UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

SERVICIO DE MENSAJERÍA INSTANTÁNEA DESCENTRALIZADO Y SEGURO

DE-CENTRALIZED AND SECURE INSTANT MESSAGING SERVICE

Realizado por
José Luis Fernández Márquez

Tutorizado por
José Francisco Chicano García

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2021

Fecha defensa: Julio de 2021

Resumen

Este trabajo de fin de grado aborda el diseño de un protocolo de mensajería instantánea descentralizado y seguro, con mensajes cifrados de extremo a extremo, y la implementación de una aplicación servidor y dos aplicaciones cliente, para la plataforma Android y equipos de escritorio, que usarán dicho protocolo.

El sistema permitirá a los usuarios el envío y recepción de mensajes de texto y de archivos. Estos podrán ser privados, de grupo o de difusión.

El servidor dispondrá además de un panel de administración web, en el que se podrán gestionar los usuarios miembros, notificar a todos estos por correo electrónico y configurar algunos ajustes del sistema.

Palabras clave: mensajería instantánea, aplicaciones distribuidas, seguridad informática, privacidad

Abstract

This final degree project describes the design of a decentralized and secure instant messaging protocol, with end-to-end encrypted messages, and the implementations of a server application and two client applications, for the Android platform and desktop computers, which will be used said protocol.

The system will allow users to send and receive text messages and files. These may be private, group or broadcast.

The server will also have a web administration panel, in which member users can be managed, notify all of them by email and configure some system settings.

Keywords: instant messaging, distributed applications, computer security, privacy

Índice

Índice de contenidos

Índice de figuras

1

Introducción

1.1 Motivación

En la actualidad, el uso de las aplicaciones de mensajería instantánea es una pieza clave para la sociedad. Cualquier grupo de estudio, trabajo, amigos... usa este tipo de herramientas para estar en contacto, coordinarse, compartir ideas... El problema es que las dos grandes soluciones actuales de la mensajería instantánea (Whatsapp [1] y Telegram [2]) son aplicaciones propietarias y su código fuente resulta imposible de leer o estudiar para saber como tratan la información de los usuarios. Por otro lado, sus servidores son centralizados, obligando al usuario a conectarse a estos si quieren comunicarse con sus contactos.

La motivación de este trabajo de fin de grado surge al no existir muchas alternativas sencillas a este tipo de servicios, que sean software libre [3] y se basen en un modelo descentralizado, permitiendo así la libertad a cualquier usuario de estudiar el código fuente para conocer como se trata su información y de montar su propio servidor para convertirse en un proveedor de este servicio.

1.2 Objetivos

El objetivo de este trabajo de fin de grado será la creación de un protocolo de comunicación de mensajería instantánea descentralizada y con cifrado extremo a extremo [4], y en el desarrollo de una aplicación servidor y dos aplicaciones cliente (una para dispositivos Android y otra de escritorio) que usarán dicho protocolo. No se realizará una aplicación web cliente ya que en una de escritorio la gestión de claves pública y privada resulta más sencilla.

Con "descentralizado" nos referimos a que cualquier usuario podrá elegir entre diferentes proveedores del servicio (como es el caso del correo electrónico por ejemplo, donde un usuario puede elegir entre Gmail, Yahoo, Hotmail...). Elija el que elija, estará conectado con todos los demás usuarios, e incluso, si no le agrada ninguno de los que haya disponibles, tendrá la posibilidad de instalar en su dispositivo la aplicación servidor y convertirse en un proveedor del servicio más. Un modelo descentralizado como este fortalece mucho la privacidad, ya que dos usuarios podrían incluso instalarse la aplicación servidor cada uno en su máquina, y comunicarse entre ellos sin necesidad de un intermediario externo.

Como el objeto principal de este trabajo de fin de grado es el desarrollo de software, se enumeran a continuación una serie de requisitos funcionales para explicar todas las funcionalidades del sistema a alto nivel:

- Un usuario administrador podrá gestionar los usuarios y notificar a todos estos desde la interfaz web de administración de la aplicación servidor.

- El sistema permitirá validar el acceso a la aplicación y el registro de nuevos usuarios.

- Los usuarios podrán enviar y recibir mensajes y archivos cifrados de extremo a extremo, tanto por mensaje privado como por chats grupales o listas de distribución.

- Los usuarios podrán gestionar su lista de contactos y la configuración de sus listas de distribución y de sus salas de chat grupales en la aplicación cliente.

- La aplicación servidor tendrá un apartado de configuración, en el que se podrá ajustar una serie de parámetros del sistema.

1.3 Metodología de trabajo empleada

Para la realización de este proyecto se siguió la metodología ágil de desarrollo Scrum [5]. Esta metodología se caracteriza por la adopción de una estrategia de desarrollo incremental y el solapamiento de las distintas fases del desarrollo, en vez de realizar cada una de estas fases una tras otra en un ciclo secuencial o en cascada.

El sprint es el núcleo central de esta metodología y se trata de cada uno de los cinco ciclos o iteraciones que se han tenido en el proyecto. Su objetivo es conseguir un incremento de valor en él. Cada uno de ellos se dividió en cinco fases:

- **Selección de las historias de usuario a implementar y descomposición en tareas:** análisis de las necesidades y características asociadas a las historias de usuario, y especificación de lo que debe hacer el sistema, sin entrar en detalles técnicos y desde la perspectiva del usuario.

- **Diseño:** descripción de la estructura interna del software y las relaciones entre las entidades que lo componen necesarias para desarrollar las historias de usuario del sprint.

- **Implementación:** se realiza la programación de cada una de las tareas especificadas teniendo en cuenta el diseño realizado en la fase anterior.

- **Verificación y pruebas:** comprobación de que todas las características implementadas en el sistema funcionen correctamente.

- **Documentación:** actualización progresiva de la memoria del proyecto con todo lo desarrollado en el sprint.

Cada sprint desarrolla una serie de historias de usuario, que se corresponden con un conjunto de requisitos que permiten al usuario realizar una funcionalidad. El proyecto se dividió en cinco de estos, con una duración de cincuenta y nueve horas cada uno.

El tutor asumió el papel de "Product Owner" y las reuniones típicas de esta metodología han sido tutorías que se hicieron con regularidad.

1.4 Software y tecnologías usadas

Al plantear el software y las tecnologías a utilizar en el proyecto, siempre se ha apostado por el software libre. A continuación vamos a exponer las tecnologías usadas en orden alfabético:

- **Android Studio [6]**

Figura 1: Logo de Android Studio

El entorno integrado de desarrollo libre para la plataforma Android oficial de Google. Se ha usado para el desarrollo de la aplicación cliente Android, en su versión 4.1.3.

- **Android SDK [7]**

Figura 2: Logo de Android SDK

Conjunto de herramientas de desarrollo oficial de la plataforma Android. Se ha usado la versión 11 (API 30) para el desarrollo de la aplicación de Android.

- **Diagrams.net** [8]

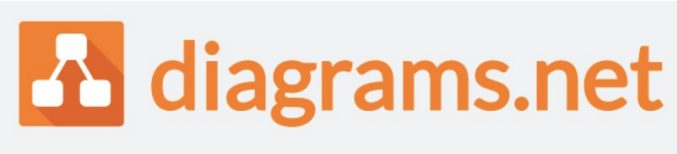

Figura 3: Logo de Diagrams.net

Este SaaS integrado en Google Drive se ha usado para la realización de los diagramas de secuencia y de flujo de navegación.

- **Docker** [9]

Figura 4: Logo de Docker

Esta herramienta se usa para automatizar el despliegue de la aplicación servidor dentro de dos contenedores software: el servidor de aplicaciones y la base de datos.

- **EclipseJEE** [10]

Figura 5: Logo de Eclipse

Este entorno de desarrollo integrado para JavaSE y JavaEE se utilizó, en su versión 2020-12, para desarrollar tanto la aplicación servidor como la aplicación cliente de escritorio.

- **Java EE** [11]

Figura 6: Logo de JavaEE

Esta plataforma de programación de aplicaciones empresariales se usó para el desarrollo de toda la aplicación servidor debido a la gran amplitud de especificaciones que incluye y a la buena cantidad de documentación que existe sobre ella.

- **Java FX** [12]

Figura 7: Logo de JavaFX

Este Framework de Java sirve para crear interfaces gráficas y en este proyecto se usó para el desarrollo de la aplicación de escritorio.

- **Java SE** [13]

Figura 8: Logo de JavaSE

Colección de API del lenguaje de programación Java, usada tanto en la aplicación servidor como en las aplicaciones cliente.

- **JMeter [14]**

Figura 9: Logo de JMeter

Herramienta de pruebas centrada en el ámbito de los servicios web. Usada en su versión 5.4.1 para las pruebas del servicio REST de la aplicación servidor.

- **Marathon/OSS [15]**

Figura 10: Logo de Marathon/OSS

Entorno de integrado para pruebas de sistema en aplicaciones Java Swing y JavaFX. Usado en su versión OSS 5 para las pruebas de sistema de la aplicación de escritorio.

- **MySQL [16]**

Figura 11: Logo de MySQL

El sistema gestor de base de datos relacionales libre más popular del mundo. Se usa como sistema de base de datos de la aplicación servidor.

- **Scene Builder [17]**

Figura 12: Logo de Scene Builder

Scene Builder es un programa muy útil para la creación rápida y cómoda de interfaces de usuario JavaFX. Usado en su versión 15.0.1 para la generación de las vistas de la aplicación de escritorio.

- **Selenium IDE [18]**

Figura 13: Logo de Selenium IDE

Entorno de pruebas software para aplicaciones web. Permite grabar la interacción del usuario con la interfaz gráfica y definir pruebas de sistema a partir de estas grabaciones. Usado en su versión 3.17.0 para la definición de pruebas de sistema del panel de administración web de la aplicación servidor.

- **SQLite [19]**

Figura 14: Logo de SQLite

Sistema gestor de base de datos relacionales embebido. Es muy ligero y es usado tanto en la aplicación de escritorio como en la de Android.

- **Wildfly [20]**

Figura 15: Logo de Wildfly

Servidor de aplicaciones JavaEE, el cual es utilizado en su versión 21 para el despliegue de la aplicación servidor.

2

Historias de Usuario

Una historia de usuario [21] es la representación de un requisito funcional, expresado en una frase en el lenguaje común del usuario, sin entrar en detalles técnicos. Son muy utilizadas para realizar la especificación de los requisitos en las metodologías ágiles de desarrollo como en las que se ha desarrollado este trabajo de fin de grado, Scrum.

En este capítulo enumeraremos y describiremos cada una de las historias de usuario que se han seleccionado para el desarrollo de este proyecto.

2.1 CRUD Usuarios

Como usuario administrador, debería poder crear, ver, modificar y eliminar usuarios desde la interfaz WEB.

2.2 Validar acceso a la aplicación

Como usuario, mediante un alias y una contraseña, debería o no interactuar en el sistema dependiendo si mis credenciales son correctas o no.

2.3 Registro de nuevo usuario

Como usuario, debería poder registrarme en la aplicación servidor (siempre y cuando esté el registro activo a nivel de configuración).

2.4 CRUD Contactos en las aplicaciones cliente

En las aplicaciones cliente, como usuario debería poder ver, crear, actualizar y eliminar contactos.

2.5 Notificación a todos los usuarios de un servidor

Como usuario administrador de un servidor, debería poder enviar correos electrónicos destinados a todos los usuarios registrados en el servidor desde la interfaz WEB de administración.

2.6 Configuración de la aplicación servidor

Como usuario administrador de un servidor, debería poder configurar algunas opciones de este como activar/desactivar el registro de nuevos usuarios, la recepción de nuevos mensajes, cuenta de correo saliente, etc.

2.7 Consultar mensajes de un chat

Como usuario, debería poder consultar todos los mensajes de un chat, tanto de texto como archivo, incluso cuando este no esté conectado a la red.

2.8 Enviar mensaje privado

Como usuario, debería poder enviar un mensaje privado (texto o archivo) a cualquiera de sus contactos. Ese mensaje será firmado y cifrado automáticamente antes del envío de forma transparente al usuario.

2.9 Recibir mensaje privado

Como usuario, debería recibir automáticamente mis mensajes privados nuevos (tanto texto como archivo) cuando esté disponible y conectado a mi servidor. Si no estoy conectado, mis mensajes se almacenarán en la base de datos del servidor a la espera de que me conecte. Cuando me conecte, el sistema me enviará todos mis mensajes nuevos y estos se borrarán automáticamente del servidor. Mi aplicación cliente comprobará la firma y descifrará automáticamente el mensaje recibido de forma transparente a mí.

2.10 Crear chat grupal

Un chat grupal, es un chat en el que varios usuarios podrán enviar mensajes e interactuar entre ellos en ese mismo chat.

Como usuario, debería poder crear salas de chat grupal y añadir como miembro a cualquiera de mi contactos.

2.11 Editar chat grupal

Como usuario debería poder modificar la información de mis salas de chat grupal: su nombre y el listado de miembros.

2.12 Eliminar chat grupal

Como usuario, debería poder eliminar los grupos de chat que he creado previamente.

2.13 Enviar mensaje grupal

Como usuario, debería poder enviar un mensaje grupal (texto o archivo) a cualquiera de los grupos de los que soy miembro. Ese mensaje será firmado y cifrado

automáticamente para cada uno de los miembros del grupo antes del envío y de forma transparente para mí.

2.14 Recibir mensaje grupal

Como usuario, debería recibir mis mensajes grupales nuevos cuando esté disponible y conectado a mi servidor. Si no estoy conectado, los mensajes se almacenarán en la base de datos a la espera de que me conecte. Cuando me conecte, el sistema me enviará todos mis mensajes nuevos y estos se borrarán automáticamente del servidor. Mi aplicación cliente comprobará la firma y descifrará automáticamente el mensaje recibido de forma transparente a mí y, posteriormente, lo clasificará en el chat de grupo que corresponda.

2.15 Crear lista de difusión

Una lista de difusión es una lista de contactos que se guarda en la aplicación. Al escribir un mensaje usando una lista de difusión, ese mensaje llegará a cada uno de los contactos pertenencientes a la lista, en vez de tener que mandar de forma manual un mismo mensaje a todos esos contactos de uno en uno.

Como usuario, debería poder crear listas de difusión con el objetivo de enviar un mismo mensaje a múltiples usuarios a la vez.

2.16 Modificar lista de difusión

Como usuario, debería poder modificar mis listas de difusión y añadir o quitar contactos de destino.

2.17 Eliminar lista de difusión

Como usuario, debería poder eliminar mis listas de distribución que haya creado previamente.

2.18 Enviar mensaje de difusión

Como usuario, debería poder enviar un mensaje de difusión usando una lista de difusión que haya creado previamente, con el objetivo de enviar ese mismo mensaje a todos los contactos pertenecientes a la lista.

3

Diseño

En este capítulo hablaremos del diseño del sistema, es decir, la descripción de la estructura interna de las aplicaciones y las relaciones entre las entidades que la componen y que son necesarias para desarrollar las historias de usuario del sprint.

3.1 Diagramas Entidad-Relación [22]

Los diagramas Entidad-Relación sirven para describir el diseño una base de datos, representando las entidades y las relaciones que se producen entre estas.

En este proyecto deberemos diferenciar dos bases de datos diferentes: la base de datos de la aplicación servidor y la de las aplicaciones cliente. Si bien se desarrollarán dos aplicaciones cliente diferentes (una para la plataforma Android y otra de escritorio), la estructura de estas bases de datos son idénticas, por lo tanto no haremos diferencia entre ellas a la hora de definirlas en este capítulo.

Para la aplicación servidor definimos el siguiente diagrama entidad-relación:

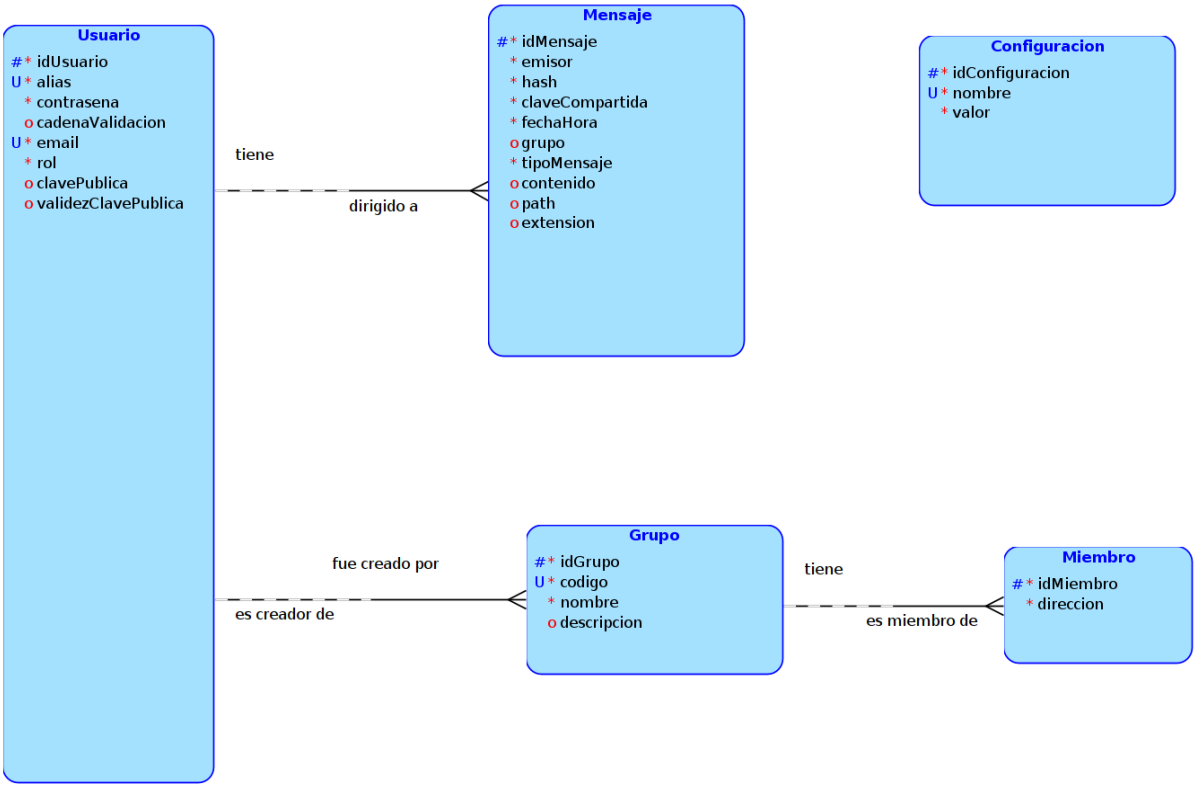

Figura 16: Diagrama Entidad-Relación de la aplicación servidor

A continuación procederemos a explicar las entidades y relaciones representadas en la figura anterior:

Entidades:

- **Usuario**

  Esta entidad representa la información de los distintos usuarios del servidor.

- **Mensaje**

  Esta entidad representa los mensajes dirigidos a los usuarios del servidor. En la base de datos del servidor solo se almacenará un mensaje si en el momento de que este reciba el mensaje no está el usuario destinatario conectado. Si se trata de un mensaje privado, la columna "grupo" será null. En cambio, si se trata de un mensaje grupal, en la columna "grupo" iría el identificador de grupo. Habrá varios tipos de mensaje diferentes: texto, archivo, actualización

de clave y actualización de grupo. En el caso de un mensaje tipo archivo, además, se incluirá en la columna "path" el path con la ubicación del archivo en el sistema de ficheros del servidor y la extensión de este en la columna "extension".

- **Grupo**

  Esta entidad representa los grupos de chat que se encuentren alojados en el servidor, es decir, aquellos grupos que han sido creados por usuarios del servidor. Cada grupo poseerá un código de grupo a modo de identificador de grupo para los usuarios. Un identificador de grupo sería el siguiente:

  SERVIDOR*/rs/grupo/****CODIGO-GRUPO***

  Donde ***CODIGO-GRUPO*** es el valor de la columna "codigo" y servidor es el nombre de dominio del servidor.

- **Miembro**

  Esta entidad representa los miembros pertenecientes a grupos.

- **Configuración**

  Esta entidad representa la configuración del servidor. Cada una de los parámetros de configuración tendrán un nombre y un valor asociado.

Relaciones:

- **Usuario – Mensaje:** Relación de uno a muchos. Un usuario puede tener varios mensajes, y cada mensaje esta dirigido solo a un usuario.

  La entidad "Mensaje" tiene el atributo "grupo" pero no está relacionada con la entidad "Grupo" porque en esta última, solo se almacenan los grupos pertenecientes al servidor. Un mensaje de un usuario podría pertenecer a un

grupo que se encuentre alojado en otro servidor diferente. Es por ello que se realiza la relación de esta forma más general.

- **Usuario – Grupo:** Relación uno a muchos. Un usuario puede haber creado varios grupos, y cada grupo es creado por un único usuario.

- **Grupo – Miembro:** Relación uno a muchos. Un grupo puede estar compuesto por varios miembros. No se usa una relación Grupo – Usuario en vez de esta para representar los miembros de los grupos, ya que puede que haya miembros del grupo que se encuentren registrados en un servidor diferente.

