# Práctica 1: aplicación de mensajería instantánea

Alberto Mateos Checa

Implementación de sistema de mensajería instantánea entre dispositivos con sistema operativo Android.

# 1. Introducción.-

El nombre del sistema implementado es **AndroMessenger** y permite establecer una comunicación de mensajería instantánea entre dispositivos móviles dotados con sistema operativo Android a través de una conexión de red.

Dicha comunicación está basada en el uso de sockets TCP.

# 2. Partes.-

El sistema AndroMessenger consta de dos subsistemas fundamentales: servidor de directorio y cliente Android.

## 2.1 Servidor de directorio.-

El servidor de directorio es una aplicación creada en lenguaje Java que establece una escucha de peticiones a través de un socket TCP en el puerto 9090.

Este servidor es el encargado de registrar la conexión de los diferentes clientes Android y almacenar los contactos "amigos" de cada uno de los contactos. Además, es capaz de enviar la lista de contactos y contactos conectados a cada uno de los clientes cuando éstos lo solicitan, de forma, que facilita a los mismos establecer una comunicación entre ellos.

Para la implementación del servidor se ha hecho uso de las siguientes clases:

- **Servidor.java:** es la clase principal de la aplicación. Inicia el socket TCP en el puerto 9090 para la escucha de peticiones, creando una hebra por cada cliente que se conecta. Además, a modo de prueba crea una serie de contactos de forma predefinida.
- **Servicio.java:** es la subclase utilizada para crear las hebras de atención al socket. Recibe los mensajes de los clientes y genera las respuestas a los mismos.
- **TiposMensaje.java:** define los tipos de mensaje que pueden ser enviados y/o recibidos en la comunicación entre servidor y cliente.
- **Usuario.java:** define y almacena la información de un usuario.

## 2.2 Cliente Android.-

Es una aplicación desarrollada para móviles con sistema operativo Android que permite la conexión con el servidor de directorio y la posterior comunicación con otros dispositivos Android. Además, permite una completa gestión de los contactos de un usuario: creación de nuevos contactos, eliminación de contactos, visualización de contactos conectados y no conectados…

Las clases involucradas en el funcionamiento de la aplicación son las siguientes:

- **Conversacion.java:** esta clase define la información propia de una conversación: contacto con el que se mantiene la conversación, texto de la misma, IP del contacto, etc.
- **IntroducirDatos.java:** muestra una pantalla que solicita al usuario introducir el nombre de un contacto a añadir o del nuevo apodo que se quiere tener. De esta forma, una misma pantalla o activity es utilizada para ambas operaciones.
- **ListaContactos.java**: almacena la lista de contactos del usuario y la muestra en la pantalla principal de la aplicación a través de una ListView.
- **Login.java:** es la pantalla que se muestra al iniciar la aplicación y solicita al usuario introducir su nombre de usuario, contraseña y apodo que se quiere mostrar. Además, es la encargada de realizar la petición de conexión al servidor y obtener la lista de contactos que va a ser mostrada posteriormente.
- **Peticiones.java:** realiza las peticiones al servidor de directorio, componiendo y enviando los mensajes que se solicitan.
- **PeticionesConversación.java:** crea y lanza las hebras oportunas para mantener las conversaciones con otros clientes a través de un socket TCP en el puerto 8888.
- **Principal.java:** es la pantalla principal de la aplicación, mostrando la lista de contactos, las conversaciones activas y los menúes que permiten ejecutar las diferentes funciones de la aplicación.
- **ProcesadorMensajesEntrada.java:** define la hebra encargada de gestionar los mensajes de entrada de una conversación.
- **ProcesadorMensajesSalida.java:** define la hebra encargada de gestionar los mensajes de salida de una conversación.
- **TiposMensaje.java:** define los tipos de mensaje que pueden ser enviados y/o recibidos en la comunicación entre servidor y cliente.

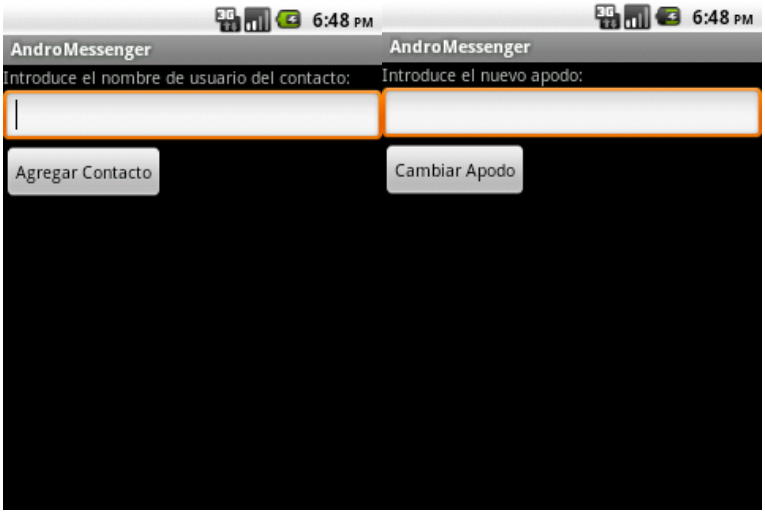
## Interfaz de usuario.

