del proyecto.....	5
2.1 - Base de Datos.....	5
2.1.1 – Inicio de la Base de Datos.....	6
2.1.2 – Diagrama UML de la BD.....	7
2.2 – Servidor.....	8
2.2.1 – Inicio del Servidor.....	10
2.2.2 – Diagrama UML del Servidor.....	11
2.3 – Cliente.....	12
2.3.1 – Diagrama UML del Cliente.....	15
3. - Funcionamiento del programa.....	16
3.1 – Login.....	16
3.2 – Seguir A... y Dejar de Seguir A.....	19
3.3 – Enviar Trino.....	21
3.4 – Logout.....	24
3.5 – Reconexión.....	27
4. - Conclusiones y mejoras.....	28

1. Introducción

1.1 - ¿Qué versión de Java hemos usado?

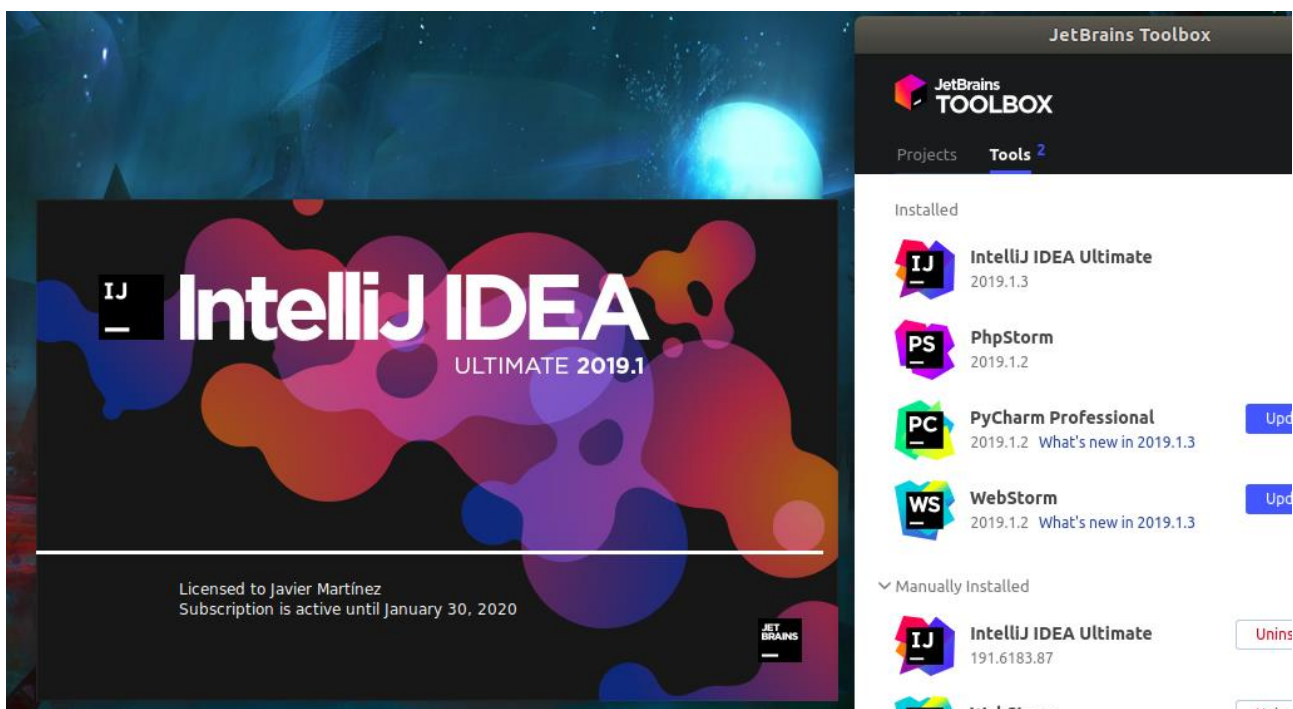
La versión de java utilizada en esta práctica ha sido el JDK11, como podemos comprobar en la captura realizada.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
esquimal@esquimal115:~$ java --version
java 11.0.2 2019-01-15 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.2+9-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.2+9-LTS, mixed mode)
```

Versión de Java utilizada

1.2 - ¿Qué IDE hemos utilizado?

A pesar de que el IDE recomendado para realizar la práctica es el Eclipse y de haberlo utilizado durante bastante tiempo, en este caso me he decantado por realizarla con el IntelliJ IDEA, ya que desde mi punto de vista es un IDE realmente completo y su versión profesional se puede tener gratis siendo estudiante universitario. A continuación se adjunta una captura de la pantalla de carga del IDE y su versión.



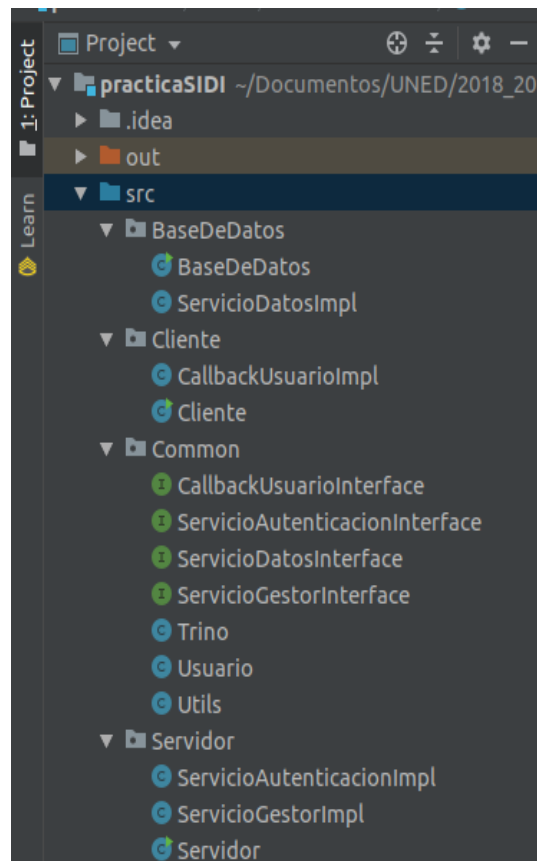
Versión del IDE utilizado

1.3 – Módulos del proyecto

Principalmente, el proyecto está estructurado en 4 carpetas principales, las cuales son:

- **Base de Datos:** Se encarga de almacenar la información de los usuarios registrados, así como la lista de Trinos que envía cada usuario. Solo el servidor tiene acceso a esta BBDD
- **Servidor:** Es el encargado de controlar el servicio gestor y de autenticación de los usuarios del chat. Hace de intermediario entre la BBDD y el proceso Cliente.
- **Cliente:** Permite el registro de los Usuarios del chat e interactuar con este pudiendo escribir mensajes, enviarlos, recibirlos, hacerse seguidor de alguien....
- **Common:** Parte del proyecto que guarda las interfaces y algunos métodos comunes al servidor, cliente o BBDD.

A continuación se adjunta una imagen que muestra la estructuración del proyecto en sus respectivas carpetas.



Organización de los módulos

Posteriormente se explicarán con más detalle cada una de estas clases y algunos de los métodos y funciones más importantes de estos.

2. Estructuración del Proyecto

2.1 – Base de Datos

Como se ha descrito anteriormente, este módulo se encarga de almacenar los usuarios registrados, con todos los datos que ello abarca. Además se encarga de ir almacenando los trinos que se han ido enviando a través de los participantes. Este módulo consta de 2 clases principales:

- **Base de Datos:** Principalmente se encarga de dar una interfaz a la persona que gestione la base de datos, a la cual llamaremos “gestor”, a través de esta interfaz el gestor puede ejecutar las siguientes acciones:

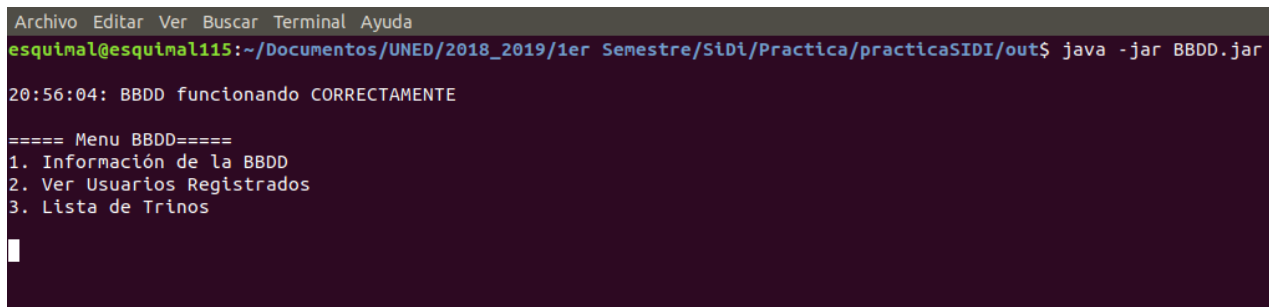
1. Obtener la información de la base de datos, por ejemplo, en que puerto se encuentra conectada.
2. Da información sobre los usuarios registrados, a través del método “registerUserBBDD”. Con este método podemos saber su nombre, nick y los usuarios que le siguen.
3. Con el último método, stopBBDD(), desconectamos la BBDD. Es decir, realizamos un unbind.

- **ServicioDatosImpl:** Como su nombre indica, se trata de la interfaz “ServicioDatos”, con la cual se implementarán los métodos que permite a la BBDD almacenar o eliminar cierta información, como por ejemplo:

1. Añadir o eliminar seguidores, con el uso de métodos como “addSeguidor()” o “removeSeguidor()”.
2. Añadir trinos a la lista de Trinos o a la lista de Trinos pendientes en caso de que el usuario no estuviera conectado, para ello utilizamos métodos como “addTrino()” y “addTrinoPendiente()”, respectivamente.
3. Similar al primer caso, también se encarga de registrar, realizar y comprobar la correcta creación de los usuarios, bajo las restricciones de que 2 usuarios no pueden tener el mismo nick.
4. Lista a todos los usuarios registrados haciendo uso de la función “getTodosUsuarios()”.

2.1.1 – Inicio de la Base de Datos

La siguiente imagen corresponde a la BD recién arrancada, pero posteriormente se mostrarán más imágenes del funcionamiento de esta.