Para las aplicaciones cliente podemos definir el siguiente diagrama entidad-relación:

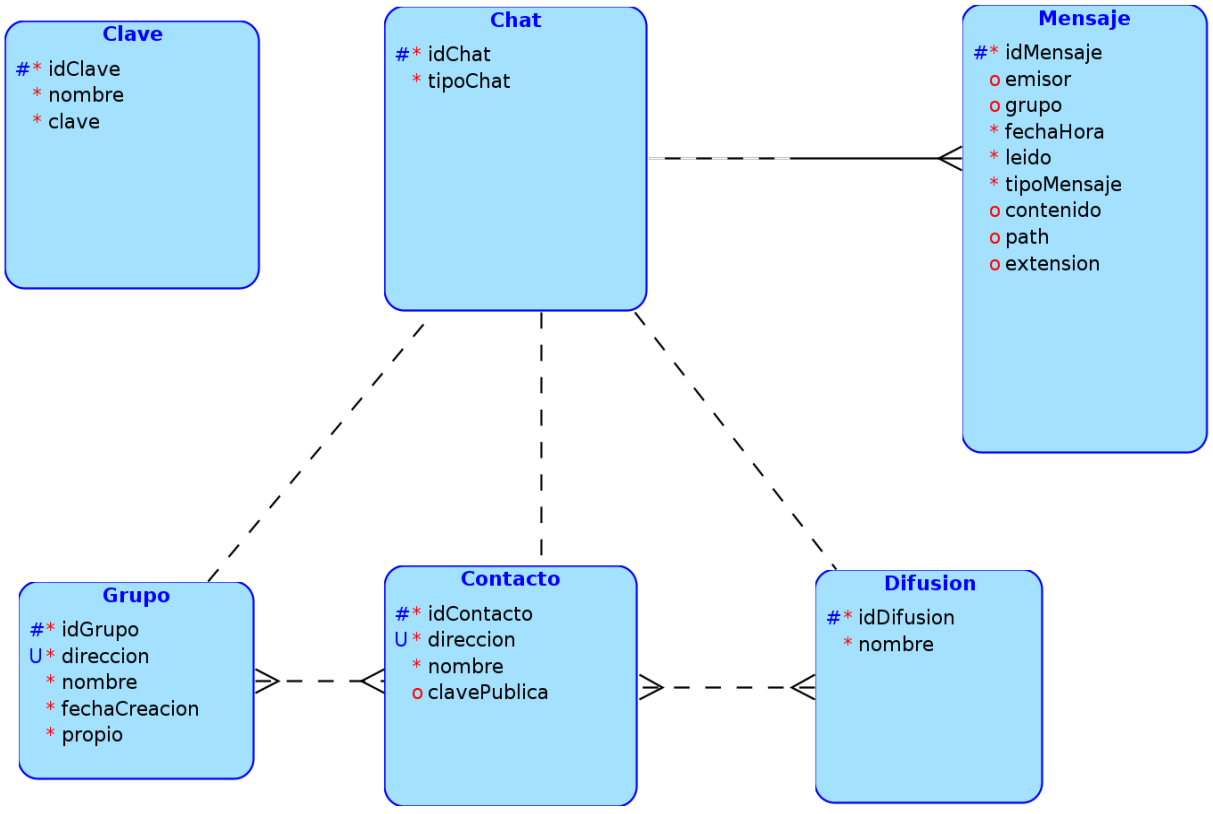

Figura 17: Diagrama Entidad-Relación de las aplicaciones cliente

Ahora explicaremos con más detalle estas entidades y relaciones entre ellas:

Entidades

- **Chat:** Esta entidad representa los diferentes chats o conversaciones que tendrá el usuario. Podrán ser de tres tipos diferentes: privado, grupal o de difusión.

- **Mensaje:** Esta entidad representa los mensajes. Si la columna emisor está a null, significa que el mensaje ha sido redactado por el propio usuario. Un mensaje podrá ser de varios tipos: texto, archivo, actualización de grupo o actualización de clave. Si el mensaje es de tipo texto, el contenido de este se almacenará en la columna "contenido". Si se trata de un mensaje tipo archivo, el archivo en sí se almacenará en el sistema de ficheros del dispositivo, y en la columna "path" se incluirá la dirección donde se ha guardado en este.

- **Grupo:** Esta entidad representa los grupos a los que pertenece el usuario. Cada grupo poseerá una dirección de grupo, que se corresponderá con una URI de este tipo:
  SERVIDOR/rs/grupo/CODIGO-GRUPO
  Además, la columna "propio" indicará si el grupo es creado o no por el usuario. Ser creador del grupo le permitirá editar y eliminar el grupo.

- **Contacto:** Esta entidad representa a los contactos del usuario. El contacto tendrá un nombre y su dirección, que será tipo: **alias@servidor.**

- **Difusión:** Esta entidad representa a las listas de difusión del usuario.

- **Clave:** Esta entidad representará a las claves del usuario. En concreto la clave pública y la clave privada.

Relaciones

Dependiendo del tipo de chat (privado, grupal o de difusión), este estará relacionado con la entidad contacto, grupo o difusión.

- **Chat – Grupo:** Relación uno a uno sin obligatoriedad. Un grupo podrá tener (o no) una conversación asociada a él.
- **Chat – Contacto:** Relación uno a uno sin obligatoriedad. Un contacto podrá tener (o no) una conversación asociada a él.
- **Chat – Difusión:** Relación uno a uno sin obligatoriedad. Una lista de difusión podrá tener (o no) una conversación asociada a ella.
- **Chat – Mensaje:** Relación uno a muchos, con el muchos por el lado de Mensaje. Un chat o conversación puede tener cero, uno o varios mensajes, y un mensaje tiene que pertenecer obligatoriamente a un chat.

3.2 Diagramas de secuencia [23]

Vamos a utilizar diagramas de secuencia para modelar la interacción de objetos en la realización de algunos procesos del sistema.

A continuación se muestra el diagrama de secuencia del proceso de login, en el que el usuario introducirá sus credenciales en la aplicación cliente, esta se comunicará con el servicio REST para comprobar que son correctas y, si lo son, se abrirá la conexión del websocket. Posteriormente el servicio de websocket mandará a la aplicación cliente los mensajes nuevos que tenga el usuario (en el caso de que tenga) y, la aplicación cliente, comprobará que la clave pública del usuario guardada en la base de datos local coincida con la de la base de datos del servidor. Si no coincidieran, la

aplicación cliente generaría un nuevo par de claves, subiría la nueva clave pública al servidor y mandaría un mensaje de tipo actualización de clave a todos sus contactos.

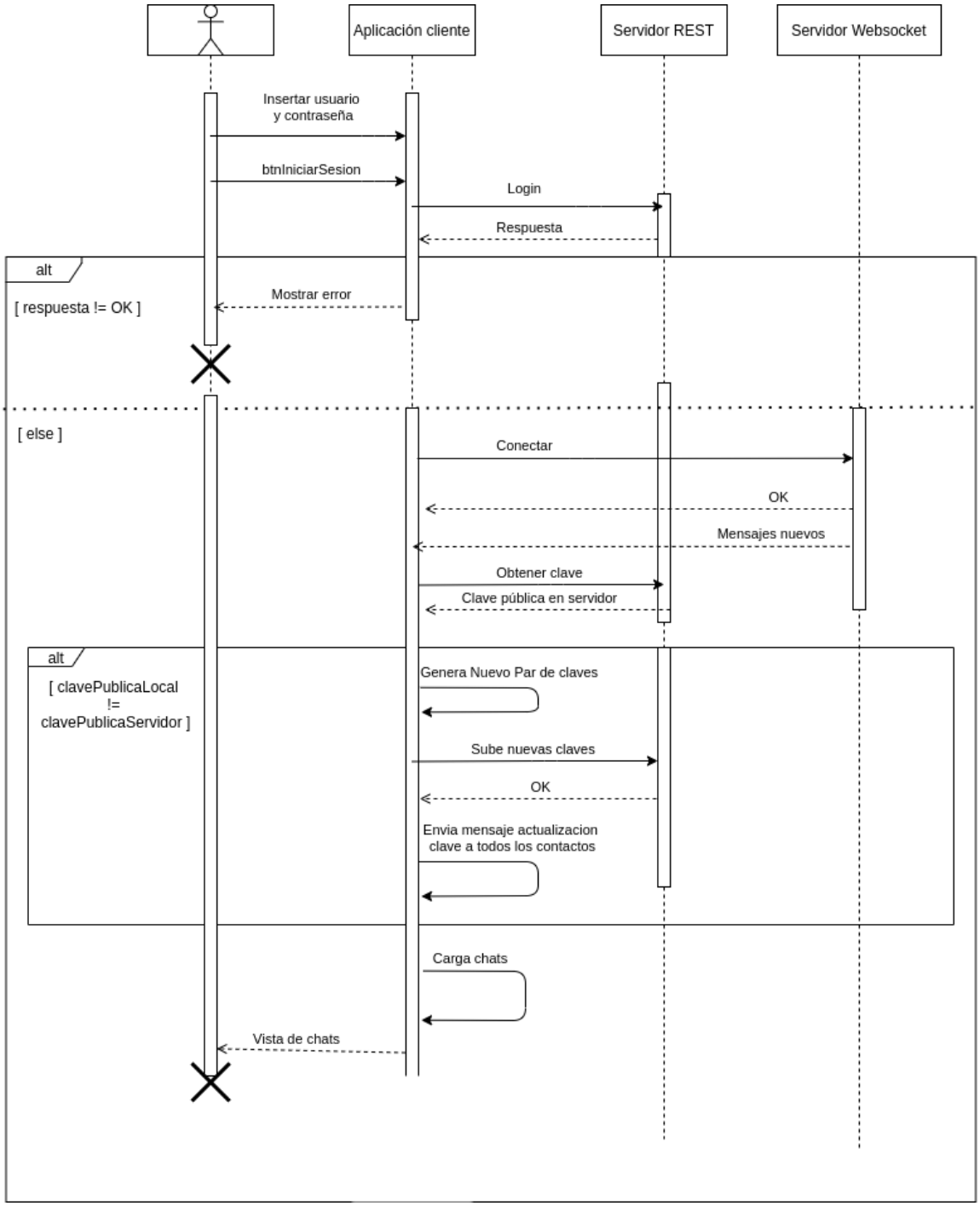

Figura 18: Diagrama de secuencia del proceso de login

Ahora veremos el diagrama de secuencia del proceso de agregar un contacto, en el que el usuario introduce un nombre de contacto y su dirección (alias@servidor), la aplicación cliente obtiene la clave pública de ese contacto haciendo una petición al servicio REST del servidor del contacto y, por último, lo guarda en la base de datos local de la aplicación cliente.

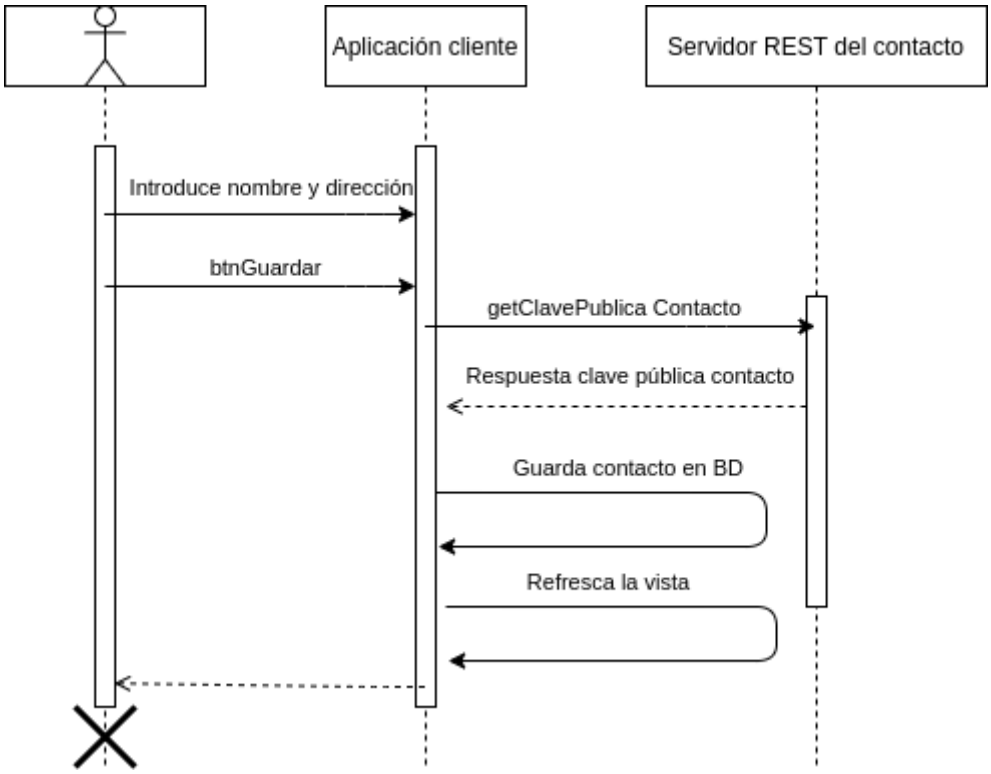

Figura 19: Diagrama de secuencia del proceso de agregar un contacto

A continuación se muestra el diagrama de secuencia del proceso de enviar un mensaje. En este proceso, el usuario escribirá un mensaje y, al pulsar el botón de enviar, la aplicación cliente lo cifrará y lo mandará por REST al servidor del contacto.

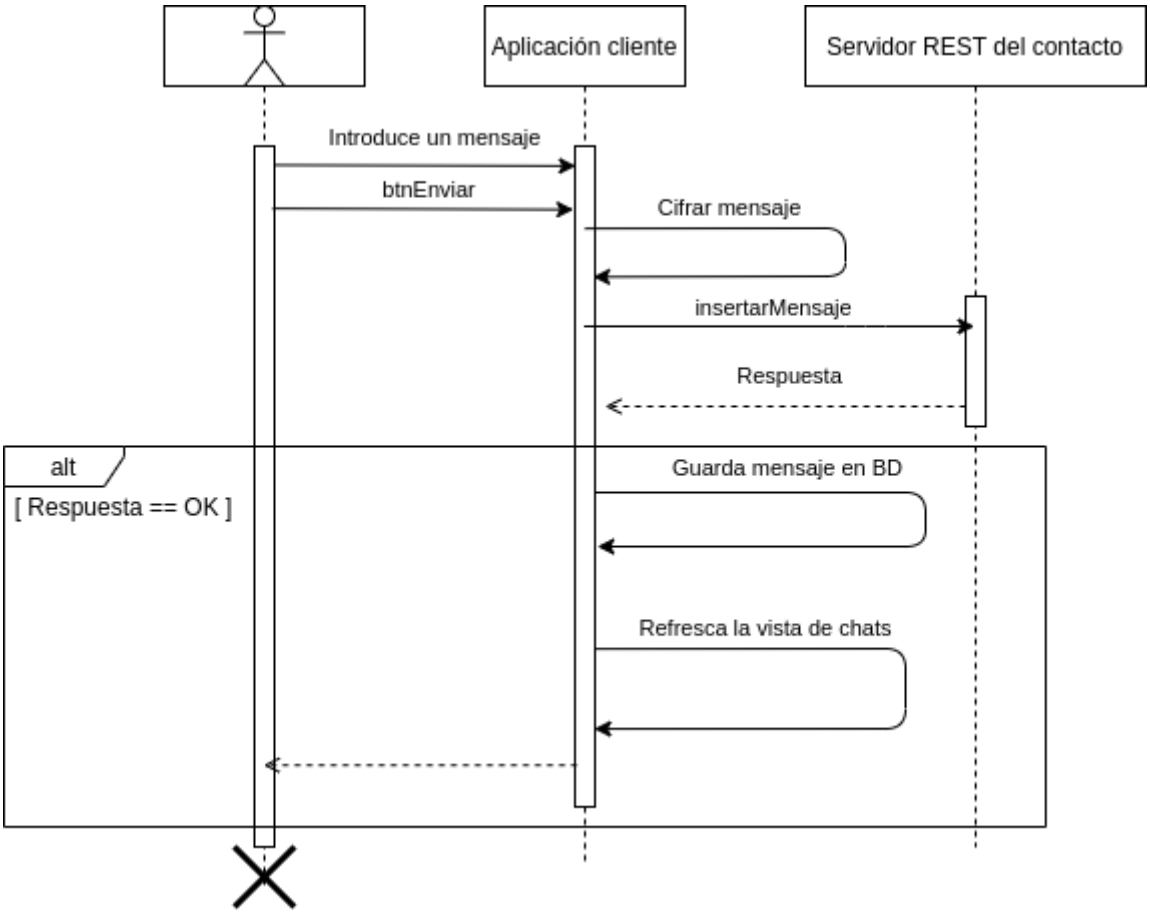

Figura 20: Diagrama de secuencia del envío de un mensaje

Ahora veremos el diagrama de secuencia de la recepción de un mensaje. En este proceso, el servidor enviará por websocket a la aplicación cliente el mensaje, y esta, comprobará que el emisor de este nuevo mensaje sea uno de los contactos del usuario. Si no lo es, el mensaje quedaría descartado, evitando así la recepción de mensajes no deseados o de spam. Si lo fuese, se descifraría el mensaje, se guardaría en la base de datos local de la aplicación cliente y se notificaría al usuario.

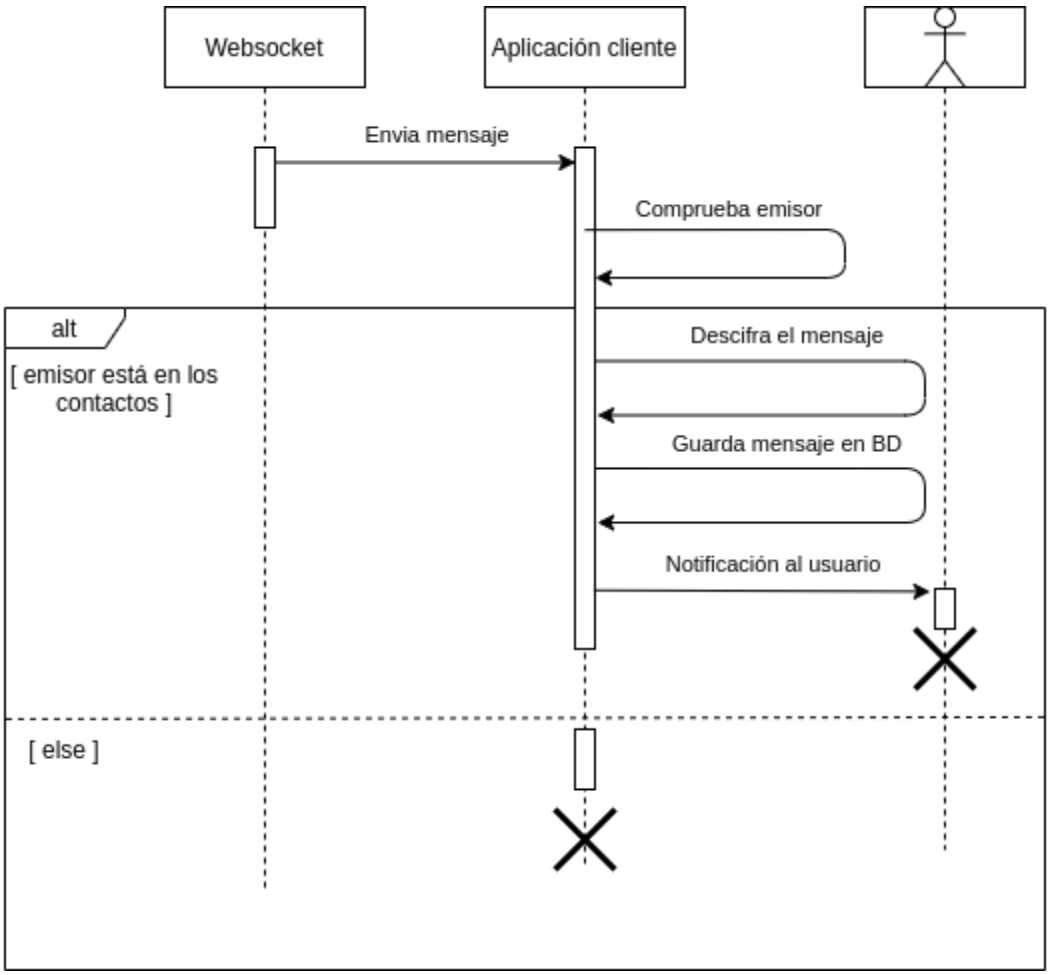

Figura 21: Diagrama de secuencia de la recepción de un mensaje nuevo

Seguidamente se muestra el diagrama de secuencia del proceso de actualización de la clave de un contacto. Este proceso se realizará de forma totalmente transparente al usuario, y se iniciará cuando la aplicación cliente reciba un mensaje de tipo actualización de claves. Al recibir un mensaje de este tipo, la aplicación cliente pedirá al servicio REST del servidor del contacto emisor del mensaje su clave pública, y posteriormente actualizará el contacto de la base de datos local con la nueva clave.

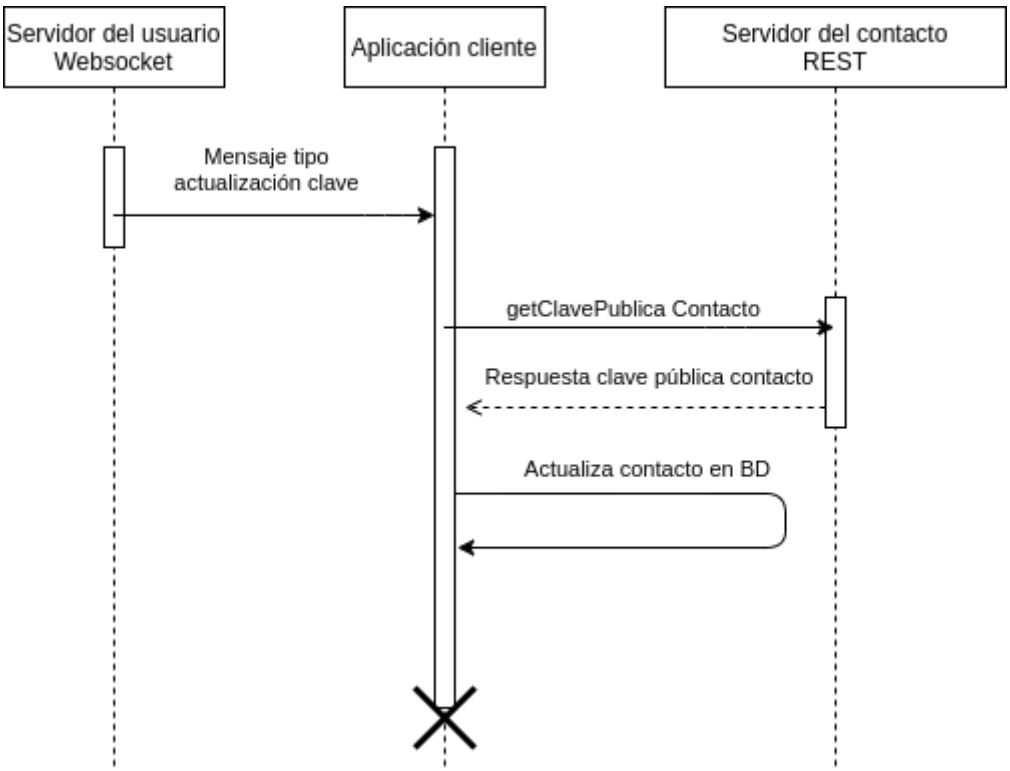

Figura 22: Diagrama de secuencia de la actualización automática de la clave de un contacto

Ahora veremos el diagrama de secuencia del proceso de creación de un grupo. En este proceso, el usuario introduce en la aplicación cliente un nombre de grupo y los contactos que van a ser miembros del grupo. A continuación la aplicación cliente hace una petición de creación de grupo al servicio REST del servidor del usuario. El servidor creará el grupo y responderá a la aplicación cliente con el código de grupo que le ha asignado. La aplicación cliente guarda el grupo en la base de datos local con el código indicado por el servidor, y finalmente, manda un mensaje de tipo actualización de grupo a todos los miembros de este.