La aplicación está formada por 3 pantallas o activities:

- **Login:** es la pantalla que se muestra al ejecutar la aplicación. Permite al usuario introducir su nombre y contraseña para realizar la conexión con el servidor de directorio y el apodo que desea utilizar.


- **Principal:** muestra a la izquierda la lista de contactos (conectados o sin conectar), en la parte central la conversación activa, que puede ser seleccionada a través del desplegable de la zona superior, y en la parte inferior una zona para ingresar texto en la conversación activa.


- **Introducción de datos:** es una pantalla que se muestra en dos ocasiones diferentes según se quiera agregar un contacto o modificar el apodo que se está utilizando y que permite al usuario introducir el nombre del contacto a agregar o del apodo nuevo a utilizar para enviar la petición al servidor.




# 3. Comunicaciones y protocolos.-

En la ejecución de la aplicación intervienen dos tipos de comunicaciones. La primera en la que se comunica a los clientes Android con el servidor de directorio y la segunda en la que se pone en contacto a los diferentes clientes, permitiendo la comunicación entre ellos.

## 3.1 Comunicación cliente – servidor de directorio.-

El diagrama de estados define el protocolo de comunicación es **exactamente el mismo que el que se propone en el guión de prácticas** con la salvedad de que han sido incluidos algunos nuevos mensajes para el correcto funcionamiento de la aplicación. Es por ello que no se muestra en este documento.

### Mensajes que envía el servidor:

- **mBienvenidaServidor**: se envía al recibir una petición de conexión de un cliente.
- **mDespedidaServidor**: se envía al recibir una petición de desconexión de un cliente.
- **mRegistroCorrecto**: se envía cuando se realiza un registro correcto de un cliente.
- **mRegistroIncorrecto**: se envía cuando el registro de un cliente no se realiza satisfactoriamente.
- **mRespuestaOK**: se envía como confirmación.
- **mRespuestaFallo**: se envía como notificación de fallo ante una petición.
- **mRespuestaLocalización**: se envía como respuesta a una petición de localización de un contacto, incluyendo la dirección IP del contacto.
- **mRespuestaListaContactos**: se envía como respuesta a una petición de obtención de lista de contactos de un cliente, incluyendo la misma.
- **mRespuestaListaContactosConectados**: se envía como respuesta a una petición del cliente, incluyendo la lista de contactos que están conectados.

- **mRespuestaApodo**: se envía como respuesta de una petición de apodo de un contacto.
- **mRespuestaCambioApodo**: se envía como respuesta de una petición de cambio de apodo de un contacto.

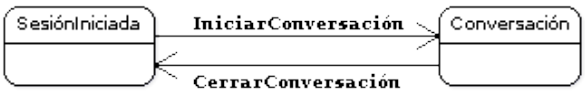
### Mensajes que envía el servidor:

- **mBienvenidaCliente**: se envía como petición de conexión de un cliente.
- **mRegistrar**: se envía para realizar una petición de registro en el servidor.
- **mSolicitarLocalización**: se envía para solicitar la IP de un contacto.
- **mAnadirContacto**: se envía como petición para agregar un contacto a la lista de contactos de un cliente.
- **mEliminarContacto**: se envía como petición para eliminar a un contacto de la lista de contactos de un cliente.
- **mSolicitudListaContactos**: se envía como petición para obtener la lista de contactos de un cliente.
- **mSolicitudApodo**: se envía para obtener el apodo de un cliente.
- **mSolicitudListaContactosConectados**: se envía como petición para obtener la lista de contactos conectados de un cliente.
- **mCierreSesion**: se envía cuando un cliente desea desconectarse del servido.
- **mSolicitudCambioApodo**: se envía para solicitar el cambio de apodo de un cliente.

## 3.2 Comunicación cliente – cliente.-

La comunicación que se realiza entre clientes Android para facilitar las conversaciones son realizadas a través de un socket TCP que es capaz de leer y escribir a través del puerto 8888 y que conecta a ambos dispositivos.