```
Archivo Editar Ver Buscar Terminal Ayuda
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out$ java -jar BBDD.jar

20:56:04: BBDD funcionando CORRECTAMENTE

===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos
█
```

Se puede comprobar que al iniciar la BD obtenemos un log con una confirmación de que está funciona correctamente y a continuación se muestra un menú para que el gestor de la base de datos pueda interactuar con ella. Cabe destacar que en el momento de la captura solo se estaba ejecutando la base de datos.

2.1.2 – Diagrama UML de la Base de Datos

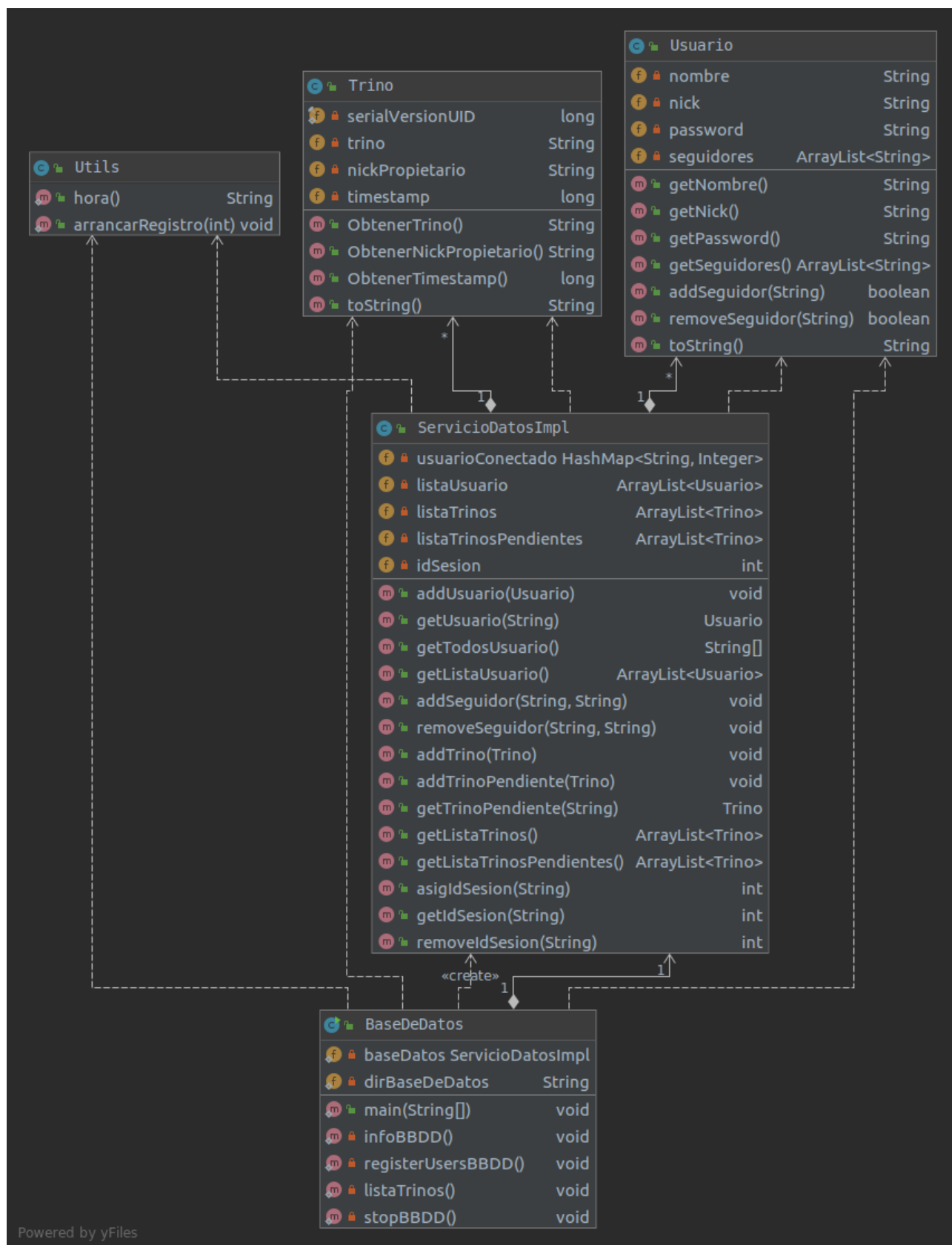


Diagrama UML de la BD creado con el IDE IntelliJ

2.2 – Servidor

El servidor tiene el trabajo de realizar de intermediario en la comunicación entre el Cliente y la BBDD. Además tiene la función de loguear y desloguear a los usuarios, así como el intercambio de Trinos entre estos. Podemos distinguir 3 clases principales en este módulo.

- **Servidor:** Es la clase principal del módulo, se encarga de levantar el servidor y conectarlo a la BBDD, además permite la conexión unificada entre todos los Clientes conectados en el sistema. También ofrece una pequeña interfaz para la gestión de este. Este módulo posee unos métodos privados muy similares a los de la BBDD. Sobre todo en el de obtener la información del servidor y en el de desconectarlo.

El servidor de forma automática lleva un registro de la gente que se ha registrado, que ha iniciado sesión, que se ha desconectado, que ha enviado un Trino, etc. A continuación podemos ver estas acciones en un usuario en la siguiente imagen, aunque posteriormente se añadirán más para mostrar todo el funcionamiento:

```
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSiDi/out$ java -jar Server.jar
11:02:04: Conexión CORRECTA con la BBDD
11:02:04: *** Servidor funcionando CORRECTAMENTE ***

===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

11:02:25: Registrado el usuario: esquimal
11:07:46: El usuario esquimal ha iniciado la sesion
11:34:30: Desconectado el usuario: esquimal
11:34:30: El usuario esquimal se ha desconectado del servidor
□
```

Server Log

El método más destacado podría ser el siguiente:

1. El método “conectedUsers()”, nos devuelve una lista con todos los usuarios que hay logueados en el sistema en ese momento.

```
=== Usuarios Conectados ===
esquimal

===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse
□
```

Usuarios Conectados

- **ServicioAutenticaciónImpl:** Esta clase permite a grande rasgos el registro de lo usuarios en el sistema y su conexión a él. Además, implementa los métodos que se describen en la interfaz de “ServicioGestorInterface”. Los métodos más destacados de esta clase son los siguientes:

1. El método “registrarUsuario (Usuario user)”, como su nombre indica, permite registrar a un nuevo usuario en el sistema. El servidor comprobará en la BD que el nick de dicho usuario no se encuentra en uso, en caso de que así sea, el usuario se registrará correctamente en el sistema y así nos lo indicará este.

2. Otros dos métodos complementarios son el de “iniciarSesionUsuario (String nick, String pass)” y “cerrarSesionUsuario (String nick)”, como sus nombres indican, se encargarán de conectar o desconectar al Cliente del sistema, pero no lo borrará de la BD.

- **ServicioGestorImpl:** Posee los métodos que permiten la correcta gestión del servidor. Implementa lo métodos que posee la interfaz “ServicioGestorInterface”. Es una clase un poco más compleja que la anterior, ya que posee bastante más métodos y funciones que coordinan los estados de los clientes y de los trinos enviados. A continuación resumiremos algunos de los métodos/funciones más importantes de esta clase:

1. El constructor se encarga de conectar el servidor a la BD. En caso de que esta no esté levantada correctamente, se producirá un error en la conexión, ya que no encontrará el objeto pertinente, que en este caso es la BD.

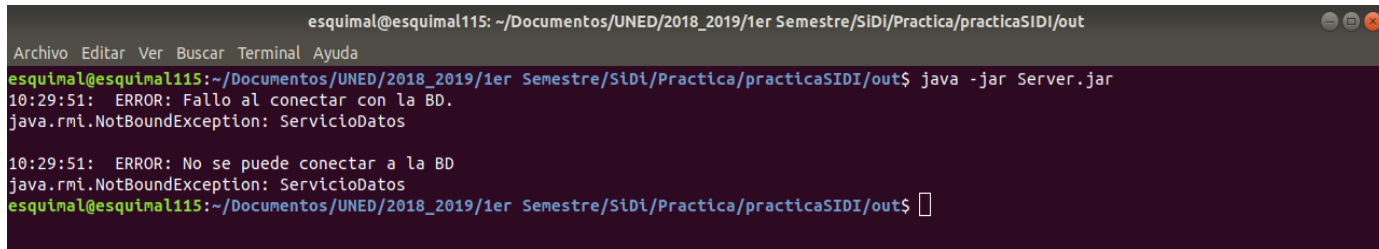
2. Las funciones “addSeguirA” y “removeSeguirA” se encargan de añadir los seguidores de un usuario en la base de datos en caso de que el usuario al que se quiera seguir esté registrado en el sistema. Ambas funciones interactúan con la base de datos para realizar la comprobación que hemos citado anteriormente.

3. Uno de los métodos clave de esta clase y posiblemente de la práctica es el de “AddTrino()”, ya que es el que se encarga de enviar el mensaje de un usuario a todos los usuarios que le sigan, para ello hace uso del callback., en caso de que el usuario no estuviera conectado, el trino se almacenaría en un Array de la BD que contiene todos los trinos pendientes de ser enviados. Dentro de estepunto también incluiremos la función de “trinosPendientes()”, la cual recuperará los trinos pendientes para cierto usuario cuando este se conecte al sistema.

4. Los métodos “registrarCallback()” y “removeRegistroCallback()” son utilizados para indicar si un usuario se ha conectado o desconectado. Esto nos permite saber si podrá recibir trinos o no.

2.2.1 – Inicio del servidor

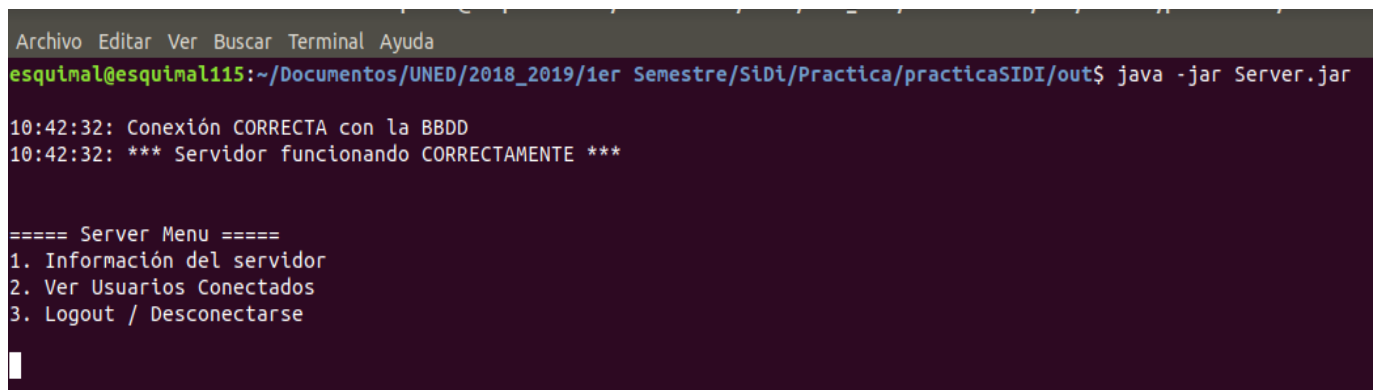
Se pueden dar dos casos, uno es que el servidor se conecte correctamente a la BD y por tanto tendrá éxito en la conexión y otro es que no se pueda conectar a esta, bien porque ha sufrido un fallo o bien porque no está levantada. A continuación se muestran capturas de ambos casos.

A terminal window titled 'esquimal@esquimal115: ~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out'. The terminal shows the command 'java -jar Server.jar' being executed. The output consists of two error messages: '10:29:51: ERROR: Fallo al conectar con la BD. java.rmi.NotBoundException: ServicioDatos' and '10:29:51: ERROR: No se puede conectar a la BD java.rmi.NotBoundException: ServicioDatos'. The prompt returns to the user.