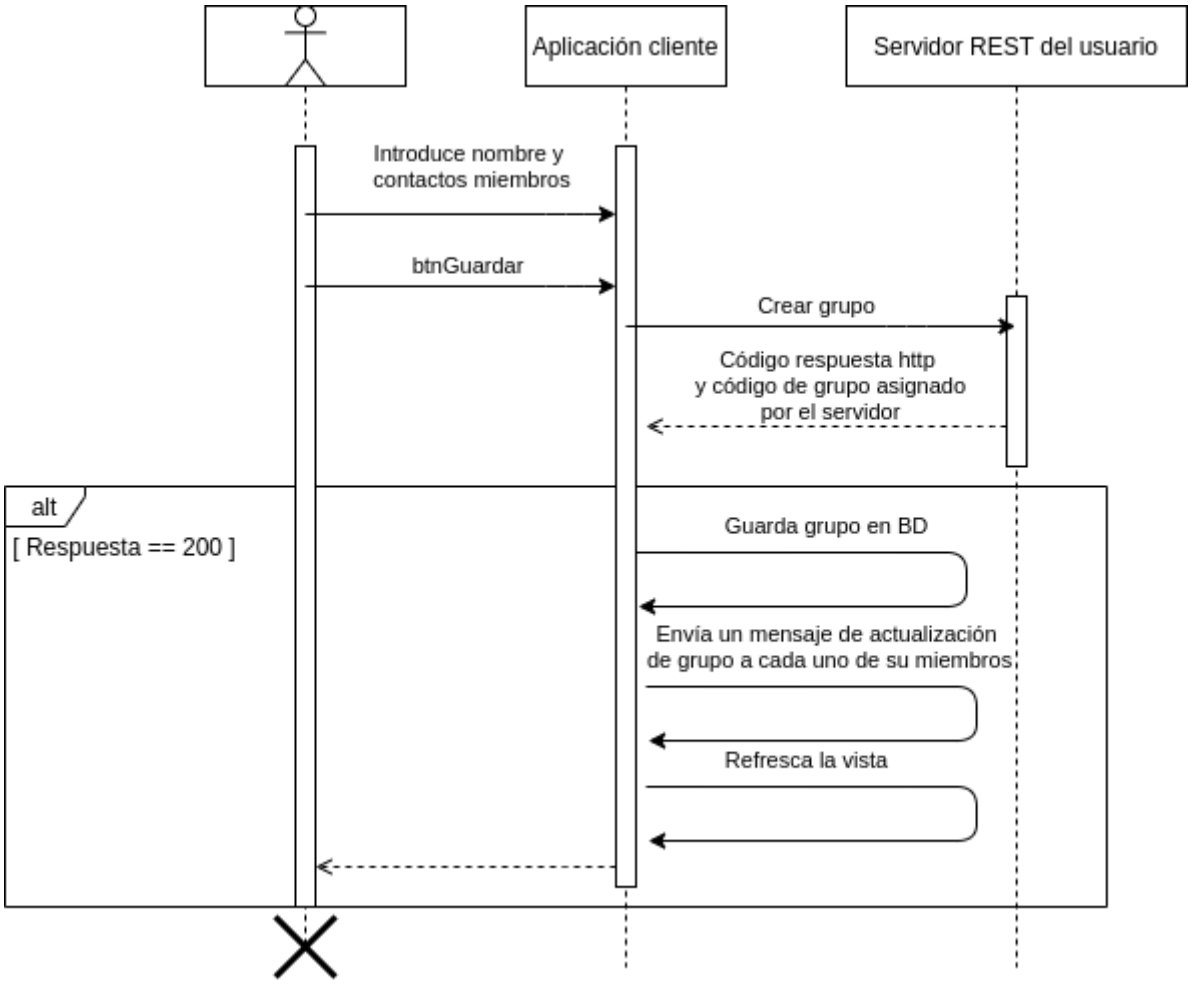

Figura 23: Diagrama de secuencia del proceso de crear un grupo

A continuación veremos el diagrama de secuencia del proceso de editar un grupo. En este proceso, el usuario creador editará el grupo (cambiar el nombre y/o añadir o quitar miembros), y la aplicación cliente realizará una petición REST de editar el grupo al servidor del usuario. Finalmente la aplicación cliente actualizará el grupo en su base de datos local y enviará un mensaje de actualización de grupo a todos los miembros de este.

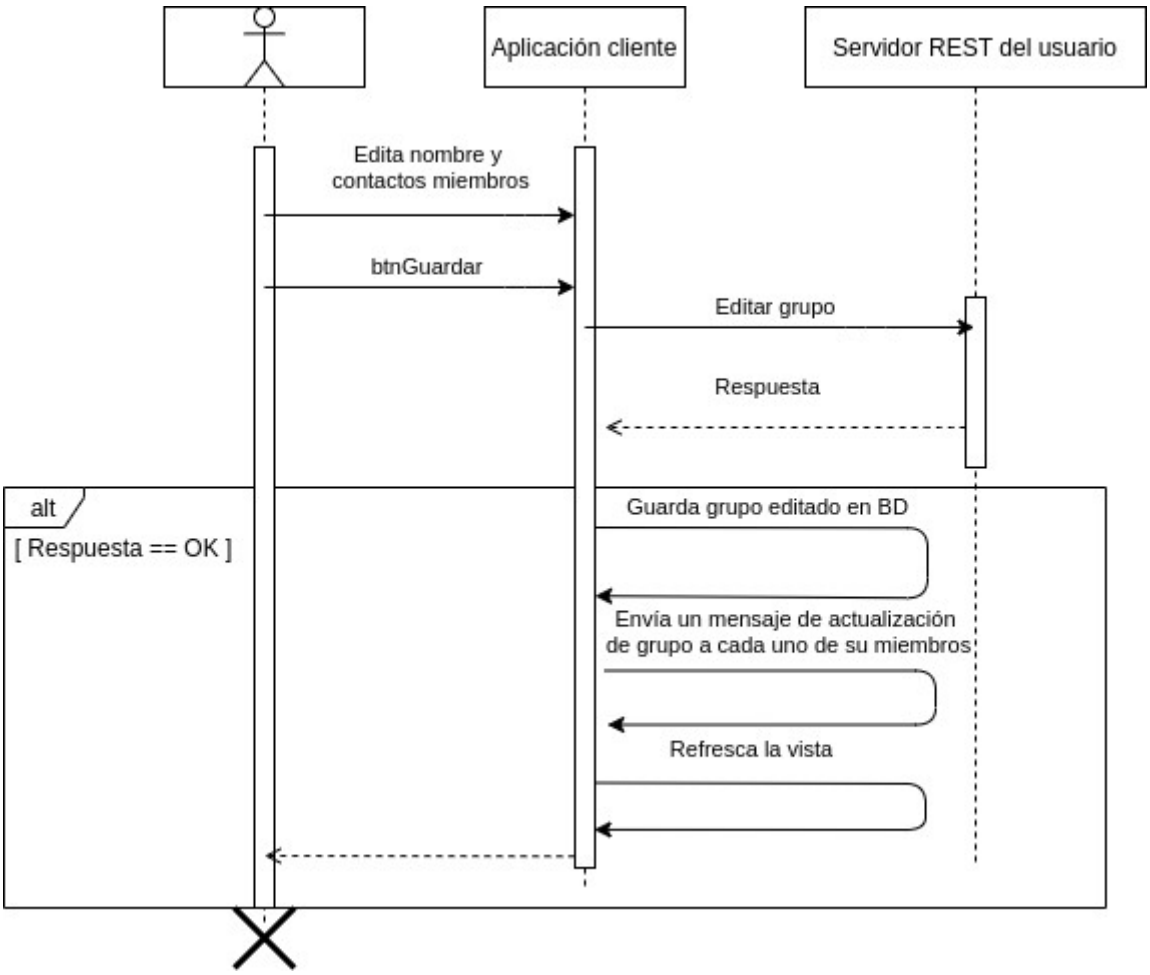

Figura 24: Diagrama de secuencia de la edición de un grupo

Finalmente veremos el diagrama de secuencia del proceso de actualización automática de grupo. Este proceso, transparente para el usuario, se realiza cuando la aplicación cliente recibe un mensaje de tipo actualización de grupo. El servidor del usuario envía a la aplicación cliente el mensaje por websocket. La aplicación cliente usa el identificador de grupo, que va contenido en este mensaje, para pedir al servidor que almacena este grupo su información por REST. Una vez obtenida la información, se actualiza la información del grupo y sus miembros en la base de datos local de la aplicación cliente.

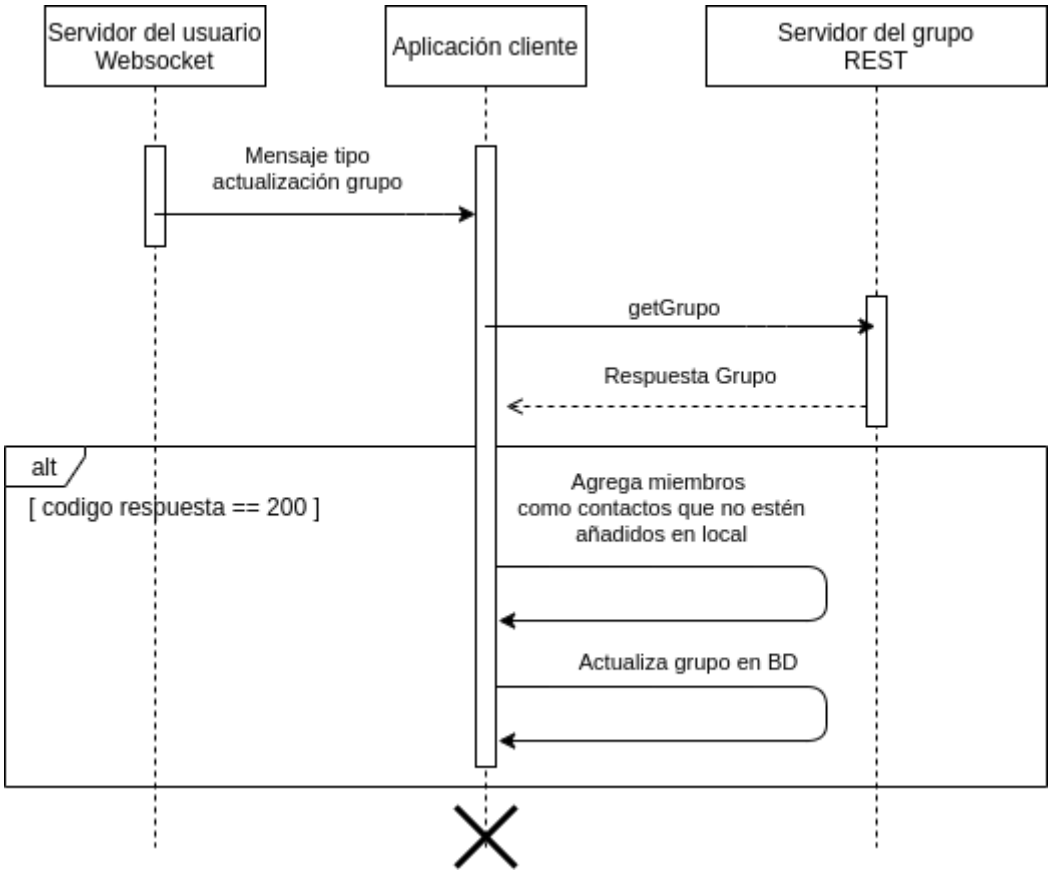

Figura 25: Diagrama de secuencia de la actualización automática de un grupo

3.3 Cifrado y descifrado

A continuación vamos a describir los procesos de cifrado y descifrado de mensajes.

- **Cifrado**

En el proceso de cifrado, primero se calcula el hash del contenido del mensaje. Una vez calculado, ciframos el hash con la clave privada del usuario (el emisor) para firmarlo y lo añadimos al objeto mensaje de salida de la función. A continuación generamos una clave aleatoria y ciframos el contenido del mensaje con ella. Una vez realizado esto, añadimos el contenido cifrado al mensaje de salida. Por último, se cifra la clave aleatoria (generada anteriormente) con la clave pública del contacto destinatario del mensaje y la añadimos al mensaje de salida.

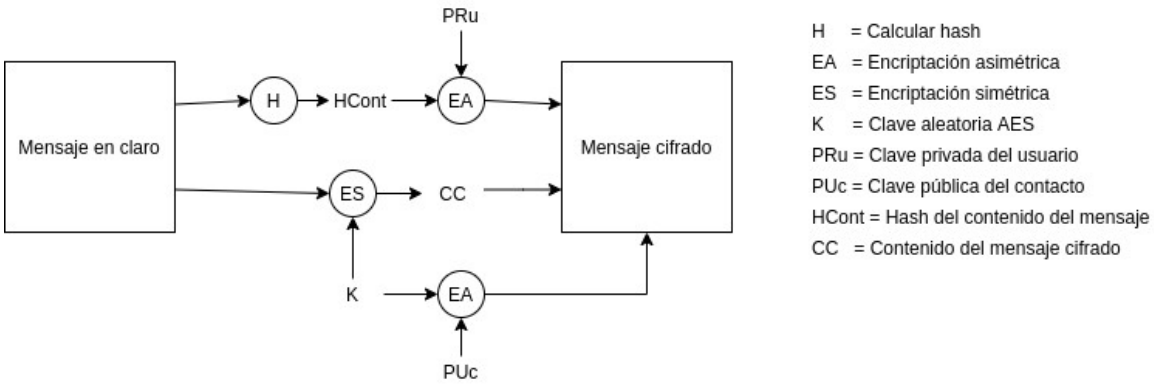

Figura 26: Diagrama del proceso de cifrado

- **Descifrado**

En el proceso de descifrado, primero desciframos la clave aleatoria compartida usando la clave privada del usuario (receptor del mensaje). A continuación, se descifra con criptografía simétrica el contenido del mensaje con la clave aleatoria ya descifrada. Posteriormente, desciframos el hash (hash descifrado) del contenido del mensaje usando la clave pública del contacto (emisor del mensaje). Una vez descifrado el hash, se calcula otro hash (hash calculado) del contenido del mensaje ya descifrado y, por último se comparan estos dos, el hash descifrado y el hash calculado.

Si ambos son iguales, la comprobación de firma es correcta. Si no lo son, no se puede garantizar la autoría del emisor y, por lo tanto, se devuelve un error.

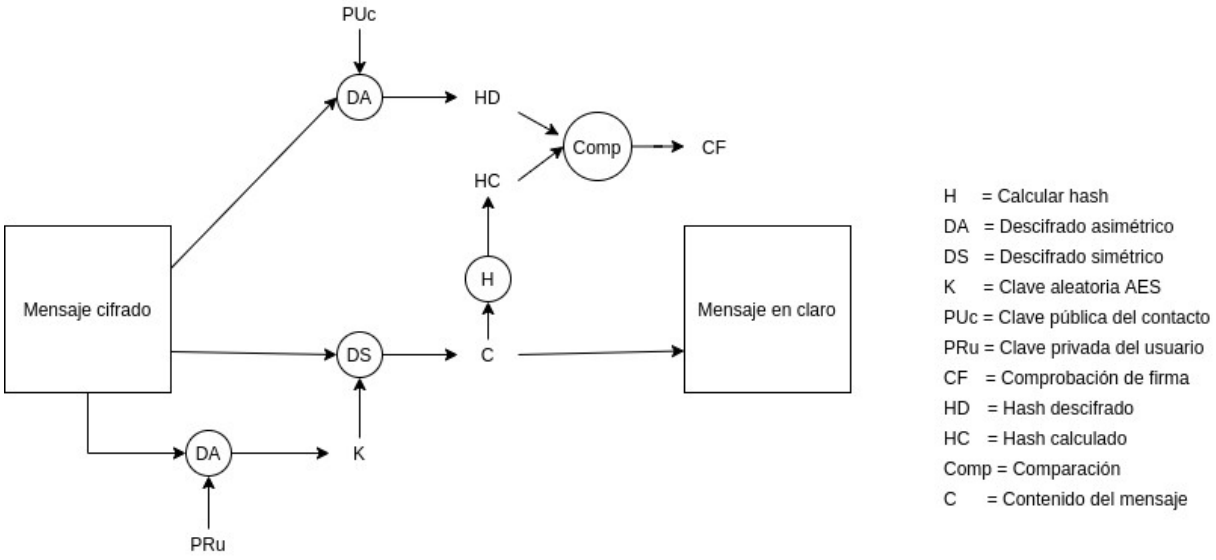

Figura 27: Diagrama del proceso de descifrado

3.4 Diagramas de flujo de navegación [24]

Los diagramas de flujos de navegación son muy útiles para representar la estructura de la navegación de una interfaz gráfica. Vamos a usar este tipo de diagramas para representar las interfaces gráficas del panel de administración web del servidor y las dos aplicaciones cliente. Cada recuadro representará una vista, y las flechas indicarán las transiciones entre una y otra.