El diagrama de estados es el siguiente:




Como puede comprobarse se tienen dos estados: “Sesión iniciada” y “Conversación”. El primero se obtiene tras realizar la conexión con el servidor de directorio. Para pasar al segundo basta con iniciar una conversación con un contacto haciendo una pulsación larga sobre el nombre del contacto con el que se desea entablar la conversación. Para volver al estado anterior debe hacerse click sobre la opción “cerrar conversación” del menú que se obtiene al pulsar la tecla menú del móvil.

NOTA: hay que decir que, aunque la comunicación entre clientes ha sido programada, no se ha podido testear la función de establecer conversaciones entre clientes. Ésto se debe a que el emulador de Android dispone de una IP para todas sus ejecuciones, es decir, si se lanzan varios emuladores todos tendrán la misma dirección IP. De esta forma, es imposible establecer sockets de conexión entre

varios emuladores, imposibilitando la programación de esta aplicación puesto que no dispongo de teléfonos con sistema operativo Android para realizar las pruebas oportunas. Es por ello que las imágenes que se muestran a continuación en las que se muestra conversaciones no son reales sino que se muestran textos que han sido introducidos "a mano" para comprobar el correcto funcionamiento de los métodos relacionados con las conversaciones y de la clase Conversacion.java.

# 4. Demostración del funcionamiento.-

A continuación se van a mostrar capturas obtenidas del emulador de cada una de las funciones de la aplicación Android:

- **Login**: el usuario introduce sus datos y realiza una conexión con el servidor.

Android


Servidor

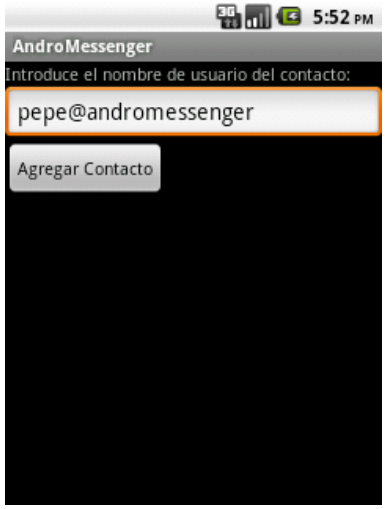

- **Pantalla principal:**

Sólo contactos conectados

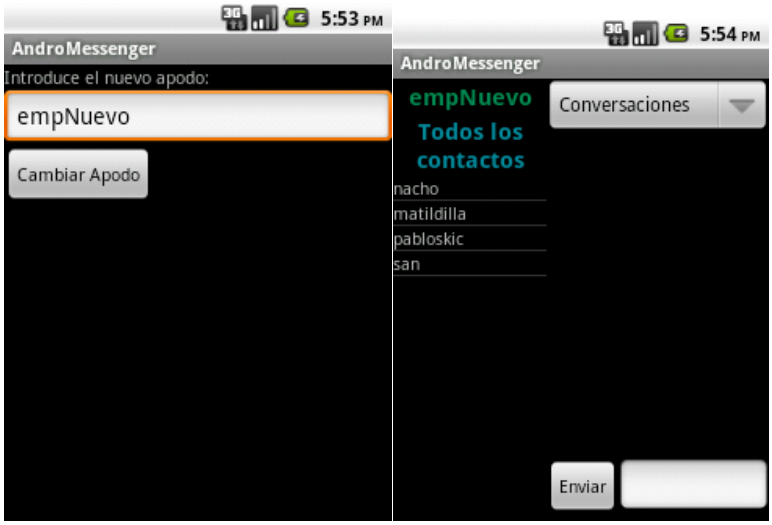

Todos los contactos

- **Menú principal** (pulsando botón menú del móvil):

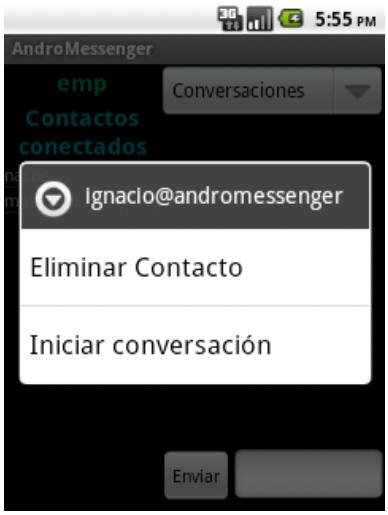

- **Función agregar contacto:**



- **Función cambiar apodo:**




- **Menú contextual** (pulsación larga sobre nombre de contacto):




- **Conversación:**

- **Selección conversación** (usando menú desplegable superior):