```
esquimal@esquimal115: ~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out
Archivo Editar Ver Buscar Terminal Ayuda
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out$ java -jar Server.jar
10:29:51: ERROR: Fallo al conectar con la BD.
java.rmi.NotBoundException: ServicioDatos

10:29:51: ERROR: No se puede conectar a la BD
java.rmi.NotBoundException: ServicioDatos
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out$
```

Intento de conexión del Server con una BD no levantada

A terminal window titled 'esquimal@esquimal115: ~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out'. The terminal shows the command 'java -jar Server.jar' being executed. The output shows a successful connection: '10:42:32: Conexión CORRECTA con la BBDD' and '10:42:32: *** Servidor funcionando CORRECTAMENTE ***'. Below this, a 'Server Menu' is displayed with three options: '1. Información del servidor', '2. Ver Usuarios Conectados', and '3. Logout / Desconectarse'. The prompt returns to the user.

```
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out$ java -jar Server.jar

10:42:32: Conexión CORRECTA con la BBDD
10:42:32: *** Servidor funcionando CORRECTAMENTE ***

===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse


```

Conexión correcta del servidor

2.2.2 – Diagrama UML del Servidor

Diagrama UML del Servidor creado con el IDE IntelliJ

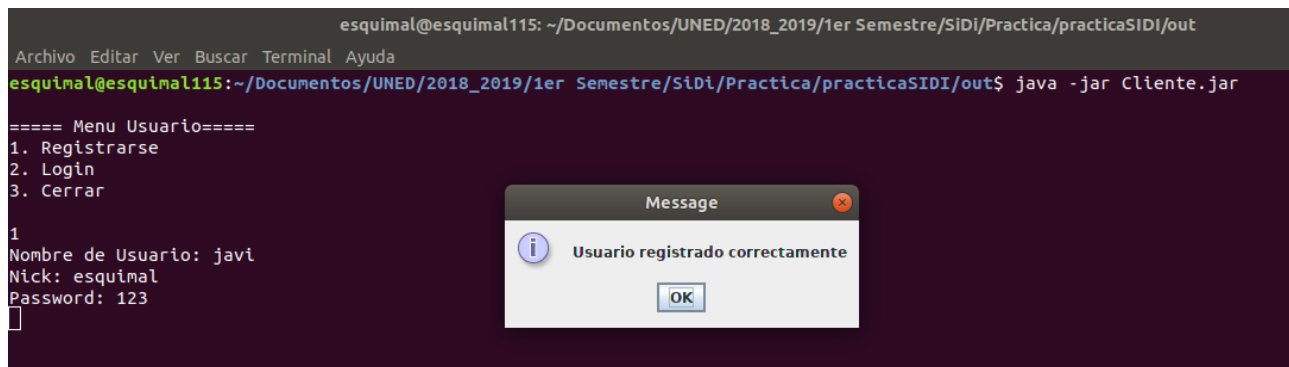
2.3 Cliente

La clase cliente es la encargada de interactuar con el usuario del chat, en ella se desarrolla toda la acción que tiene lugar de cara a la persona que utiliza el servicio.

Antes que nada debemos hacer una importante diferenciación entre “Cliente” y “Usuario”. Podríamos decir que la clase Cliente es el proceso que se encarga de realizar todas las interacciones entre los objetos Usuario como por ejemplo, la creación de nuevos Usuarios, el acceso de estos al chat, la desconexión del Usuario, etc.

Dentro de la clase Cliente podemos encontrarnos con 2 menús diferentes. Uno de ellos es el de inicio, en el cual le permitirá al usuario del programa registrarse como un nuevo usuario, realizar un login en caso de que ya tenga una cuenta o cerrar la aplicación. A continuación los trataremos con mayor detalle estas funcionalidades.

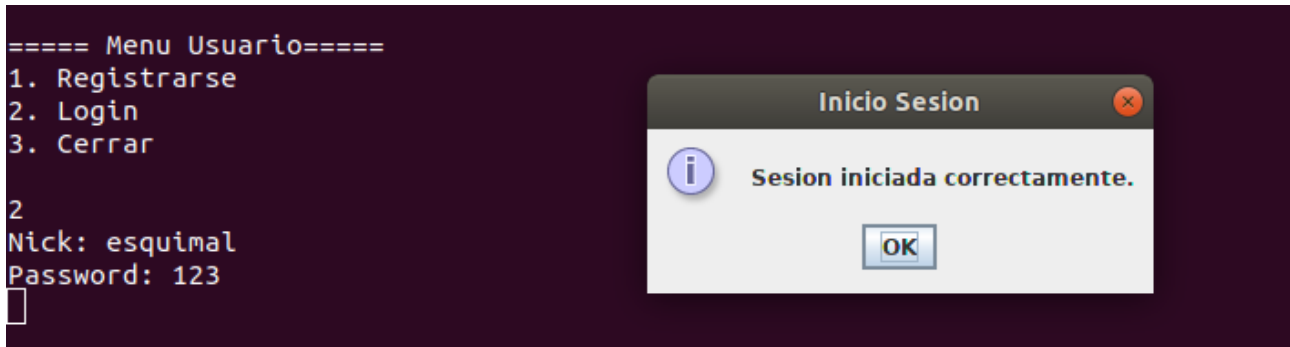
1. Registrarse: Permite a una persona física registrarse como usuario en el sistema, para ello se le pedirá un nombre de usuario, un nick (el cual será su nombre en el chat) y una contraseña que será pedida cuando desee acceder a la cuenta. Debemos tener en cuenta que NO pueden existir dos Usuarios registrados con el mismo nick, pero sí con el mismo nombre. Esta restricción es necesaria para que en el chat no hayan confusiones para saber quien habla, o para que el sistema sepa a quien enviar los mensajes cuando realice el callback.