- **Navegación de la interfaz web del servidor**

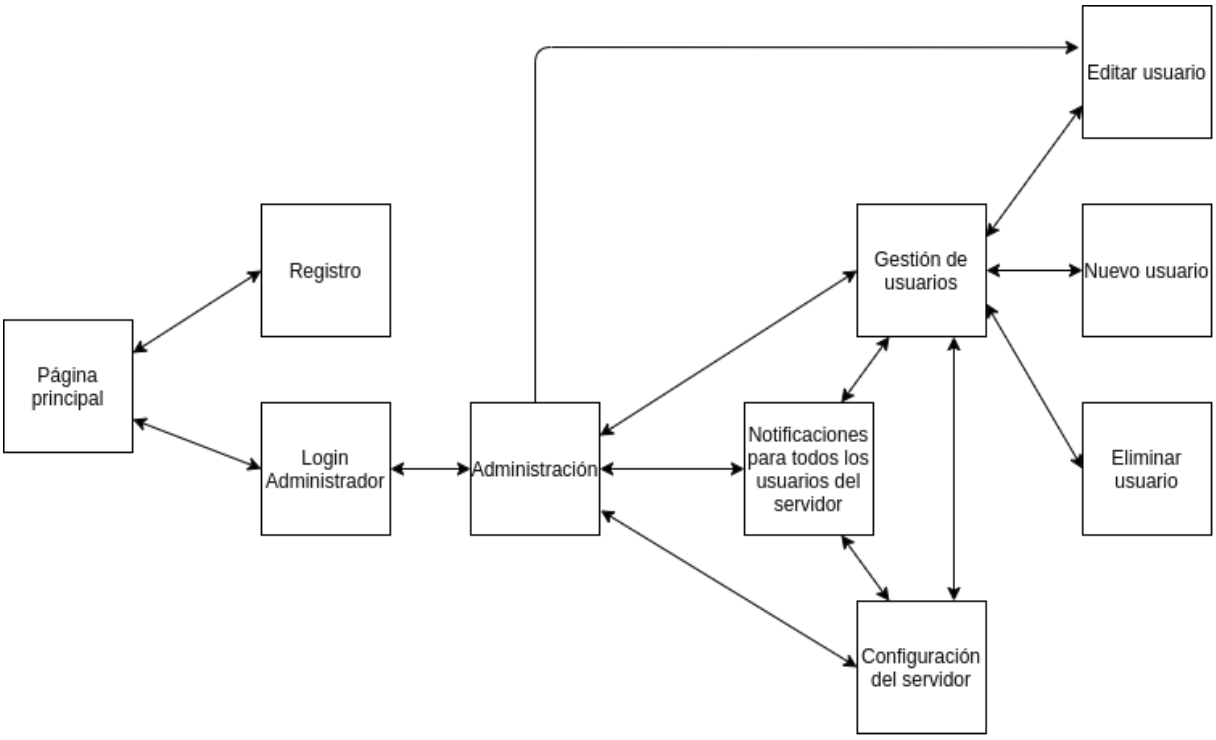

Figura 28: Diagrama de flujo de navegación de la interfaz web del servidor

- **Navegación de la aplicación cliente de escritorio**

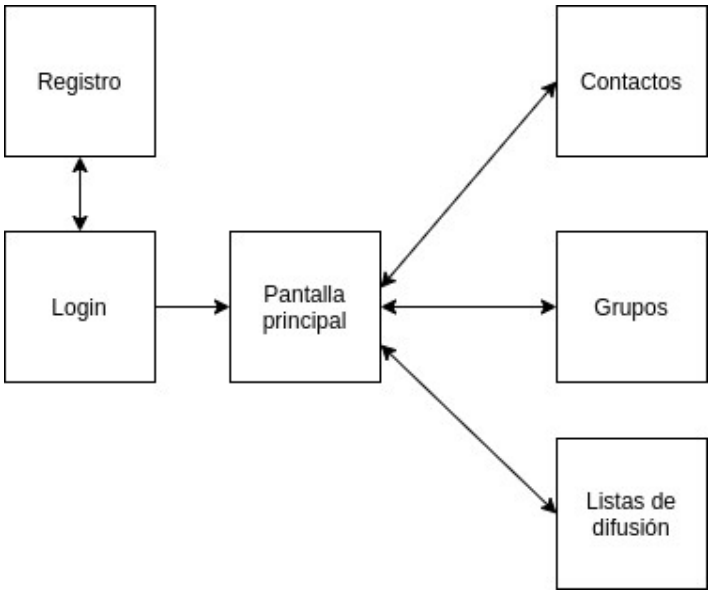

Figura 29: Diagrama de flujo de navegación de la aplicación de escritorio

- **Navegación de la aplicación cliente Android**

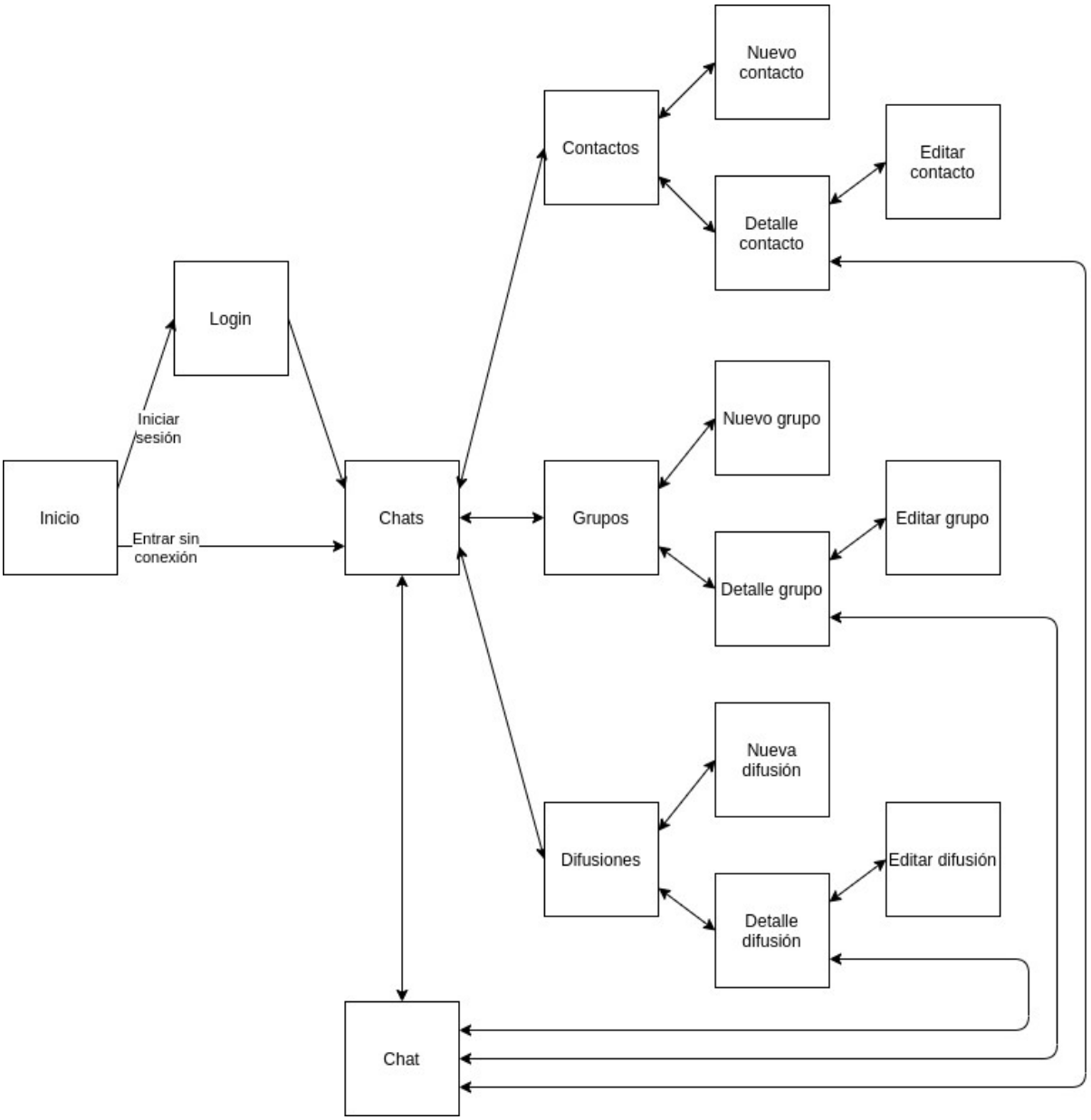

Figura 30: Diagrama de flujo de navegación de la aplicación Android

4

Verificación

4.1 Aplicación servidor

Se han realizado dos tipos de pruebas diferentes para la verificación de la calidad de la aplicación servidor. Por un lado se ha probado la funcionalidad de la API REST usando JMeter, y por otro, el funcionamiento correcto de la interfaz de administración web del servidor usando Selenium.

4.1.1 JMeter

Por cada uno de los métodos de la API REST del servidor se ha definido varios tests. Cada uno de estos se dará por válido si el servidor responde con el código esperado.

Los test implementados son los siguientes:

- **Login correcto**

  En esta prueba, se consume el servicio de login con el usuario y la contraseña de la cuenta de administrador que se crea por defecto al arrancar por primera vez el servidor. Se da por correcto si el código de respuesta es 200.

- **Login incorrecto**

En este test se consume el servicio de login con unas credenciales incorrectas. El test se da por aprobado si el código de respuesta es 401.

- **Get Clave Pública Correcto**

  En esta prueba, se consume el servicio de petición de la clave pública de un usuario existente en la base de datos del servidor. Se da por correcto si el código de la respuesta es 200.

- **Get Clave Pública Usuario Inexistente**

  En esta prueba, se consume el servicio de petición de la clave pública de un usuario no existente en la base de datos del servidor. Se da por correcto si el código de la respuesta es 404.

- **Insertar Clave Pública Correcto**

  En esta prueba, se consume el servicio de subida de clave pública de un usuario al servidor, con la clave a subir y las credenciales de un usuario existente en la base de datos del servidor. Se da por correcto el test si el código de respuesta es 200.

- **Insertar Clave Pública Credenciales Incorrectas**

  En esta prueba, se consume el servicio de subida de clave pública de un usuario al servidor, con la clave a subir y unas credenciales de usuario incorrectas. Se da por aprobado el test si el código de respuesta es 401.

- **Insertar Mensaje Correcto**

  En esta prueba, se consume el servicio de envío de un nuevo mensaje al servidor dirigido a un usuario existente en la base de datos de este. Se da por correcto el test si el código de la respuesta es 200.

- **Insertar Mensaje Usuario Inexistente**

  En esta prueba, se consume el servicio de envío de un nuevo mensaje al servidor dirigido a un usuario no existente en la base de datos de este. Se da por correcto el test si el código de la respuesta es 404.

- **Crear Grupo Correcto**

  En esta prueba, se consume el servicio de creación de un nuevo grupo en el servidor con las credenciales del usuario creador existente en la base de datos del servidor. La prueba se da por correcta si el código de respuesta es 200.

- **Crear Grupo Credenciales Incorrectas**

  En esta prueba, se consume el servicio de creación de un nuevo grupo en el servidor con las credenciales incorrectas de un usuario del servidor. La prueba se da por correcta si el código de respuesta es 401.

- **Editar Grupo Correcto**

  En esta prueba, se consume el servicio de edición de un grupo ya existente en el servidor mediante su código de grupo y las credenciales del usuario creador. El test se da por aprobado si el código de la respuesta es 200.

- **Editar Grupo Grupo Inexistente**

  En esta prueba, se consume el servicio de edición de un grupo no existente en el servidor mediante su código de grupo y las credenciales de un usuario. El test se da por aprobado si el código de la respuesta es 401.

- **Editar Grupo Credenciales Incorrectas**

  En esta prueba, se consume el servicio de edición de un grupo existente en el servidor mediante su código de grupo y las credenciales incorrectas del usuario creador de este. El test se da por aprobado si el código de la respuesta es 401.

- **Get Grupo Correcto**

  En esta prueba, se consume el servicio de obtención de un grupo existente en el servidor mediante su código de grupo. La prueba se da por correcta si el código de respuesta es 200.

- **Get Grupo Grupo Inexistente**

En esta prueba, se consume el servicio de obtención de un grupo no existente en el servidor mediante un código de grupo. La prueba se da por correcta si el código de respuesta es 404.

- **Eliminar Grupo Correcto**

  En esta prueba, se consume el servicio de eliminación de un grupo existente en el servidor mediante su código de grupo y las credenciales del usuario creador. El test se da por correcto si el código de respuesta es 200.

- **Eliminar Grupo Credenciales Incorrectas**

  En esta prueba, se consume el servicio de eliminación de un grupo existente en el servidor mediante su código de grupo y las credenciales incorrectas del usuario creador. El test se da por correcto si el código de respuesta es 401.

- **Eliminar Grupo Inexistente**

  En esta prueba, se consume el servicio de eliminación de un grupo no existente en el servidor mediante un código de grupo y las credenciales de un usuario. El test se da por correcto si el código de respuesta es 401.

4.1.2 Selenium

Se han definido los siguientes tests:

- **CRUD Usuarios**

  Este test comprueba que se puede realizar sin problema la creación, la lectura, la actualización y el borrado de usuarios en el servidor. Para ello, primero hace un login en el panel de administración, entra al apartado de usuarios, crea uno nuevo, lo edita, y finalmente lo elimina. Si todo ha ido bien, se da por hecho que el CRUD de usuarios está funcionando y se asume que el test está correcto.

- **Configuración**

Este test comprueba que la funcionalidad de cambio de configuración funciona correctamente. Para ello, Selenium inicia sesión en el panel de administración del servidor, va al apartado de configuración, modifica algunos parámetros, guarda la configuración, vuelve a modificarlo para dejarlo como estaba y vuelve a guardar. Si el proceso ha ido bien se da la prueba como correcta.

- **Leer usuarios**

  En esta prueba se comprueba que el listado de usuarios del panel de administración cargue correctamente. Simplemente, hace un login al panel de administración y entra en el apartado de usuarios.

- **Login**

  Este test comprueba que el login del panel de administración funciona adecuadamente. Para ello, hace un login y a continuación cierra sesión. Si durante la ejecución no ha podido pulsar el enlace de cerrar sesión, significaría que ni ha podido hacer login. Si ha podido cerrar sesión, se entiende que pudo hacer login primero, por lo tanto, el resultado del test sería correcto.

- **Notificar a todos**

  Esta test prueba que funcione la característica de envío de una notificación a todos los usuarios del servidor mediante correo electrónico. Para ello inicia sesión en el panel de administración, va al apartado de "Notificar a los usuarios", redacta una notificación de prueba y la envía. Queda por parte de la persona que lance la prueba comprobar la bandeja de entrada de uno de los correos de los usuarios registrados en el servidor. Si lo ha recibido damos por aprobado el test.

- **Página de inicio**

Este test únicamente comprueba que funcione la navegabilidad de la página principal, sin entrar al panel de administración.

4.2 Aplicaciones cliente

En las aplicaciones cliente también se han definido dos tipos de pruebas diferentes: un test de JUnit y otros de interfaz de usuario.

4.2.1 JUnit [25]

Se ha definido un test de JUnit para probar la funcionalidad de los algoritmos de cifrado y descifrado de mensajes. El test genera dos pares de claves pública/privada diferentes (simulando que pertenecen a dos personas), crea un objeto mensaje con un contenido tipo String dentro, lo cifra y descifra usando las claves necesarias (de la misma forma que se haría en realidad) y al final se compara la cadena del contenido del mensaje descifrado con el contenido de la cadena inicial.

4.2.2 Pruebas de interfaz de usuario

Tanto para la aplicación cliente de Android como para la de escritorio, se han definido los mismos tests, pero la implementación ha tenido que hacerse usando diferentes tecnologías para cada uno: Espresso [26] para la aplicación Android y Marathon para la de escritorio.

Estos son los tests que se han definido:

- **CRUD Contactos**

  En este test se prueba la funcionalidad de crear, listar, actualizar y eliminar contactos en la aplicación. Para ello se hace un inicio de sesión en la aplicación, se agrega un contacto, se edita, y por último se elimina.

- **CRUD Difusiones**

En este test se prueba la funcionalidad de crear, listar, actualizar y eliminar listas de difusión en la aplicación. Para ello se hace un inicio de sesión en la aplicación, se agregan dos contactos, se crea una lista de difusión con estos contactos creados al principio del test, se edita, y por último se elimina la lista de difusión y los dos contactos.

- **CRUD Grupos**

  En este test se prueba la funcionalidad de crear, listar, actualizar y eliminar grupos en la aplicación. Para ello se hace un inicio de sesión en la aplicación, se agregan dos de contactos, se crea un grupo con estos contactos creados al principio del test, se edita, y por último se elimina el grupo y los dos contactos.

- **Enviar mensaje difundido**

  En este test se comprueba que se puede enviar un mensaje difundido. Primeramente se hace un inicio de sesión, se agregan dos contactos, se crea una lista de difusión y se añaden a estos dos contactos, se crea un chat para esta difusión y se escribe un mensaje. Finalmente se elimina la lista y los dos contactos.

- **Enviar mensaje grupal**

  En este test se comprueba que se puede enviar un mensaje grupal. Para ello, primeramente se hace un inicio de sesión, se crean dos contactos, se crea un grupo y se agrega como miembros a estos dos contactos, se crea un chat para este grupo y se escribe un mensaje. Finalmente se elimina el grupo y los dos contactos.

- **Enviar mensaje privado**

En este test se comprueba que se puede enviar un mensaje privado. Primeramente se hace un inicio de sesión, se agrega un contacto y se le escribe un mensaje. Finalmente se elimina el contacto.

- **Login**

  Este test comprueba que funciona el login correctamente. Para ello hace un inicio de sesión y entra en la vista de contactos. Si no pudiese entrar en la vista de contactos, es que el login a fallado.

5

Conclusiones y Trabajo Futuro

En este capítulo expondremos las conclusiones a las que hemos llegado al finalizar este trabajo de fin de grado. En concreto, enumeraremos las conclusiones y objetivos alcanzados y las funcionalidades o posibles mejoras que podrían añadirse en el futuro.

5.1 Conclusiones

El objetivo de este trabajo de fin de grado era crear un sistema de mensajería instantánea descentralizado y seguro, que brinde la oportunidad a cualquier usuario de estudiar el código fuente y de convertirse en un proveedor más del servicio, y de proporcionar cifrado de extremo a extremo usando criptografía asimétrica. El objetivo ha sido cumplido tras finalizar cada uno de los cincos sprints en los que se ha dividido el proyecto.

De entre las primeras cosas que se realizaron en el proyecto, podemos destacar la elección de tecnologías a usar, que tuviesen como base el lenguaje Java, que fuesen

relativamente modernas, sigan mantenidas y que sean compatibles con las plataformas más usadas por los usuarios del mundo de la informática. También se hizo un análisis y enumeración de historias de usuario que deberían cubrir la funcionalidad del sistema, así como el diseño del diagrama entidad relación de la aplicación servidor y de las aplicaciones cliente, aunque estos últimos fueron modificándose levemente a lo largo de cada sprint.

Se realizó correctamente la implementación de la aplicación servidor y las dos aplicaciones cliente (escritorio y móvil) y se diseñó una batería de pruebas para comprobar que todo el funcionamiento de las tres aplicaciones era correcto.

Otro objetivo importante, que me animó a la realización de este trabajo de fin de grado era el aprendizaje en sí y, personalmente, pienso que se ha logrado. Siento que he aprendido muchísimo acerca de la programación de aplicaciones de escritorio con JavaFX y de aplicaciones móviles Android, y reforzado mis conocimientos de JavaEE para aplicaciones del lado del servidor.

Algunos problemas que he tenido durante el desarrollo de este trabajo han sido la escasa documentación de JavaFX y la gran cantidad de información y documentación ya desactualizada de Android. No obstante, logré encontrar todo lo que necesitaba tanto en las documentaciones oficiales, blogs y video-tutoriales en Youtube.

5.2 Trabajo futuro

En este trabajo de fin de grado se ha realizado la implementación de las aplicaciones cliente para dispositivos de escritorio y para dispositivos Android. Un posible trabajo futuro sería la implementación de otra aplicación cliente para dispositivos iOS [27], y así dejar prácticamente cubierto todo el ecosistema de los dispositivos móviles. También sería interesante valorar la posibilidad de crear otra

implementación de la aplicación cliente para navegadores web con algún Framework de Javascript/Typescript, como por ejemplo Angular [28], el cuál permitirá no requerir hacer actualizaciones a los usuarios, eliminar problemas de incompatibilidad entre diferentes versiones, no instalar nada en los dispositivos de los usuarios y garantizar el funcionamiento en cualquier sistema operativo.

También se podría cambiar el modo de autentificación por uno basado en tokens [29] en vez de incluir el par usuario-contraseña en cada petición, tanto en los métodos del servicio REST como en el websocket. Esto supondría una mejora de la seguridad, ya que la contraseña del usuario no se quedaría almacenada en ningún lugar de la memoria del dispositivo cliente. En lugar de almacenar en memoria la contraseña, se almacenaría el token, que tendría una caducidad y se podría revocar en caso de que el servidor lo considere oportuno, pidiendo las credenciales de nuevo al usuario para demostrar que es él de verdad.

Un último trabajo o mejora futura, podría ser implementar una característica en las aplicaciones cliente que avise que la conexión no es segura cuando se está conectando a un servidor con certificado TLS autofirmado, y que se pida la confirmación al usuario de continuar con la conexión aceptando el riesgo de seguridad o de abortarla. De esta forma se podrían evitar posibles intentos de suplantación de un servidor con la intención de obtener las credenciales del usuario por parte de un tercero.

Referencias

[1] Wikipedia. Whatsapp, 2021

Sitio web: https://es.wikipedia.org/wiki/WhatsApp

[2] Wikipedia. Telegram, 2021

Sitio web: https://es.wikipedia.org/wiki/Telegram

[3] Sistema Operativo GNU, Software libre, 2021

Sitio web: https://www.gnu.org/philosophy/free-sw.es.html

[4] Wikipedia, Cifrado extremo a extremo, 2021

Sitio web: https://es.wikipedia.org/wiki/Cifrado_de_extremo_a_extremo

[5] Wikipedia, Scrum, 2021

Sitio web: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[6] Wikipedia, Android Studio, 2021

Sitio web: https://es.wikipedia.org/wiki/Android_Studio

[7] Wikipedia, Android SDK, 2021

Sitio web: https://es.wikipedia.org/wiki/Android_SDK

[8] Diagrams, App Diagrams, 2021

Sitio web: https://app.diagrams.net/

[9] Wikipedia, Docker, 2021

Sitio web: https://es.wikipedia.org/wiki/Docker_(software)

[10] Eclipse, Eclipse JEE, 2020

Sitio web: https://www.eclipse.org/downloads/packages/release/2020-12/r/eclipse-ide-enterprise-java-developers

[11] Wikipedia, Java EE, 2021

Sitio web: https://es.wikipedia.org/wiki/Java_EE

[12] Openjfx, Java FX, 2021

Sitio web: https://openjfx.io/

[13] Wikipedia, Java SE, 2021

Sitio web: https://es.wikipedia.org/wiki/Java_SE

[14] Wikipedia, JMeter, 2021

Sitio web: https://es.wikipedia.org/wiki/JMeter

[15] Marathon Testing, Marathon, 2021

Sitio web: https://marathontesting.com/marathon/

[16] Wikipedia, MySQL, 2021

Sitio web: https://es.wikipedia.org/wiki/MySQL

[17] Gluon, Scene Builder, 2021

Sitio web: https://gluonhq.com/products/scene-builder/

[18] Wikipedia, Selenium, 2021

Sitio web: https://es.wikipedia.org/wiki/Selenium

[19] Wikipedia, SQLite, 2021

Sitio web: https://es.wikipedia.org/wiki/SQLite

[20] Wikipedia, Wildfly, 2021

Sitio web: https://es.wikipedia.org/wiki/WildFly

[21] Wikipedia, Historias de usuario, 2021

Sitio web: https://es.wikipedia.org/wiki/Historias_de_usuario

[22] Wikipedia, Modelo Entidad-Relación, 2021

Sitio web: https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

[23] Wikipedia, Diagrama de secuencia, 2021

Sitio web: https://es.wikipedia.org/wiki/Diagrama_de_secuencia

[24] Universitat Oberta de Catalunya, Diagramas de flujo de navegación, 2021

Sitio web: http://cv.uoc.edu/UOC/a/moduls/90/90_574b/web/main/m5/c4/3.html

[25] Wikipedia, JUnit, 2021

Sitio web: https://es.wikipedia.org/wiki/JUnit

[26] Android Developers, Espresso, 2021

Sitio web: https://developer.android.com/training/testing/espresso

[27] Wikipedia, iOS, 2021

Sitio web: https://es.wikipedia.org/wiki/IOS

[28] Wikipedia, Angular, 2021

Sitio web: https://es.wikipedia.org/wiki/Angular_(framework)

[29] Arsyrs, Autentificación basada en Token, 2021

Sitio web: https://www.arsys.es/blog/programacion/autenticacion-api-rest-token/

Apéndices

Apéndice A

Manual del servidor

A.1 Instalación

Antes de comenzar, debemos asegurarnos de tener Docker y Docker Compose instalados en la máquina donde queramos desplegar la aplicación servidor. Si no lo tuviésemos, instalaríamos estas dos herramientas usando las siguientes guías:

- Docker: https://docs.docker.com/engine/install/
- Docker Compose: https://docs.docker.com/compose/install/

Una vez tengamos estas dos herramientas listas, descomprimiremos el fichero comprimido "smids-server.zip", entraremos en el directorio smids-server y abrimos con un editor de texto el archivo "docker-compose.yml".

Si queremos que la aplicación se inicie al arrancar el sistema descomentaremos las líneas "restart: always":

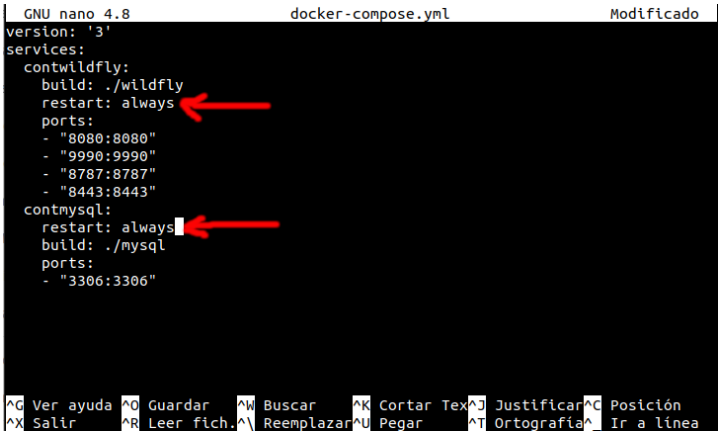

Figura 31: Fichero docker-compose.yml del servidor

Dentro de ese mismo directorio ejecutaremos el siguiente comando en una terminal:

$ > docker-compose up -d

Adicionalmente, podríamos quitar el parámetro "-d" si necesitamos ver el log de la aplicación en primer plano.

Ahora abriremos un navegador web y entraremos en la URL *http://localhost:8080*

Una vez allí pulsaremos en el enlace del menú superior "Administración":

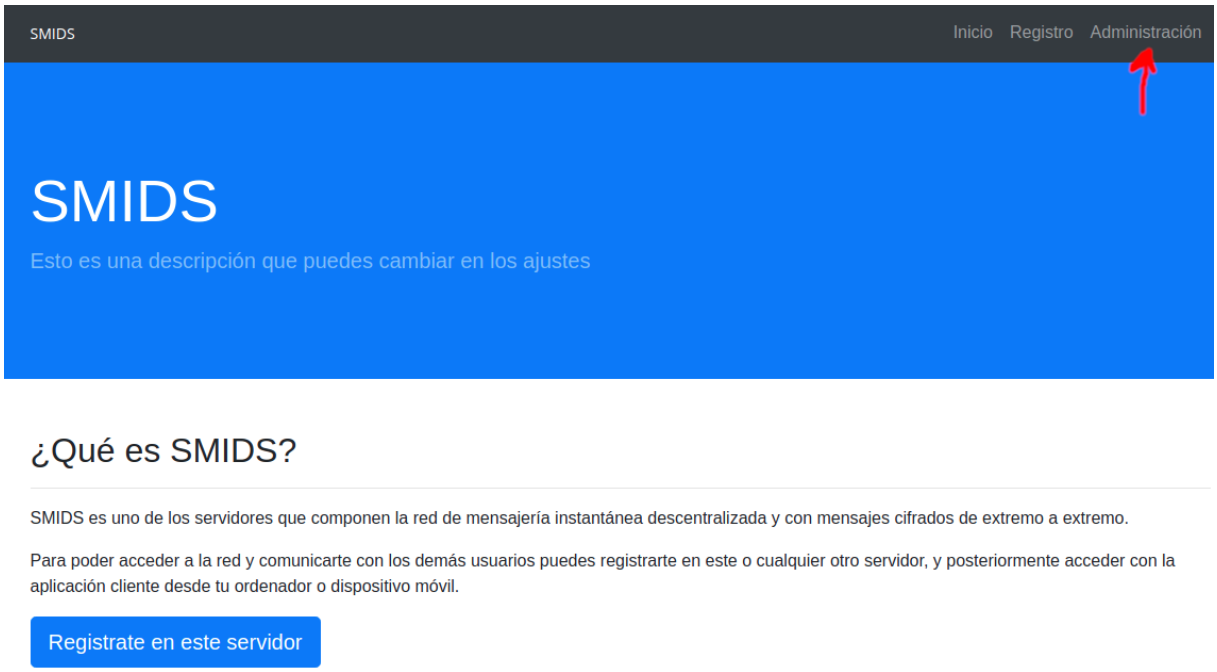

Figura 32: Página principal del servidor con enlace a administración resaltado

Ingresaremos al panel de administración usando el usuario administrativo que se crea por defecto:

Usuario: **admin**

Contraseña: **cambiame**

Una vez dentro, nos iremos a la parte superior derecha, abrimos el desplegable y pulsamos en “Editar perfil”:

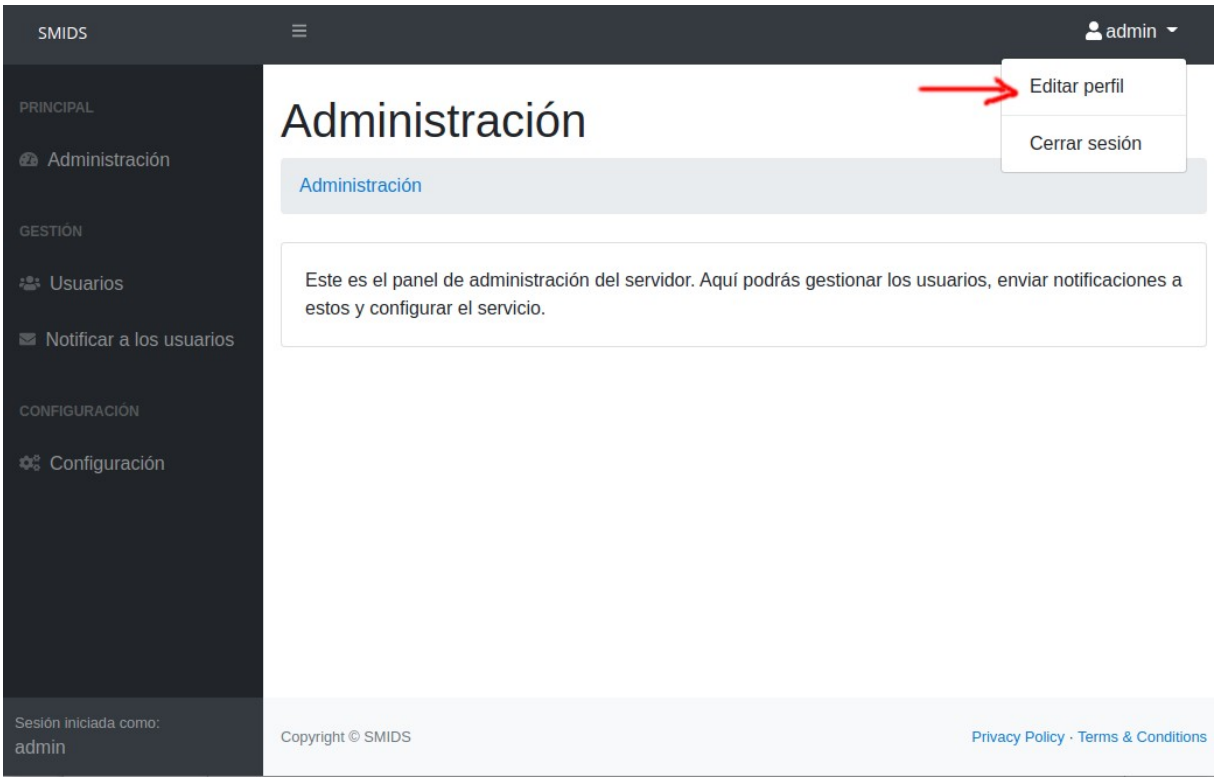

Figura 33: Panel de administración del servidor con enlace a Editar Perfil resaltado

Ponemos una nueva contraseña y un correo electrónico, y después pulsamos en "Aceptar":

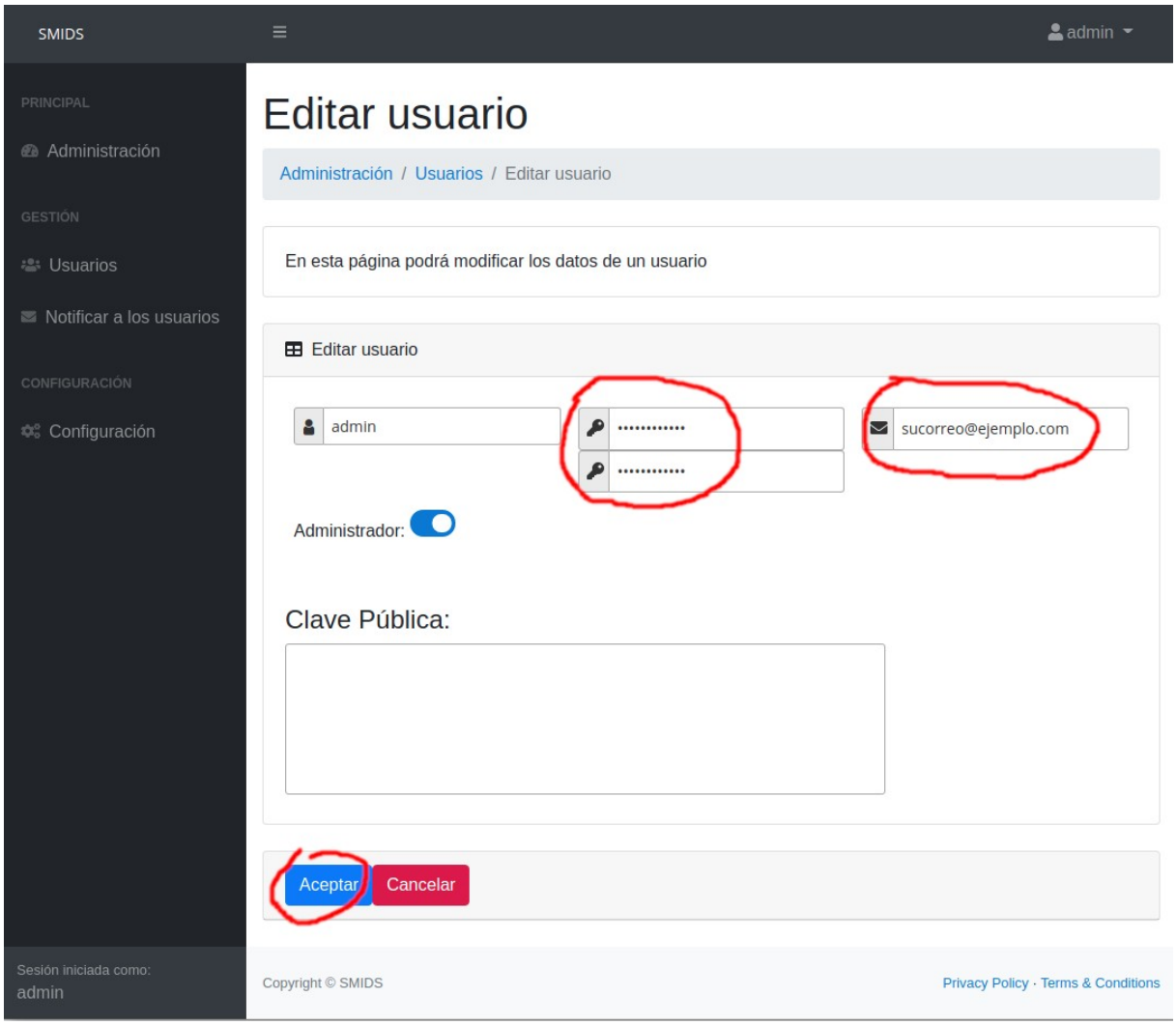

Figura 34: Página de editar perfil

Por último, nos vamos al apartado de configuración (accesible mediante el menú de navegación de la izquierda) y editamos estos ajustes a nuestro gusto. Para que estén disponibles las funcionalidades de registro por parte de los usuarios y de notificaciones a todos los miembros, será necesario tener configurada una cuenta de correo SMTP y habilitar el envío de emails.

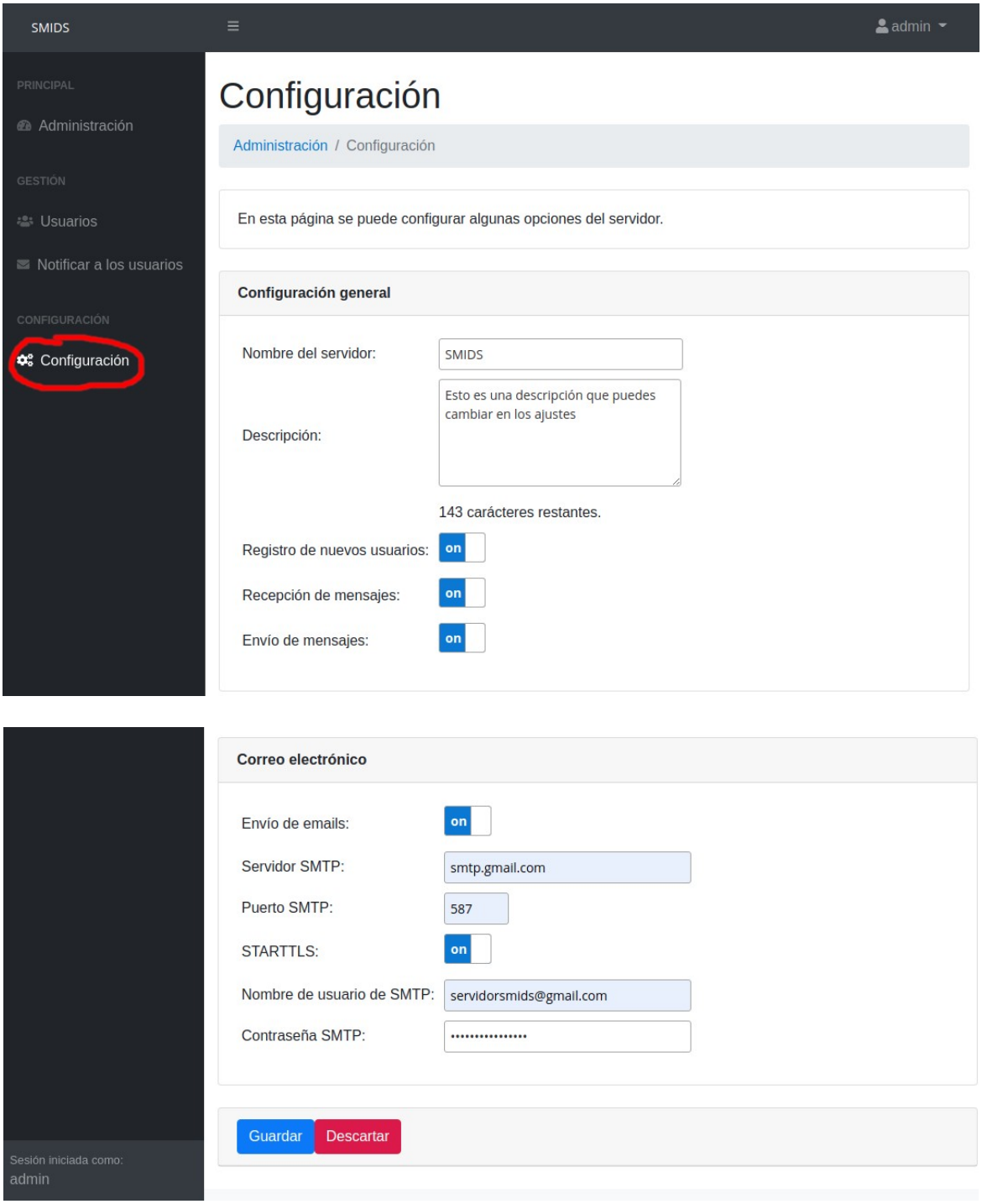

Figura 35: Página de configuración del servidor

A.2 Guía de uso

Desde la página principal del servidor, los usuarios podrán registrarse en el sistema:

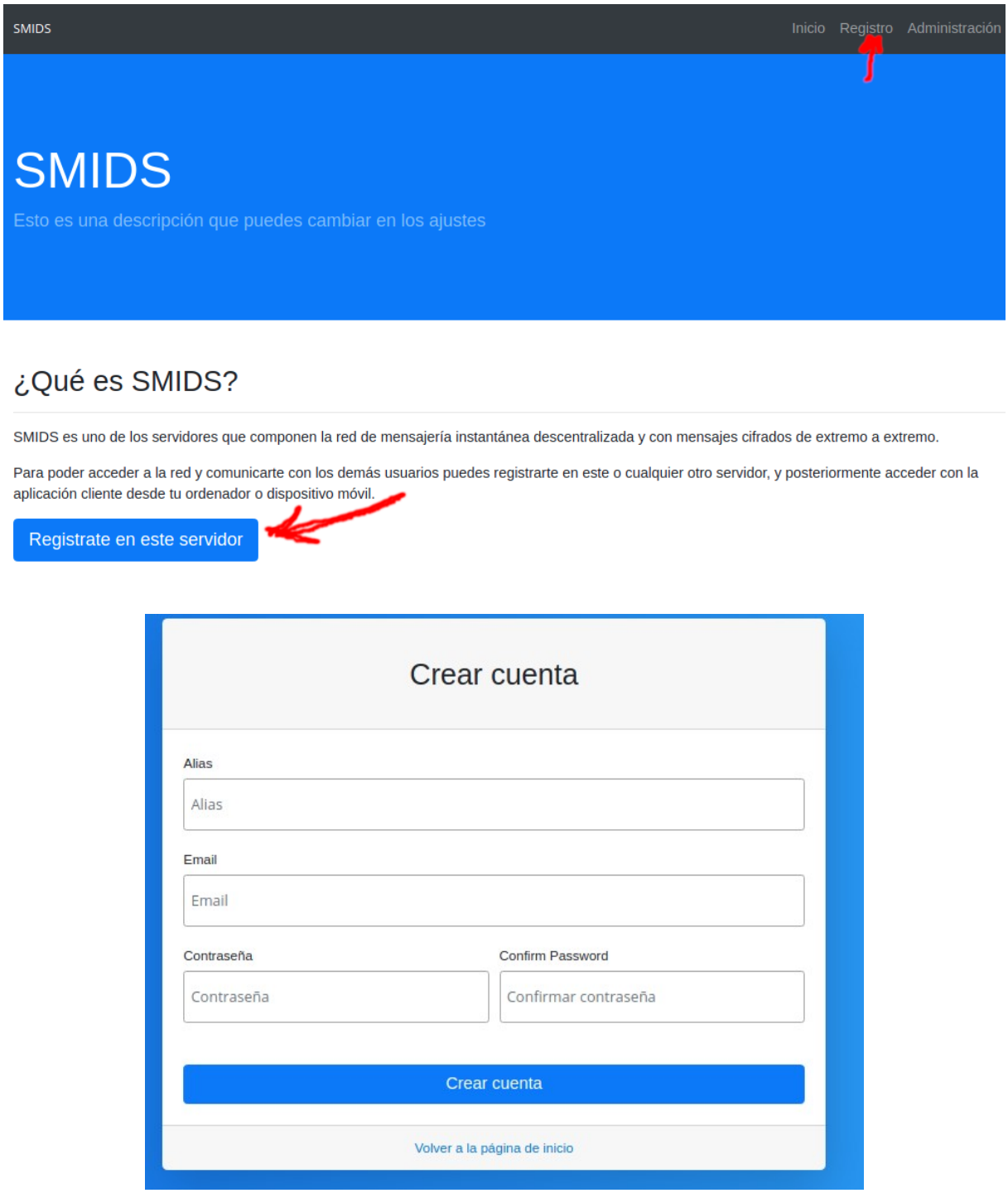

Figura 36: Registro de usuario en el servidor

También podrán descargar las aplicaciones cliente:

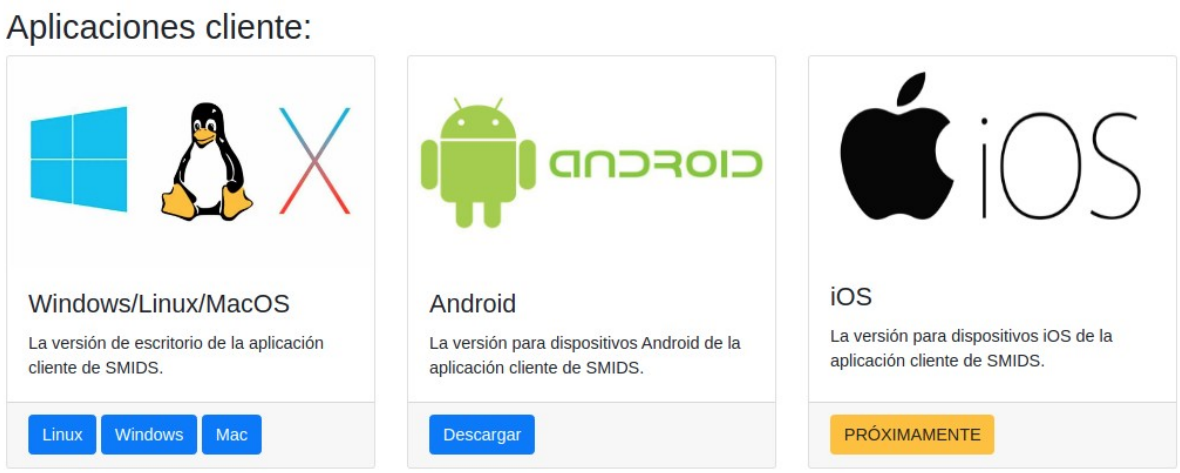

Figura 37: Apartado de descarga de las aplicaciones cliente del servidor

Al panel de administración (solo accesible a usuarios con rol de administrador) se podrá acceder mediante el enlace "Administración" del menú superior:

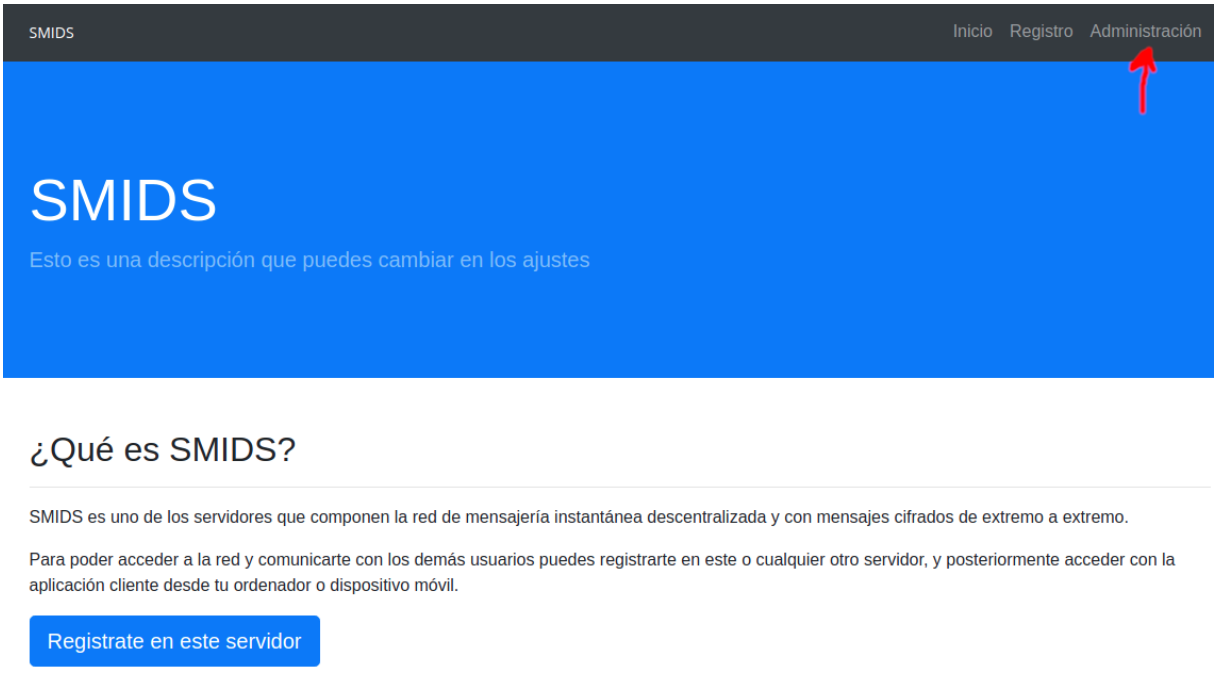

Figura 38: Página de inicio del servidor con el enlace de Administración resaltado

Accedemos con el usuario y la contraseña de administrador.