```
esquimal@esquimal115: ~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSiDi/out
Archivo Editar Ver Buscar Terminal Ayuda
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSiDi/out$ java -jar Cliente.jar
===== Menu Usuario=====
1. Registrarse
2. Login
3. Cerrar
1
Nombre de Usuario: javi
Nick: esquimal
Password: 123
[OK]
```

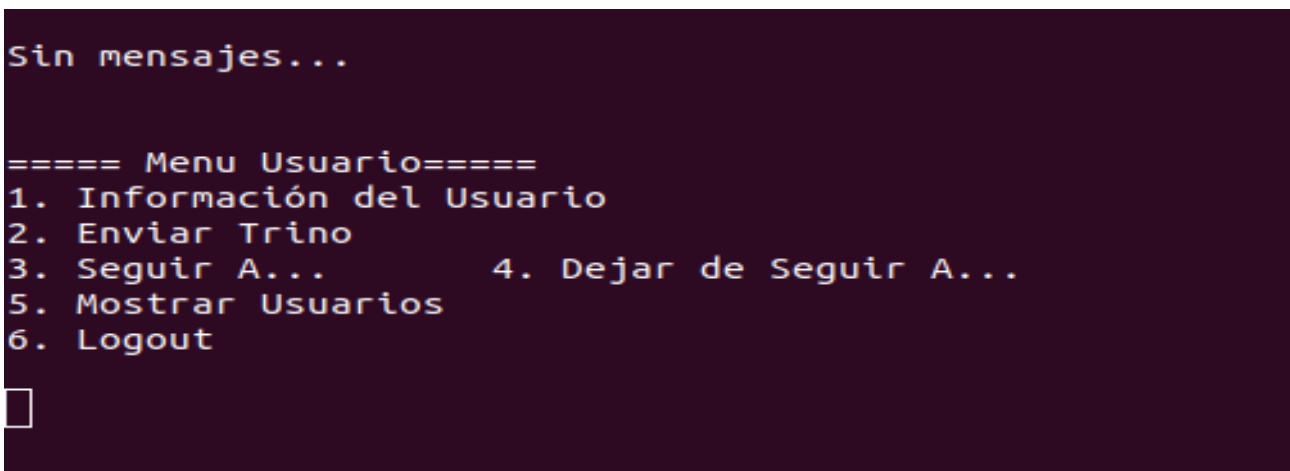
Registro correcto de Usuario

2. Login: Una vez el usuario se ha registrado correctamente, puede hacer un login en la aplicación para acceder a la cuenta que ha creado y de esta manera poder interactuar con otros usuarios.



Login correcto

Una vez el usuario ha hecho login de forma correcta, se le notificará al usuario de ello y aparecerá un nuevo menú con más opciones. Además aparecerá un mensaje antes del menú indicándole si tiene mensajes pendientes de los usuarios a los que sigue. En este caso al ser el único usuario registrado no va a tener mensajes pendientes de sus seguidores.



Menú de Usuario

A continuación se explicaran brevemente las acciones que puede realizar el usuario dentro del menú de la imagen anterior.

1. Con la opción de información del usuario podemos obtener nuestro nombre de usuario y el nick.

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...          4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

1

=== Informacion sobre el Usuario ===

Nombre: javi
Nick: esquimal
```

Información del Usuarios

2. La segunda opción sería “Enviar Trino” y lo recibirían todos aquellos usuarios que siguiesen nuestra cuenta, básicamente es el servidor y la base de datos en quien recae todo este proceso de callback.
3. Con las opciones de “Seguir A...” y “Dejar de Seguir A...” podemos seguir o dejar de seguir a un usuario para recibir los Trinos que este envíe. En el apartado de pruebas de ejecución se mostrarán algunos ejemplos de esto.
4. “Mostrar Usuarios” nos proporcionará una lista de todos los nicks de los usuarios registrados en el sistema, se trata de una función bastante útil en caso de que queramos seguir a alguien.
5. Por último si deseamos desconectarnos del sistema, realizaremos un “Logout”, que es la última opción del menú. Debemos tener en cuenta que si un usuario cierra la terminal sin desconectarse este permanecerá logeado en el sistema, ya que la función de auto-logout no está implementada porque no correspondía a la práctica.

2.3.1 - Diagrama UML del Cliente

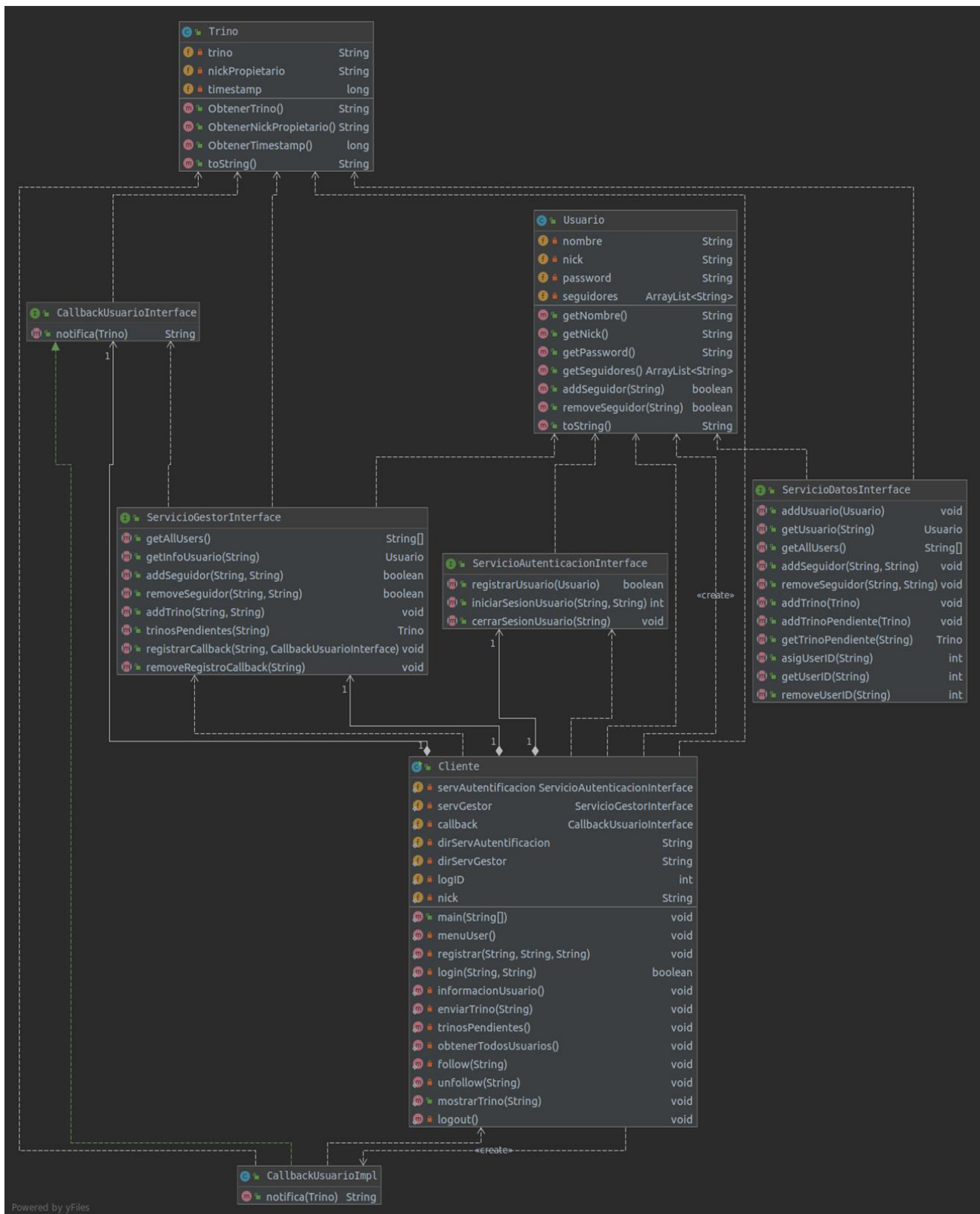


Diagrama UML del Cliente creado con el IDE IntelliJ

3. Funcionamiento del programa

Para mostrar el funcionamiento del programa vamos a tener 4 participantes en el chat o en el sistema, y a partir de sus acciones iremos mostrando los logs de la BD, del Servidor y el completo funcionamiento del cliente.

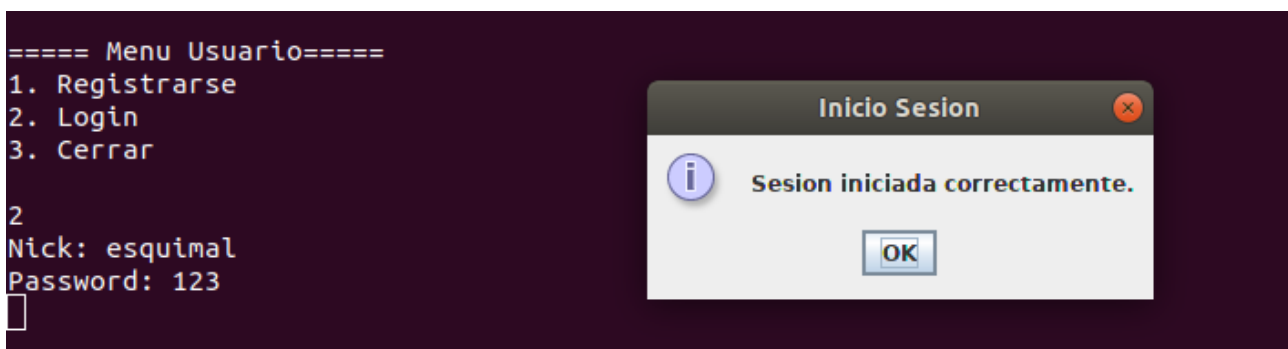
El nombre y el nick de cada participante serán respectivamente:

1. javi – esquimal
2. carmen – carmencin
3. jorge – jorgito
4. adrian – adri

Vamos a tratar en imágenes un correcto funcionamiento del programa, pero se comentarán brevemente cuales son los posibles fallos que se podrían producir en caso de que no se introdujeran los datos correctamente.

3.1 - Login

El login comienza en el proceso del Cliente, cuando un usuario se registra como nuevo usuario, en caso de no tener cuenta, y posteriormente se logea para poder acceder al chat.



Login del usuario esquimal

Una vez el usuario se ha registrado podemos ver los logs de la BD para comprobar que se han registrado los usuarios, además podemos usar también la función de ver todos los usuarios registrados, para comprobarlo de forma más concisa.

```
Archivo Editar Ver Buscar Terminal Ayuda
esquimal@esquimal115:~/Documentos/UNED/2018_2019/1er Semestre/SiDi/Practica/practicaSIDI/out$ java -jar BBDD.jar

12:02:10: BBDD funcionando CORRECTAMENTE

===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos

12:02:37: Nuevo usuario registrado
12:02:51: Nuevo usuario registrado
12:03:04: Nuevo usuario registrado
12:03:12: Nuevo usuario registrado
```


Log de la BD

```
=== Usuarios registrados ===  
  
Nº de usuarios registrados: 4  
Nombre || Nick || Usuarios que le siguen  
  
javi || esquimal || []  
carmen || carmencin || []  
jorge || jorgito || []  
adrian || adri || []  
  
===== Menu BBDD=====  
1. Información de la BBDD  
2. Ver Usuarios Registrados  
3. Lista de Trinos
```



Lista de usuarios registrados

Además, si observamos al proceso del Servidor, también podemos observar que hace un log del nuevo usuario registrado, así como su conexión en el sistema, como muestra la siguiente imagen.

```
===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

12:02:37: Registrado el usuario: esquimal
12:02:51: Registrado el usuario: carmencin
12:03:04: Registrado el usuario: jorgito
12:03:12: Registrado el usuario: adri
12:03:55: El usuario esquimal ha iniciado la sesion
12:04:09: El usuario carmencin ha iniciado la sesion
12:04:24: El usuario jorgito ha iniciado la sesion
12:15:37: El usuario adri ha iniciado la sesion
█
```

Log del Server

Al igual que en la Base de Datos, también podemos ver en el servidor los usuarios que hay conectados en este momento, haciendo uso de la opción 2 del gestor del Server. Como muestra el log de la imagen anterior, deberían haber 4 usuarios conectados y efectivamente es así, como se muestra en la siguiente imagen:

```
=== Usuarios Conectados ===

carmencin
adri
esquimal
jorgito

===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse
█
```

Usuarios conectados en el sistema

3.2 – Seguir y Dejar de Seguir A...

Como se ha comentado anteriormente, esta función del sistema, permita a usuarios registrados poder seguir a otros usuarios registrados en el sistema para que cuando estos últimos escriban un Trino lo reciban todas aquellas personas que le siguen. En este caso haremos que “jorgito” y “carmen” sigan al usuario “Esquimal”, de tal forma que cuando “Esquimal” escriba algún trino será recibido por “jorgito” y “carmen”, pero no por “Adri”. Si se intentase seguir a alguien que no existe en el sistema nos daría un aviso de que no se puede seguir a X usuario. También ocurriría un error en caso de que un usuario se intentase seguir a si mismo.

A continuación se mostrará el **funcionamiento en el Cliente** con las siguiente imágenes:

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...           4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

3

A quien desea seguir? esquimal
Ahora sigues al usuario: esquimal
```

Función de Seguir A...

A continuación, podemos observar que el **log de la Base de Datos** se ha actualizado, así como la información guardada dentro del Usuario “esquimal”. En la siguiente imagen se distingue la actualización del log, así como la información de los usuarios registrados, ya que ahora “esquimal” tiene 2 seguidores, que son los que aparecen entre corchetes.

```
===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos

12:32:07: Actualizada la información de esquimal
12:33:43: Actualizada la información de esquimal
2

=== Usuarios registrados ===

Nº de usuarios registrados: 4
Nombre || Nick || Usuarios que le siguen

javi || esquimal || [jorgito, carmencin]
carmen || carmencin || []
jorge || jorgito || []
adrian || adri || []

===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos
```

Log de la BD y actualización de los seguidores en “esquimal”

Además de la BD, el servidor también registra los usuarios que siguen a otros, pero en este caso la información es más detallada ya que muestra quien ha empezado a seguir a quien. Podemos verlo con más claridad en la siguiente imagen del **log del Server**:

```
===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

12:32:07: El usuario jorgito ha empezado a seguir a esquimal
12:33:43: El usuario carmencin ha empezado a seguir a esquimal
█
```

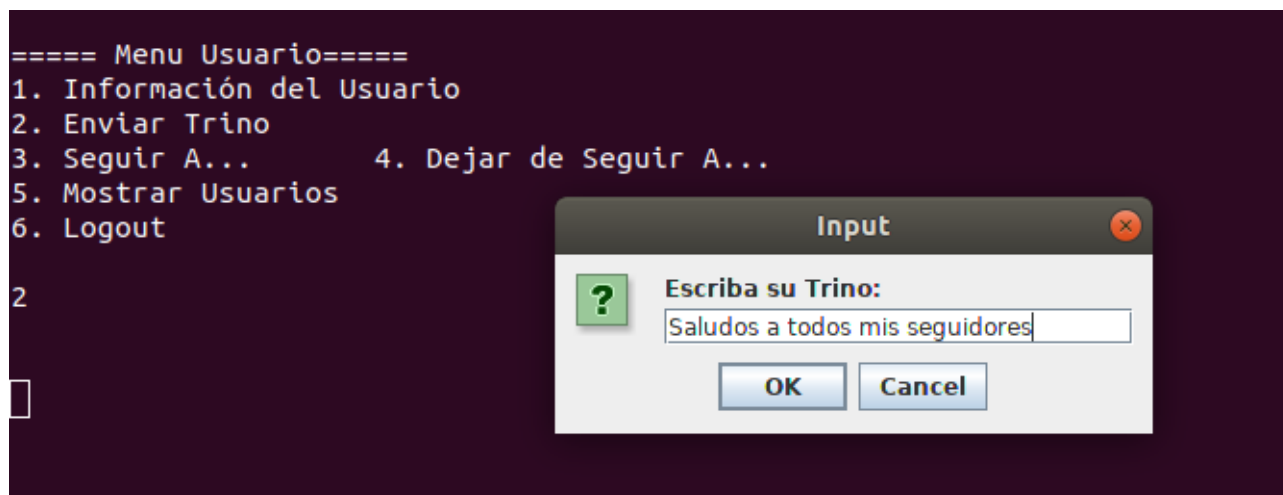
Log de seguidores en el Server

El caso de dejar de seguir a alguien sería muy similar al que hemos explicado de “Seguir A...”, simplemente que el seguidor desaparecería de la lista de seguidores de la Base de Datos y dicha persona ya no recibiría los mensajes del usuario que ha dejado

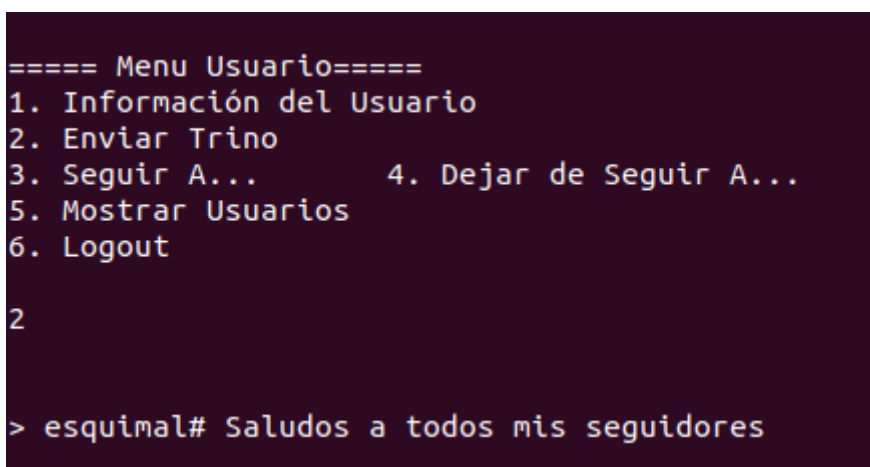
de seguir. Pero en este caso no realizaremos esta acción ya que me interesa explicar el envío y recepción de mensajes con al menos 2 seguidores de un usuario.

3.3 – Enviar Trino

En este caso se mostrará el uso de envío de los Trinos haciendo uso del usuario “esquimal” ya que es el único usuario que le sigue alguien, en este caso “jorgito” y “carmen”.



Escritura del Trino



El trino ha sido enviado con el usuario “esquimal”

En las siguientes imágenes podemos observar como el usuario “jorgito” y “carmen” han recibido el mensaje enviado por el usuario “esquimal”, pero el usuario “adri” no ha recibido tal mensaje ya que no sigue al usuario “esquimal”. Se ha usado también el comando de info del usuario, para corroborar que efectivamente el usuario ha recibido el mensaje.

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

> esquimal# Saludos a todos mis seguidores

1

=== Informacion sobre el Usuario ===

Nombre: jorge
Nick: jorgito
```

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

> esquimal# Saludos a todos mis seguidores

1

=== Informacion sobre el Usuario ===

Nombre: carmen
Nick: carmencin
```

Los seguidores de un usuario reciben su mensaje

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

1

=== Informacion sobre el Usuario ===

Nombre: adrian
Nick: adri
```

En caso de no ser seguir a un usuario que ha publicado un trino,
no recibes sus mensajes

Para comprobar que efectivamente el usuario “adri” no ha recibido el mensaje y que no hemos hecho “trampas” podemos ir al log tanto de la BD, como del servidor y comprobar a quien se envían los mensajes.

```
===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos

12:57:37: esquimal ha escrito un mensaje
█
```

Log de la BD por el mensaje enviado

```

===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

12:32:07: El usuario jorgito ha empezado a seguir a esquimal
12:33:43: El usuario carmencin ha empezado a seguir a esquimal
12:57:37: El usuario esquimal ha publicado un trino
*** Se envía un Trino al usuario jorgito
*** Se envía un Trino al usuario carmencin

```

Log del Server por los mensajes enviados

Observando las anteriores imágenes podemos observar que efectivamente el mensaje enviado por “esquimal” se ha enviado a “jorgito” y “carmen”, pero no al usuario “adri”.

Cabe destacar que todo este funcionamiento es gracias al callback que se realiza para que el servidor envíe el mensaje a todos los usuarios que sean seguidores de X usuario y dicho mensaje se refleje en la pantalla de cada usuario.

3.4 – Logout

Esta función se usará cuando un usuario ya no quiera participar más en el chat, por el motivo que sea. Si un usuario se desconecta, este ya no recibirá Trinos en su consola. Si algún seguidor suyo publica Trinos, estos serán enviados al usuario cuando se reconecte. Para mostrar esta función vamos a desconectar a los usuarios “adri” y “jorgito”, luego el usuario “esquimal” enviará un par de trinos, que en este caso serán recibidos solo por “carmencin”, ya que es la única conectada que sigue a “esquimal”.

```

=== Información sobre el Usuario ===

Nombre: jorge
Nick: jorgito

===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

6

===== Menu Usuario=====
1. Registrarse
2. Login
3. Cerrar

```

Desconexión del usuario “jorgito”

Para asegurarnos de que efectivamente jorge ha sido desconectado podemos revisar el log del servidor, el cual nos tendría que informar de esto.

```
===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

12:32:07: El usuario jorgito ha empezado a seguir a esquimal
12:33:43: El usuario carmencin ha empezado a seguir a esquimal
12:57:37: El usuario esquimal ha publicado un trino
*** Se envía un Trino al usuario jorgito
*** Se envía un Trino al usuario carmencin
13:26:41: Desconectado el usuario: adri
13:26:41: El usuario adri se ha desconectado del servidor
13:27:03: Desconectado el usuario: jorgito
13:27:03: El usuario jorgito se ha desconectado del servidor
█
```

Log del server para comprobar la desconexión

Una vez comprobado esto, el usuario “esquimal” enviará un par de trinos, y reconectaremos la cuenta de “jorgito”, para efectivamente comprobar que los recibe al iniciar sesión de nuevo.

```
===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

2

> esquimal# Prueba 1 para seguidor desconectado

===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout

2

> esquimal# prueba 2 para seguidor desconectado
```

El usuario “esquimal” envía 2 trinos a sus seguidores

En la BD, podemos ver como queda registrado el hecho de que el usuario “esquimal” ha enviado un trino, pero que alguno de sus seguidores está desconectado, por tanto guarda el trino en una lista de trinos pendientes.

```
===== Menu BBDD=====
1. Información de la BBDD
2. Ver Usuarios Registrados
3. Lista de Trinos

12:57:37: esquimal ha escrito un mensaje
13:37:01: esquimal ha escrito un mensaje
13:37:01: Se añade un trino a la listaTrinosPendientes.
13:41:22: esquimal ha escrito un mensaje
13:41:22: Se añade un trino a la listaTrinosPendientes.
```

“esquimal” envía 2 trinos y cada uno se añade a trinosPendientes ya que tiene un seguidor desconectado, log de la BD

A continuación en el **log del server** podemos ver como “esquimal” ha enviado 2 trinos y que este ha sido enviado correctamente al usuario “carmencin” pero queda pendiente para “jorgito”, como podemos ver en la siguiente imagen:

Log

```
===== Server Menu =====
1. Información del servidor
2. Ver Usuarios Conectados
3. Logout / Desconectarse

12:32:07: El usuario jorgito ha empezado a seguir a esquimal
12:33:43: El usuario carmencin ha empezado a seguir a esquimal
12:57:37: El usuario esquimal ha publicado un trino
*** Se envía un Trino al usuario jorgito
*** Se envía un Trino al usuario carmencin
13:26:41: Desconectado el usuario: adri
13:26:41: El usuario adri se ha desconectado del servidor
13:27:03: Desconectado el usuario: jorgito
13:27:03: El usuario jorgito se ha desconectado del servidor
13:37:01: El usuario esquimal ha publicado un trino
*** Trino pendiente para el usuario: jorgito
*** Se envía un Trino al usuario carmencin
13:41:22: El usuario esquimal ha publicado un trino
*** Trino pendiente para el usuario: jorgito
*** Se envía un Trino al usuario carmencin
```

del

servidor mostrando que hay un trino pendiente para “jorgito”

3.5 – Reconexión

Si un usuario que ya se ha registrado se desea reconectar en el sistema es libre de hacerlo, ya que la base de datos no pierde sus credenciales de registro. Cuando un usuario se reconecta, es posible que alguna persona de las que sigue haya escrito algún trino, como ha ocurrido anteriormente. Cuando el usuario “jorgito” se vuelve a conectar, podemos ver como recibe los mensajes que tenía pendientes de “esquimal”. Esta función también es posible gracias al callback que se realiza.

```
===== Menu Usuario=====
1. Registrarse
2. Login
3. Cerrar

2
Nick: jorgito
Password: 123

Mensajes pendientes:

>esquimal# Prueba 1 para seguidor desconectado
>esquimal# prueba 2 para seguidor desconectado

===== Menu Usuario=====
1. Información del Usuario
2. Enviar Trino
3. Seguir A...      4. Dejar de Seguir A...
5. Mostrar Usuarios
6. Logout
```

Trinos pendientes del usuario “jorgito”

4. Conclusiones y mejoras

Personalmente, es una de las mejores prácticas que por ahora he hecho en la carrera. Es totalmente distinta a lo que se ve en otras asignaturas de programación y eso le da un aire novedoso al trabajo. Creo que lo que más me ha agradado es la idea de realizar un prototipo de proyecto que realmente tiene una función final, que en este caso es crear una especie de chat haciendo uso del RMI, un tema de programación totalmente novedoso, ya que en general las prácticas que hacemos se basan en estructuras de datos y al final se hace algo repetitivo.

Mejoras

- A pesar de no ser obligatorio, podría haber realizado una interfaz gráfica con Java Swing para que fuera algo más visual, pero prefería centrarme en lo realmente obligatorio de esta.
- Otra mejora en la práctica, a pesar de que creo que no se podía (tal vez por su complejidad), es que cuando un usuario se desconectase cerrando la terminal, es decir, sin hacer uso del logout, este fuera también desconectado del sistema ya que conforme está implementado, seguiría conectado y al intentar reconectarse saldría como usuario ya conectado.
- La base de datos en este caso es volátil, es decir, que cuando se desconecta, todo lo que esté almacenado en esta se pierde. Por ello se podría haber hecho una BD de forma no volátil, guardando toda la información en un archivo y cada vez que volviese a ser levantada, que recuperase los datos de dicho archivo, de esta forma sería más correcta. Pero claro... Lo realmente correcto sería realizar la BD en SQL y no esta BD improvisada en java.