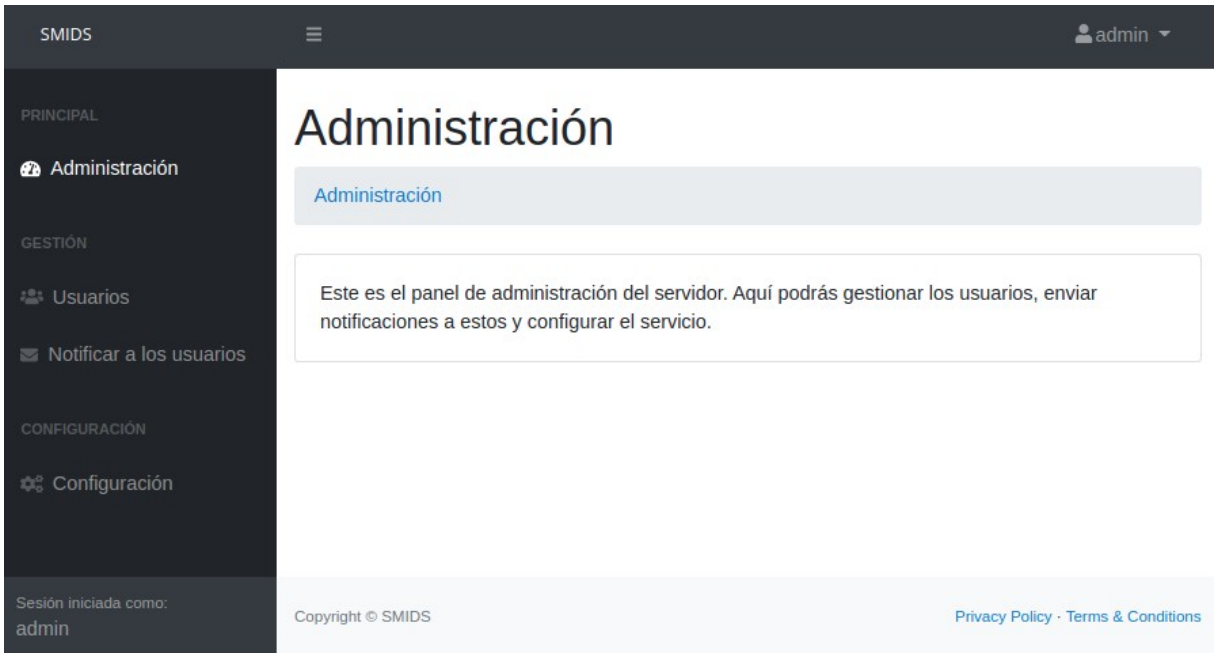

Figura 39: Panel de administración del servidor

Pulsando en el enlace "Usuarios" del menú de navegación de la izquierda, podremos gestionar los usuarios del servidor (crear, editar, eliminar, validar, etc.):

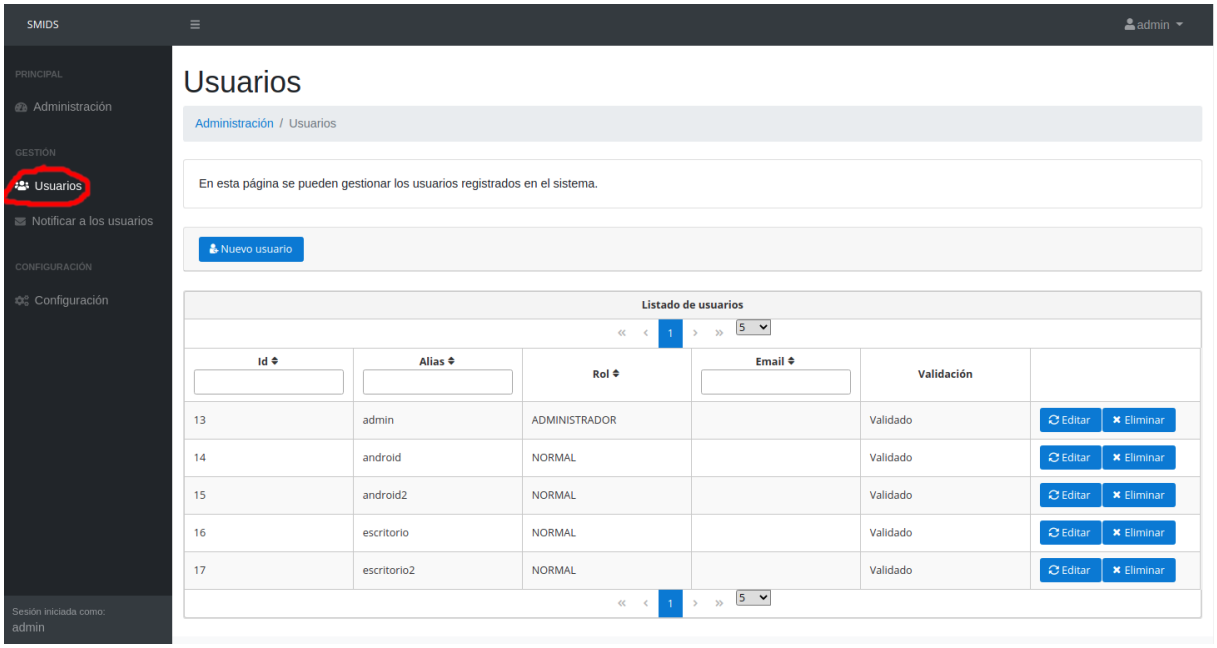

Figura 40: Página de gestión de usuarios del servidor

En el apartado “Notificar a los usuarios”, también de este mismo menú, podremos enviar un correo electrónico, a modo de comunicado, a todos los usuarios del servidor, con el fin de mantenerles informados del estado del servicio, posibles bajas de este por mantenimiento, etc.

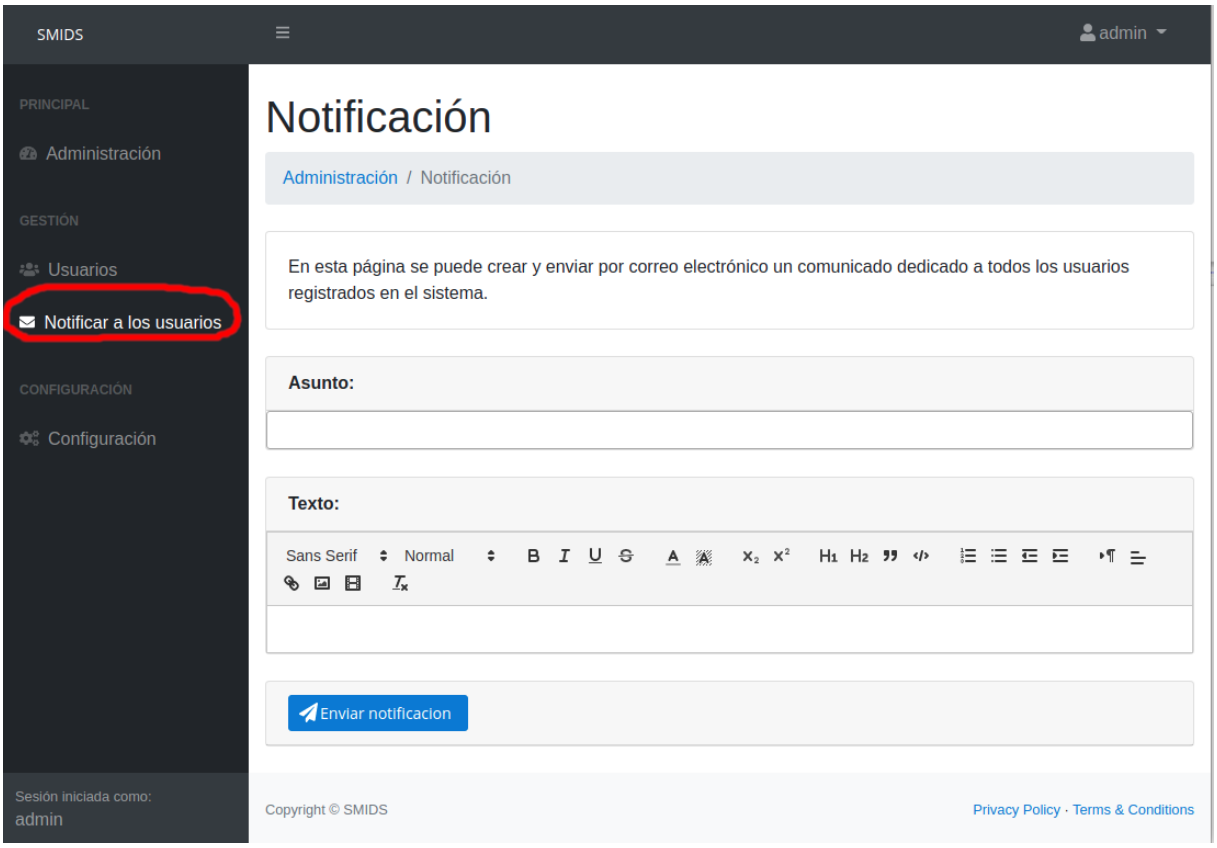

Figura 41: Página de notificar a los usuarios del servidor

En el apartado de “Configuración” podremos cambiar el nombre del servidor y la descripción, los cuáles saldrán en la página inicial, activar o desactivar el registro de nuevos usuarios, recepción y envíos de nuevos mensajes, y configurar la cuenta de correo SMTP que la aplicación usará para mandar correos electrónicos:

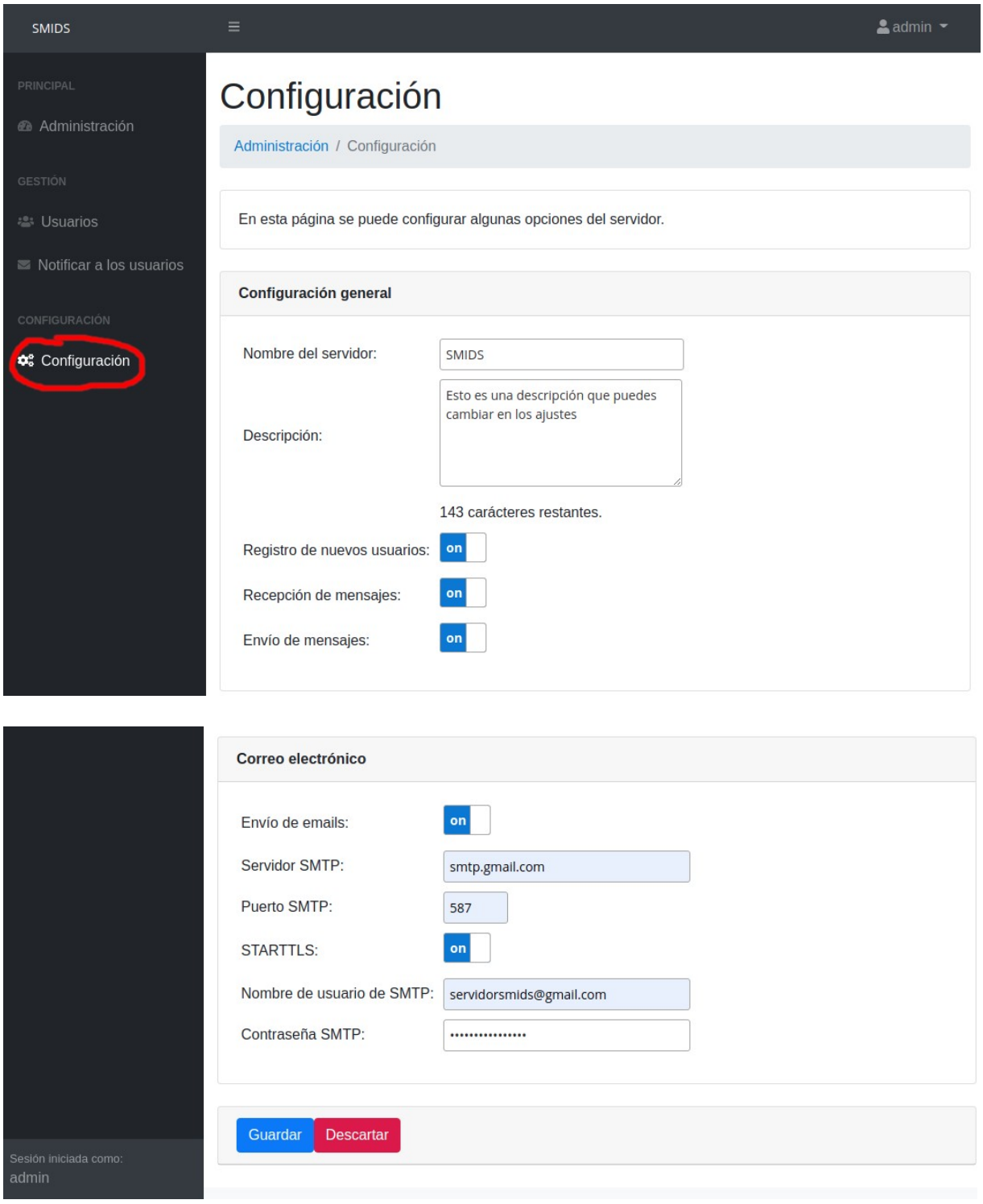

Figura 42: Página de configuración del servidor

Apéndice B

Manual de la Aplicación Android

B.1 Instalación

Simplemente abrimos el fichero smids-cliente-android.apk y pulsamos en instalar. Una vez completada la instalación se recomienda desactivar el ahorro de energía para esta aplicación en los ajustes de Android.

B.2 Inicio de sesión

Al abrir la aplicación podremos elegir entre acceder iniciando sesión o acceder sin conexión. El modo sin conexión nos permite acceder a la aplicación sin necesidad de autentificarnos y de tener conexión a internet, para consultar los mensajes, información de grupos, difusiones... pero en este modo no podremos enviar ni recibir nada. En cambio, iniciando sesión, sí tendremos toda la funcionalidad, pero es necesario tener conexión a internet y haberse registrado en un servidor previamente.

Figura 43: Vista de inicio de la aplicación Android

Al pulsar sobre iniciar sesión, el programa nos pedirá las credenciales para poder autentificarse en el servidor:

Figura 44: Inicio de sesión en la aplicación Android

B.3 Chats

Al iniciar sesión, lo primero que veremos será la lista de todos los chats que tengamos, y una serie de botones en la barra superior:

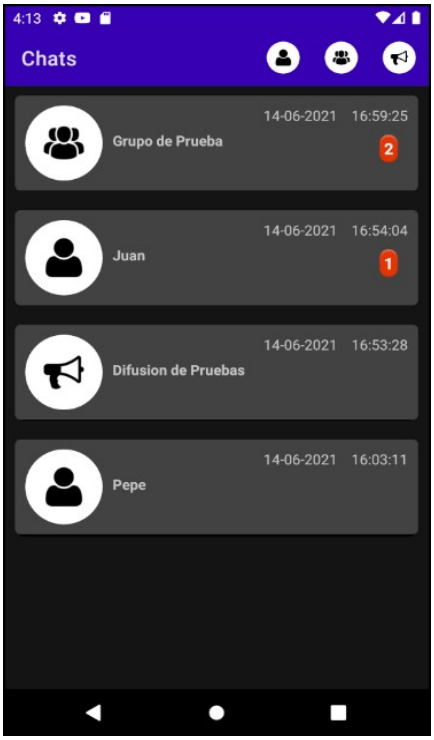

Figura 45: Vista de chats de la aplicación Android

Pulsando sobre cualquier chat, podremos acceder a él y leer y escribir mensajes:

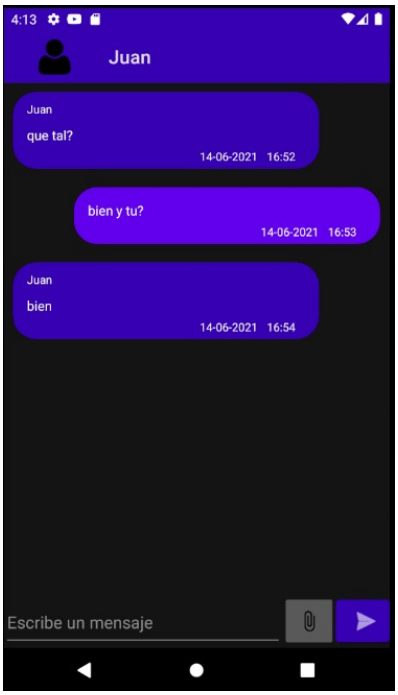

Figura 46: Chat privado Android

Figura 47: Chat grupal Android

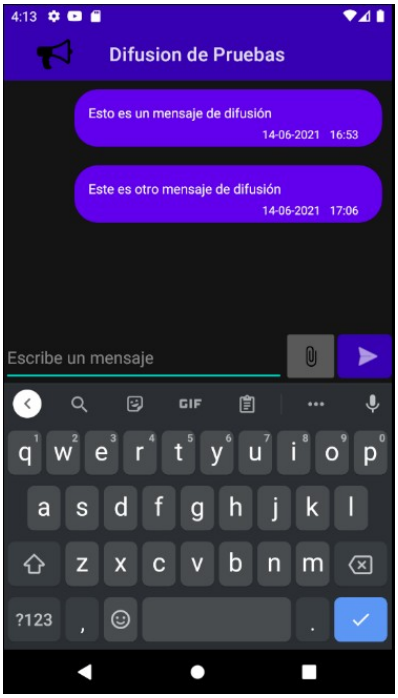

Figura 48: Chat de difusión Android

B.4 Contactos

Para gestionar los contactos pulsaremos el botón con el icono de un contacto situado en la barra superior de la vista de chats:

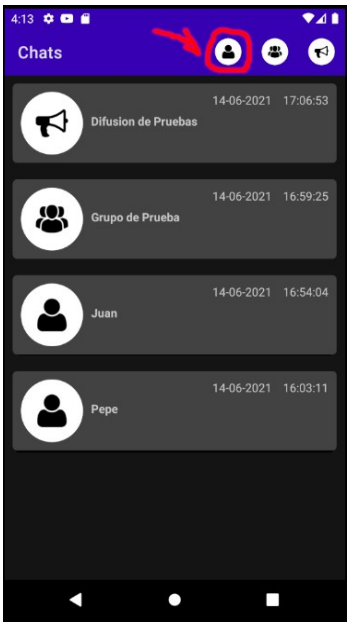

Figura 49: Vista de chats de la aplicación Android con botón de contactos resaltado

A continuación, se nos mostrará un listado con todos los contactos que tengamos registrados en el programa:

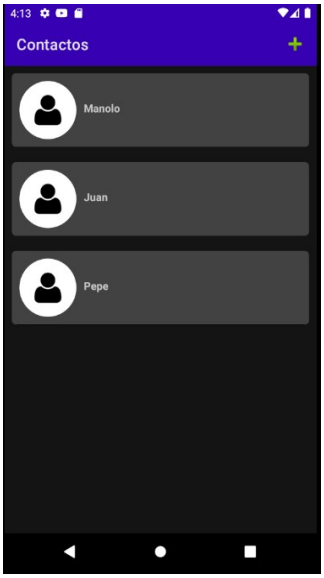

Figura 50: Vista de contactos de la aplicación Android

Podremos agregar nuevos contactos pulsando en el icono "+" situado a la derecha de la barra superior y, posteriormente, introduciendo un nombre de contacto y la dirección de este:

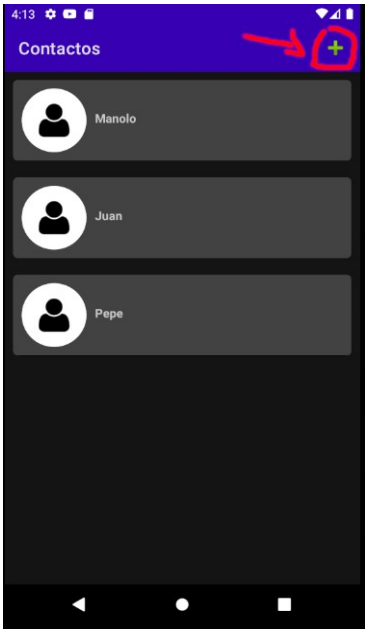

Figura 51: Botón de añadir contacto en la app Android resaltado

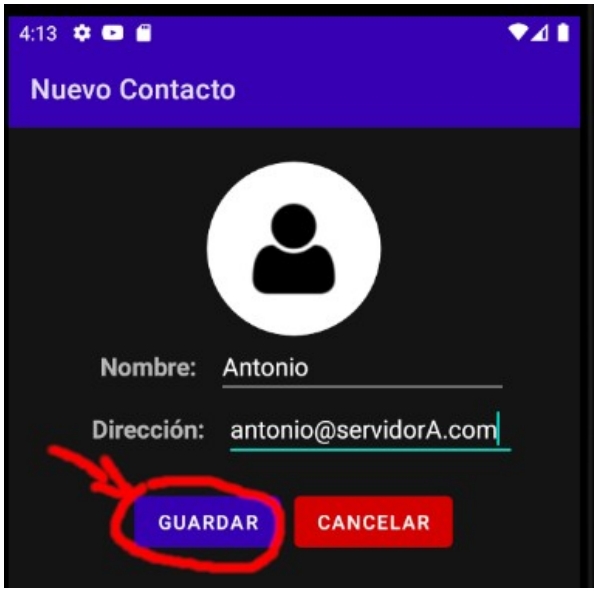

Figura 52: Vista de añadir contacto de la app Android

En el listado de contactos, si pulsamos en cualquiera de ellos, se nos abrirá una vista con los detalles de este. Desde esta vista también podremos escribirle (se crearía un chat en caso de que no existiera previamente), eliminar toda la conversación, y editar o eliminar el contacto usando los dos botones de la barra superior:

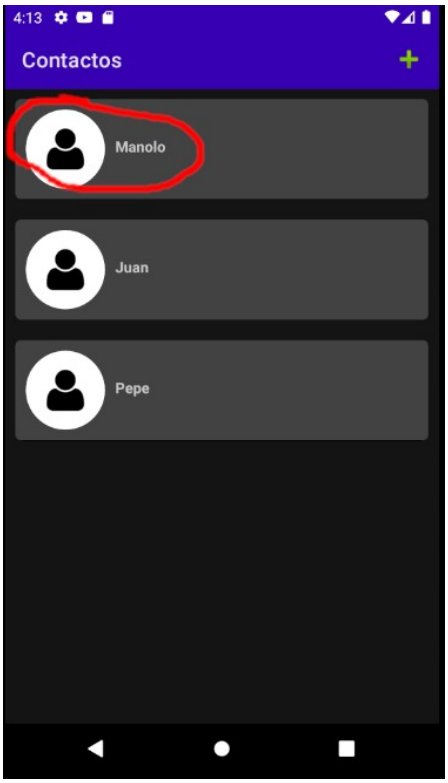

Figura 53: Resalto de un contacto del listado de contactos de la app Android

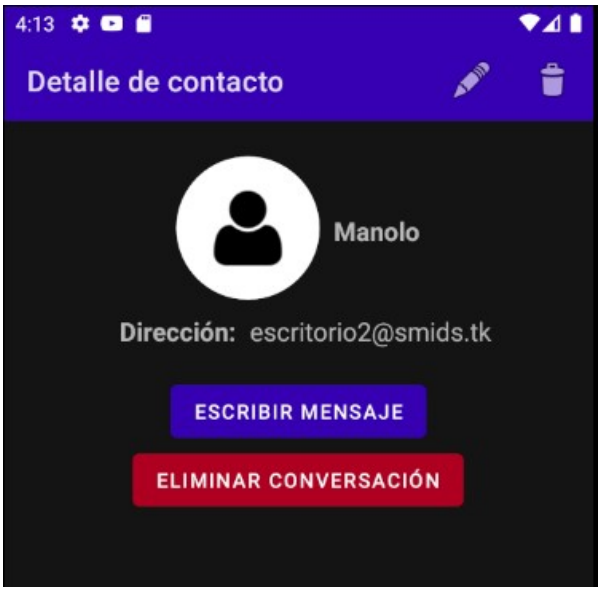

Figura 54: Vista de detalle de contacto de la app Android

B.5 Grupos

Para gestionar los grupos, nos iremos a la vista de chats y pulsaremos sobre el botón con el icono de grupo, situado en la barra superior de la vista:

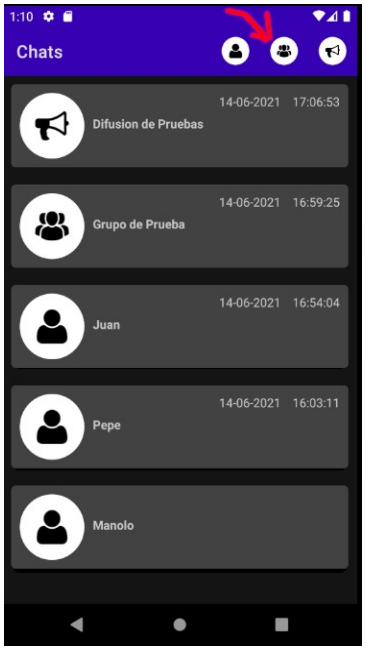

Figura 55: Vista de chats de la app Android con el botón de grupos resaltado

A continuación, se nos mostrará un listado con todos los grupos a los que pertenecemos:

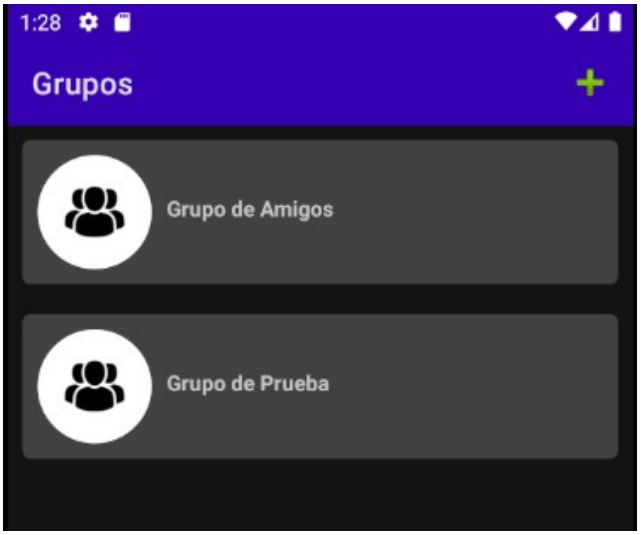

Figura 56: Vista de grupos de la app Android

Podremos crear nuevos grupos pulsando en el icono "+" situado a la derecha de la barra superior y, posteriormente, introduciendo un nombre de grupo y todos aquellos contactos que queramos que sean miembros:

Figura 57: Vista de grupos de la app Android con el botón de añadir grupo resaltado

Figura 58: Vista de nuevo grupo de la app Android

En el listado de grupos, si pulsamos en cualquiera de ellos, se nos abrirá una vista con los detalles de este. Desde esta vista de detalle también podremos escribir en el grupo (se crearía un chat en caso de que no existiera previamente), eliminar toda la conversación, y editar o eliminar el grupo (siempre y cuando el usuario sea el creador) usando los dos botones de la barra superior. En caso que el usuario no sea el creador del grupo, los botones de editar y eliminar no aparecerán en la vista de detalle de grupo.

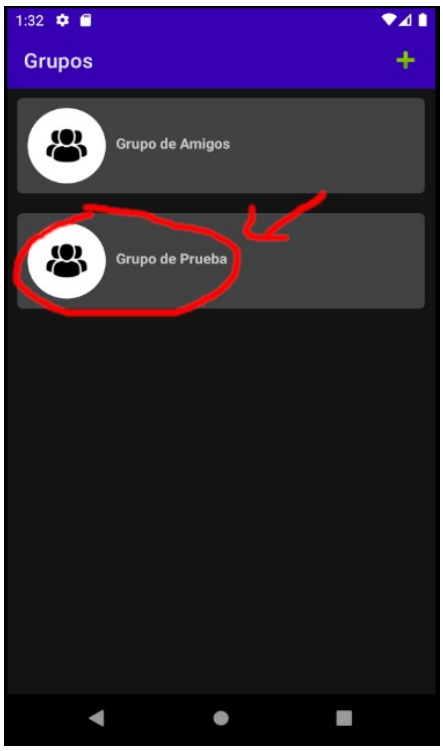

Figura 59: Resalto de un grupo del listado de grupos de la app Android

Figura 60: Vista detalle de grupo de la app Android

B.6 Listas de Difusión

Para gestionar las listas de difusión, nos iremos a la vista de chats y pulsaremos sobre el botón con el icono de un megáfono, situado en la barra superior de la vista:

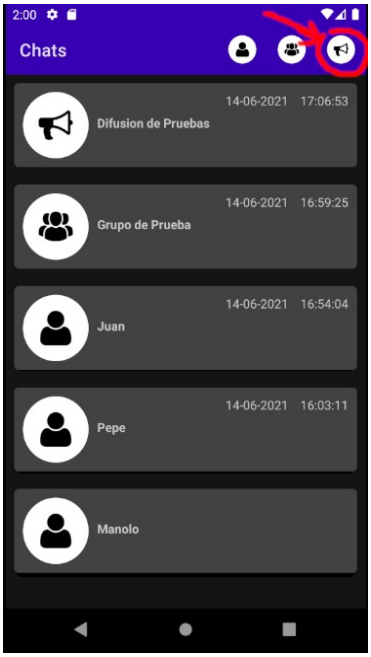

Figura 61: Vista de chats de la app Android con el botón de grupos resaltado

A continuación, se nos mostrará un listado con todas las listas de difusión del usuario:

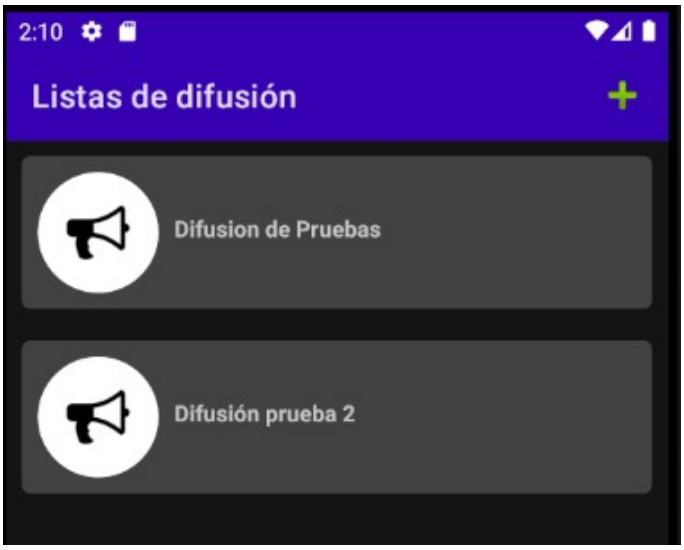

Figura 62: Vista de listas de difusión de la app Android

Podemos crear nuevas listas de difusión pulsando en el icono "+" situado a la derecha de la barra superior y, posteriormente, introduciendo un nombre de la lista y todos aquellos contactos que queramos que sean miembros:

Figura 63: Vista de listas de difusión de la app Android con el botón de añadir resaltado

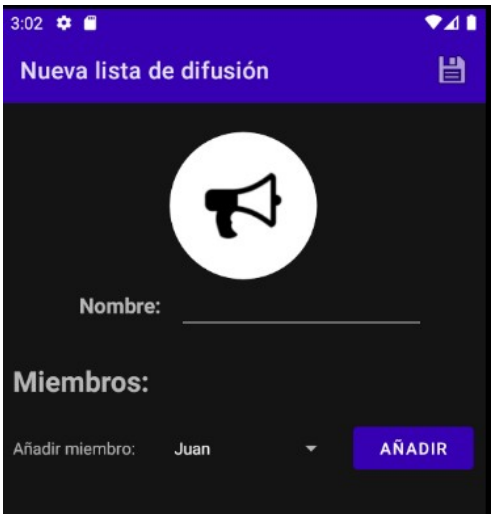

Figura 64: Vista de nueva lista de difusión de la app Android

En la vista de listas de distribución, si pulsamos en cualquiera de ellas, se nos abrirá otra vista con los detalles de la lista. Desde esta vista de detalle también podremos escribir en esa lista (se crearía un chat en caso de que no existiera previamente), eliminar toda la conversación, y editar o eliminarla usando los dos botones de la barra superior.

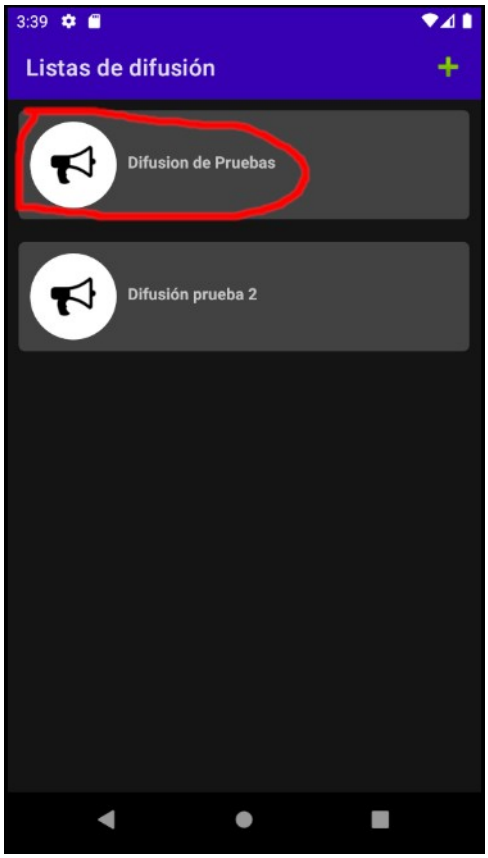

Figura 65: Resalto de una lista de difusión en la vista de difusiones de la app Android

Figura 66: Vista detalle de lista de difusión de la app Android

Apéndice C

Manual de la Aplicación de Escritorio

Esta aplicación no dispone de ningún instalador. Simplemente, descomprimimos el fichero **smids-cliente-escritorio.zip** y ejecutamos el script **smids.sh** o **smids.bat** dependiendo del sistema operativo en el que nos encontremos.

C.1 Inicio de sesión

Al abrir la aplicación podremos elegir entre acceder iniciando sesión o acceder sin conexión. El modo sin conexión nos permite acceder a la aplicación sin necesidad de autentificarnos y de tener conexión a internet, para consultar los mensajes, información de grupos, difusiones... pero en este modo no podremos enviar ni recibir nada. En cambio, iniciando sesión, sí tendremos toda la funcionalidad, pero es necesario tener conexión a internet y haberse registrado en un servidor previamente.

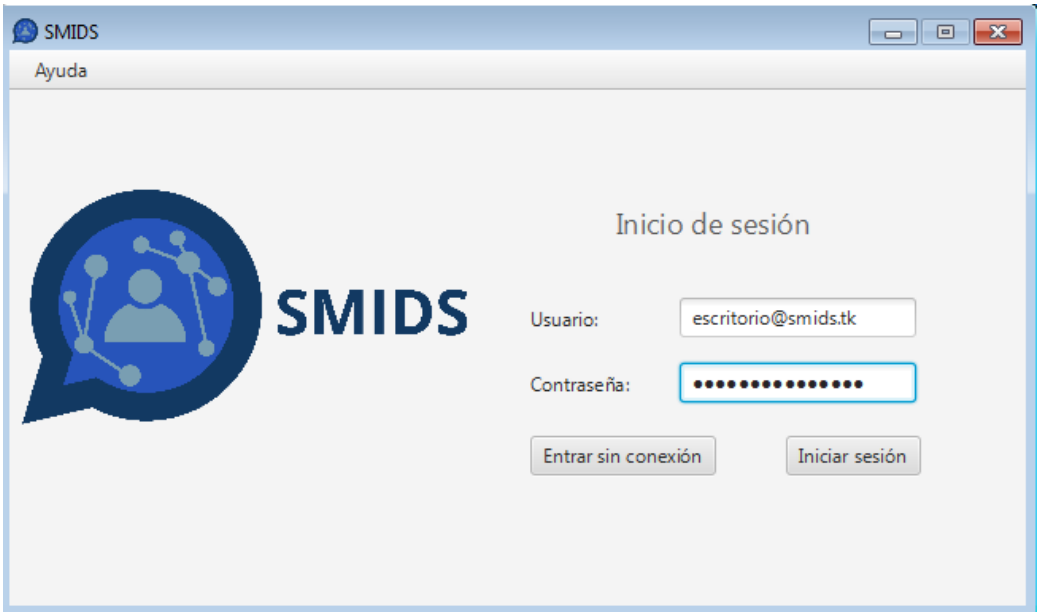

Figura 67: Vista de inicio de sesión de la app de escritorio

C.2 Chats

Al iniciar sesión, veremos en la parte izquierda la lista de todos los chats que tengamos, y una barra de menú en la parte superior.

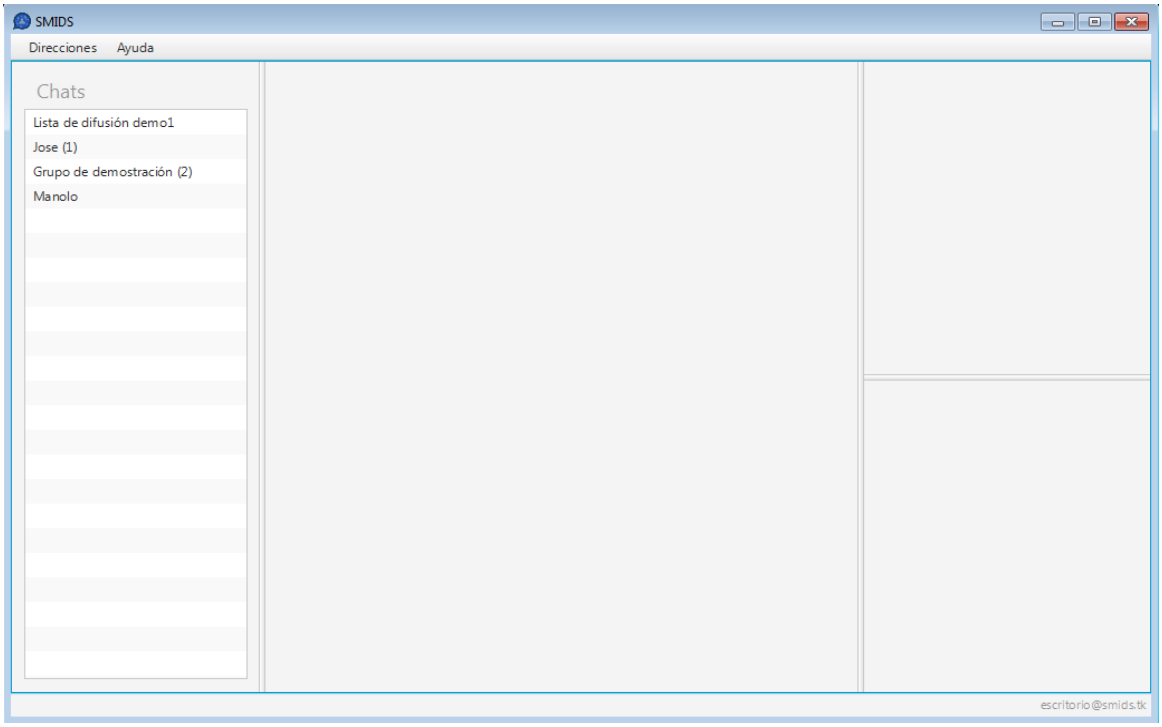

Figura 68: Vista de chats de la app de escritorio

Pulsando sobre cualquier chat, podremos acceder a él, leer y escribir mensajes, ver un detalle del contacto, grupo o lista de difusión asociada a ese chat y eliminar la conversación pulsando en el botón “Eliminar chat”:

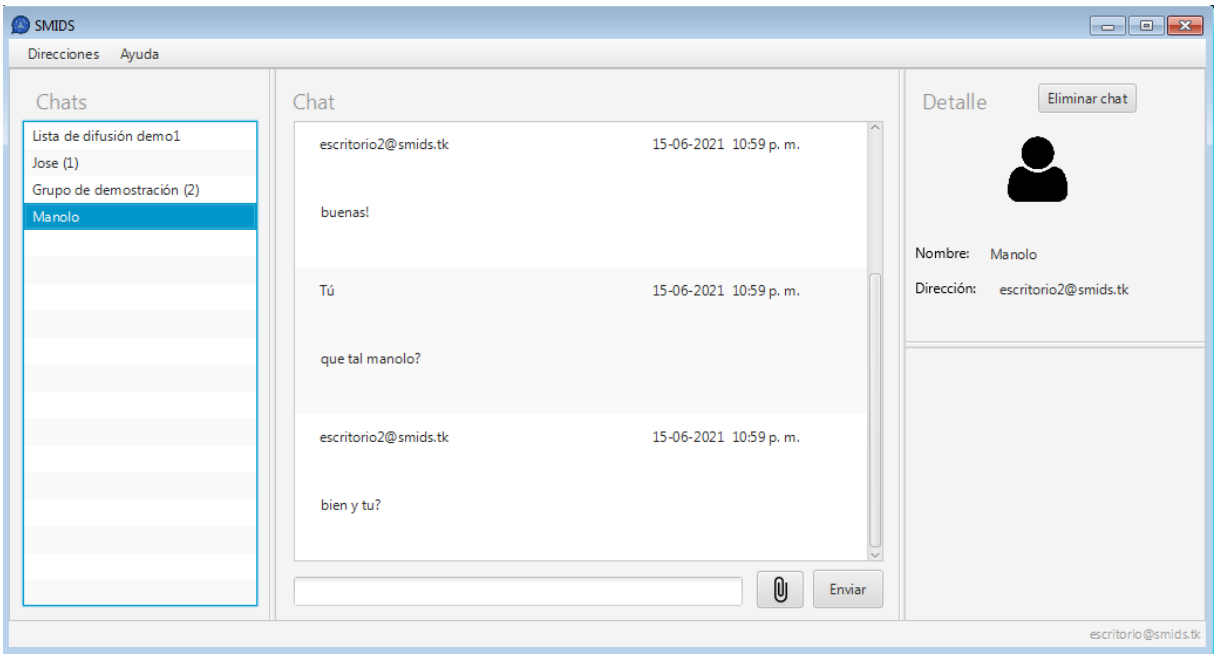

Figura 69: Vista de chat privado de la app de escritorio

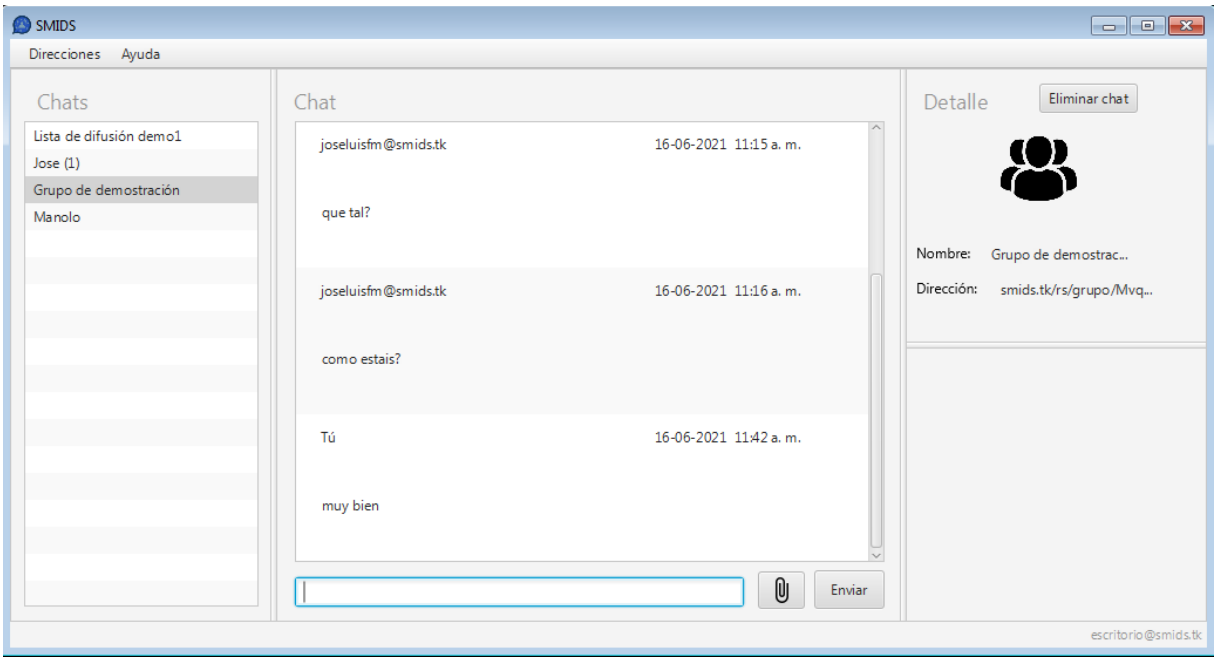

Figura 70: Vista de chat grupal de la app de escritorio

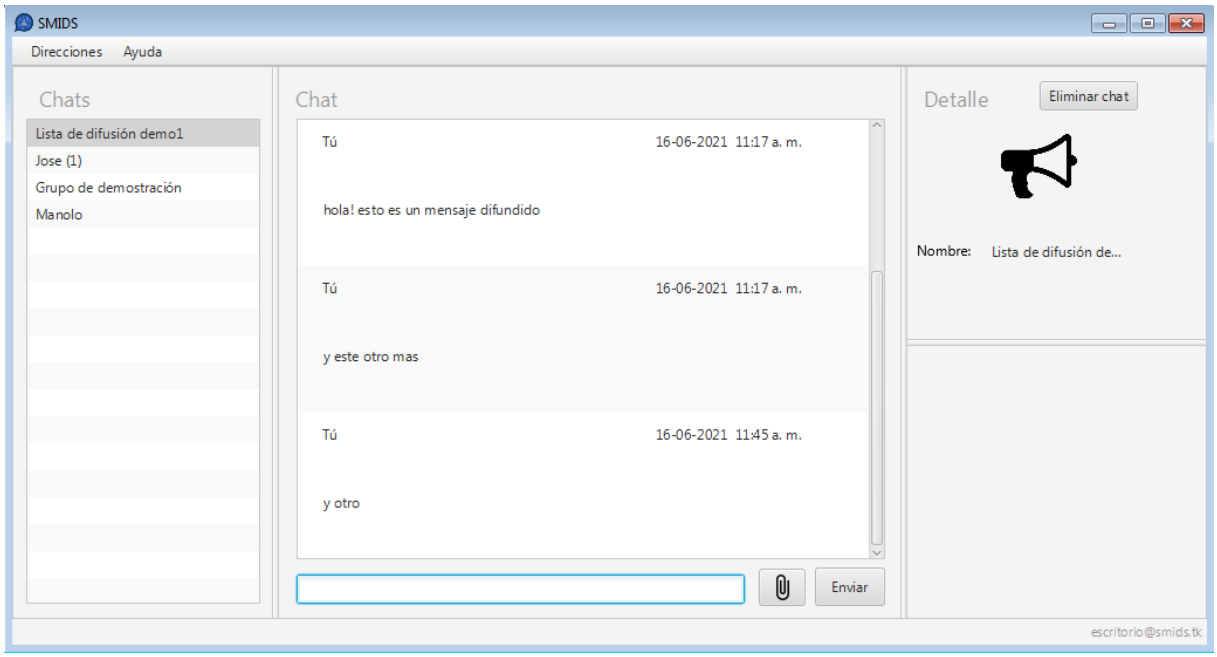

Figura 71: Vista de chat de difusión de la app de escritorio

C.3 Contactos

Para gestionar los contactos nos iremos a la barra superior de menú de la vista de chats y haremos click en Direcciones > Contactos:

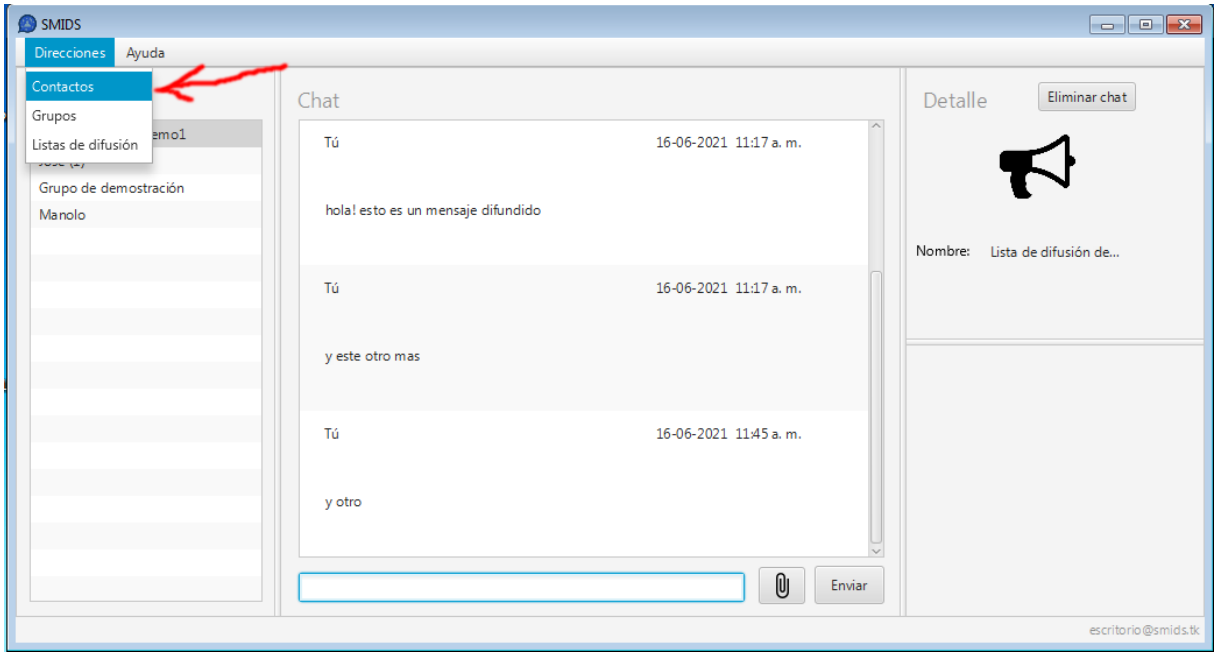

Figura 72: Vista de chats de la app de escritorio con el elemento "Contactos" del menú resaltado

Allí encontraremos en la parte izquierda un listado de todos los contactos registrados en la aplicación. Pulsando en cualquiera de estos, veremos la información de este en la parte derecha de la ventana. Desde aquí podremos editarlos, eliminarlos o escribirles. Pulsando el botón "+" podremos añadir un contacto nuevo:

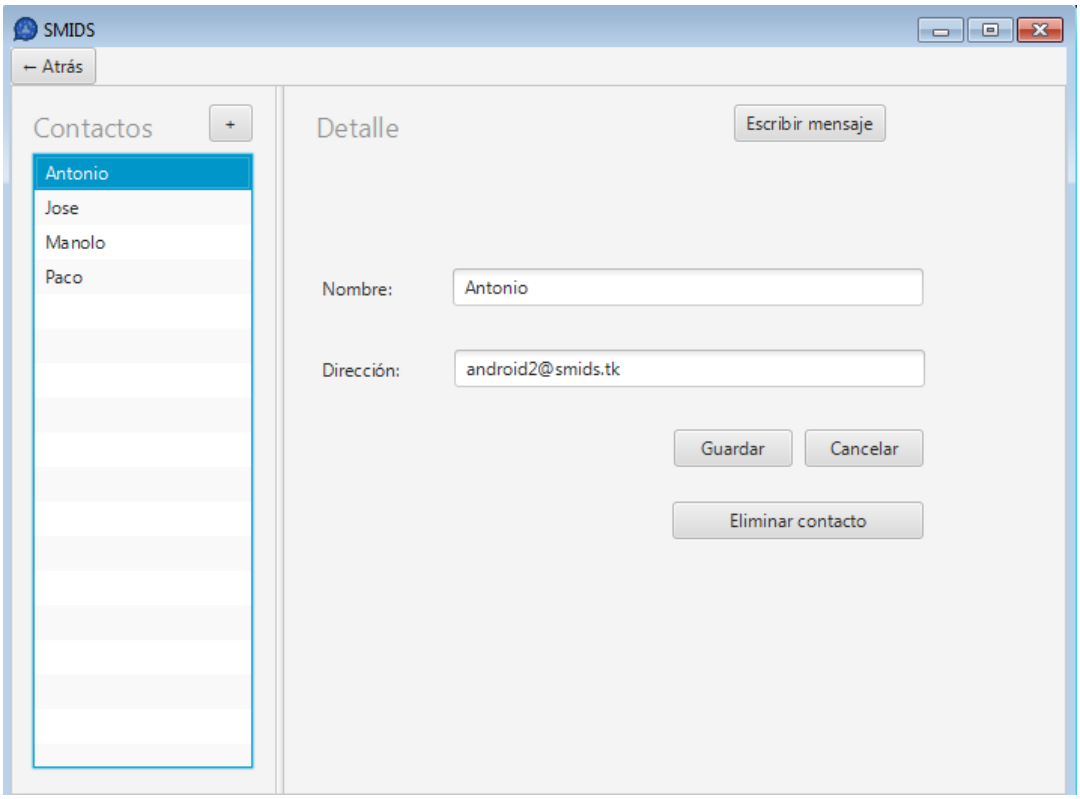

Figura 73: Vista de contactos de la app de escritorio

C.4 Grupos

Para gestionar los grupos nos iremos a la barra superior de menú de la vista de chats y haremos click en Direcciones > Grupos:

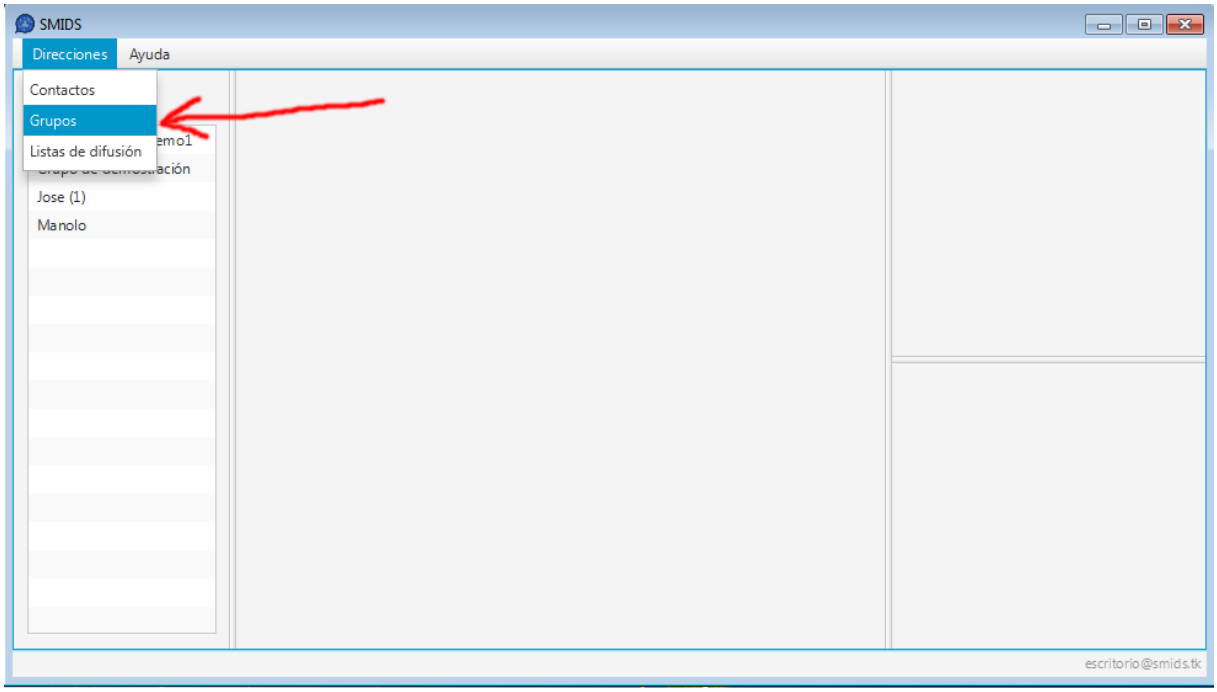

Figura 74: Vista de chats de la app de escritorio con el elemento "Grupos" del menú resaltado

Allí encontraremos en la parte izquierda un listado de todos los grupos a los que pertenecemos. Pulsando en cualquiera de estos, veremos la información de este en la parte derecha de la ventana. Desde aquí el usuario podrá, siempre y cuando este sea el creador del grupo, editarlos, agregar o quitar miembros y eliminarlos. Pulsando el botón "+" podremos añadir un grupo nuevo nuevo, y pulsando el botón "Escribir mensaje" podremos escribir en él (se creará un chat en caso de que no lo hubiese previamente).

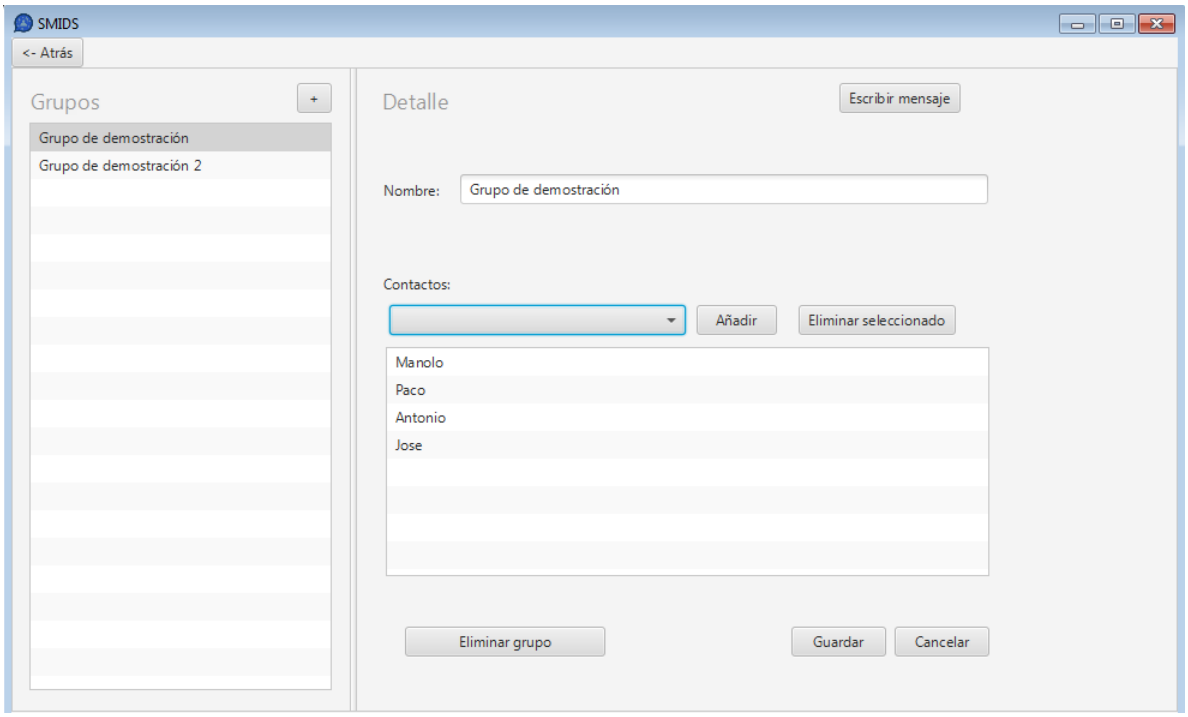

Figura 75: Vista de grupos de la app de escritorio

C.5 Listas de difusión

Para gestionar las listas de difusión nos iremos a la barra superior de menú de la vista de chats y haremos click en Direcciones > Listas de difusión:

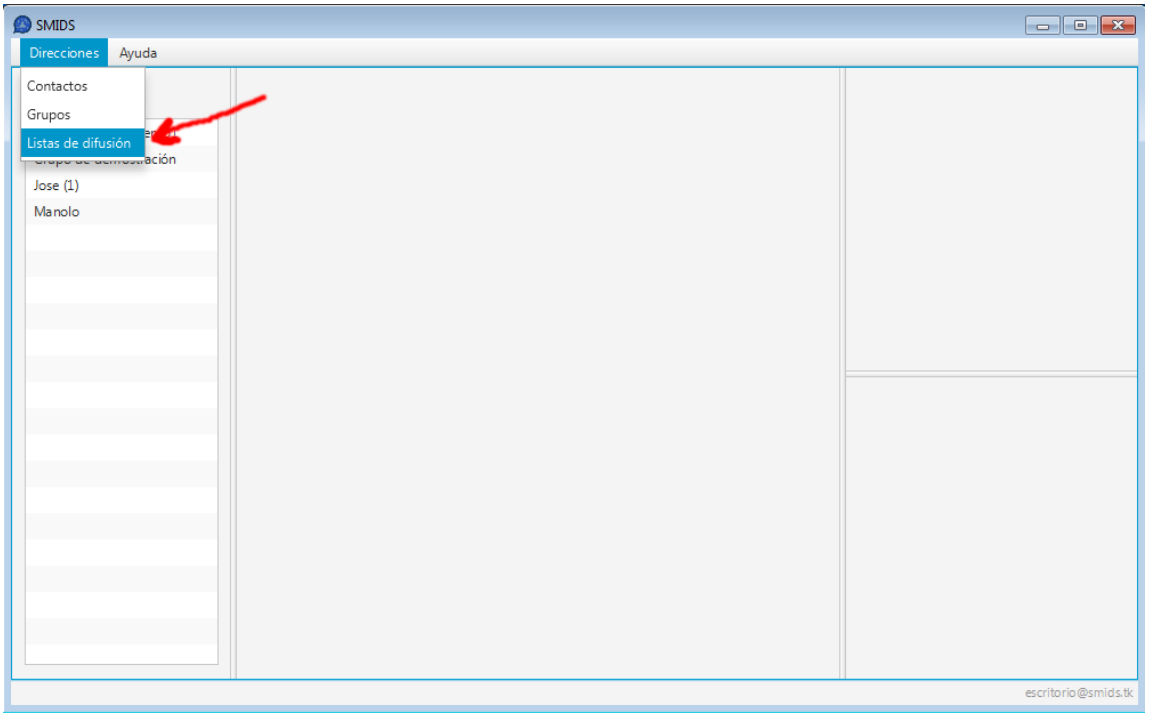

Figura 76: Vista de chats de la app de escritorio con el elemento "Listas de difusión" del menú resaltado

Allí encontraremos en la parte izquierda un listado de todas las listas de difusión que tengamos registradas en la aplicación. Pulsando en cualquiera de ellas, veremos la información de la lista en la parte derecha de la ventana. Desde ahí el usuario podrá editarlas, agregar o quitar miembros y eliminarlas. Pulsando el botón "+" podremos añadir una nueva lista de difusión, y pulsando el botón "Escribir mensaje" podremos escribir en ella (se creará un chat en caso de que no lo hubiese previamente).

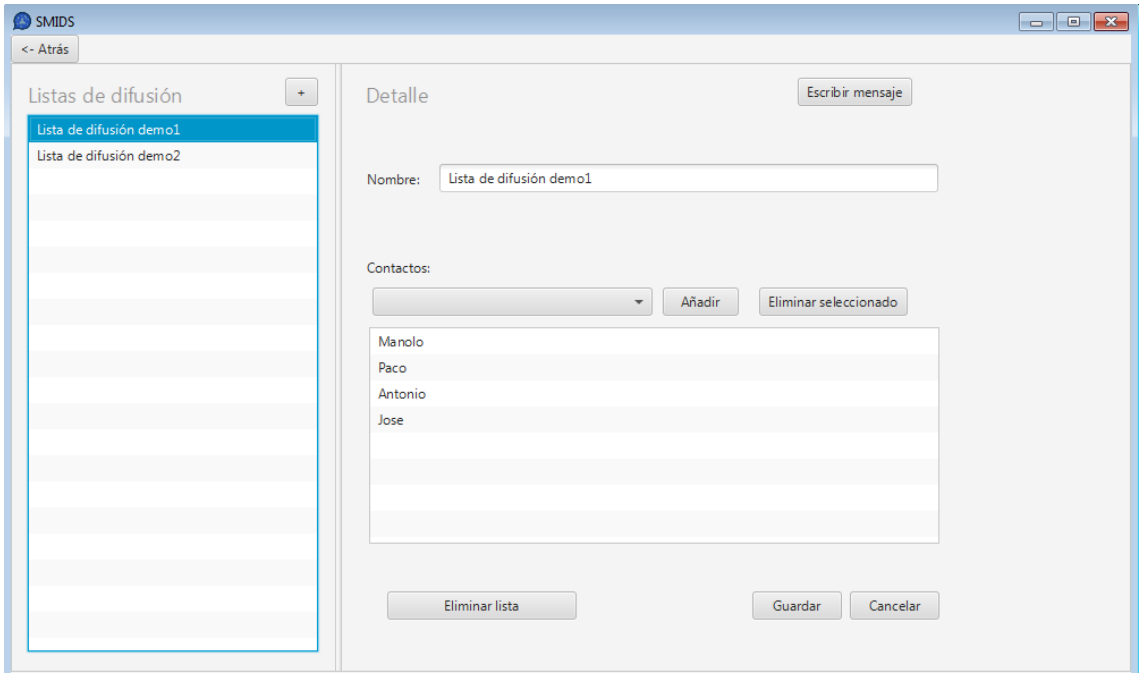

Figura 77: Vista de gestión de listas de difusión de la app de escritorio

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga