‘AppChat’

Caso Práctico

Tecnologías de Desarrollo de Software

3º de Grado en Ingeniería Informática

Trabajo Realizado Por:

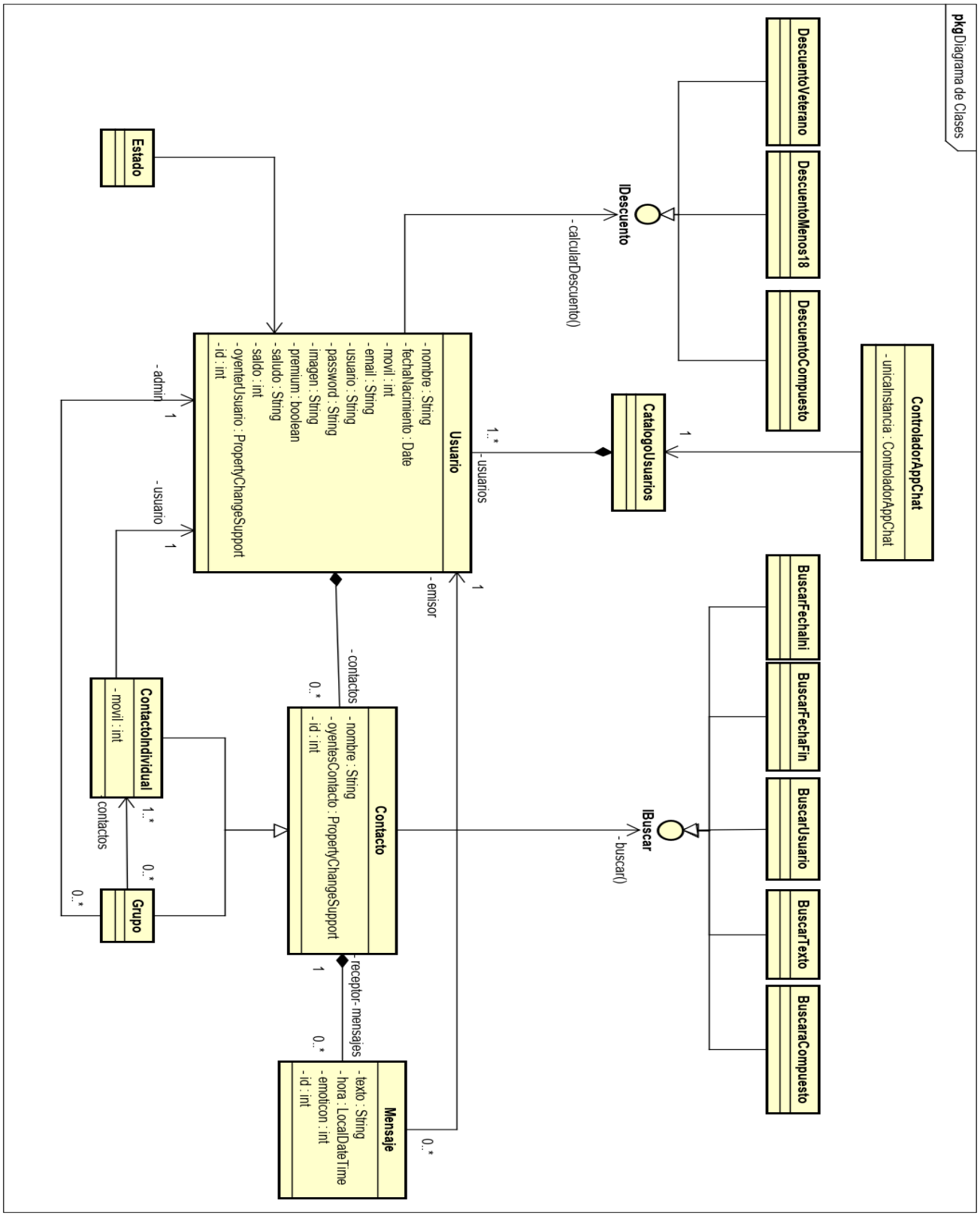
José Antonio García Cartagena

joseantonio.garciac@um.es

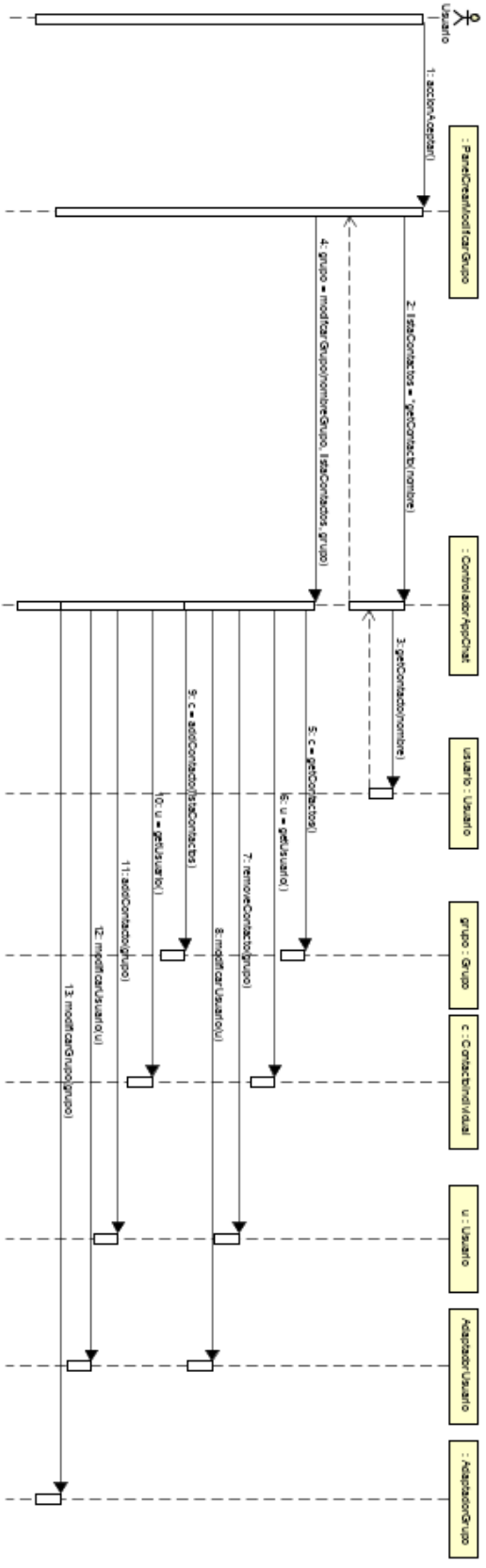
48700263V

Índice

1.Diagrama de clases del dominio.



2.Diagrama de Secuencia para la operación añadir un nuevo contacto.



Precondiciones:

- En clase <<PanelCrearModificarPanel>>:
  - existe la variable grupo de tipo Grupo.
  - existe la variable nombreGrupo de tipo String.

La Acción empieza en <<PanelCrearModificarGrupo>>. Esta clase extiende de JPanel y se utiliza tanto para la vista de creación como la modificación de un Grupo. Esto se diferencia por una variable de tipo entero que se inicializa junto a la instancia de la clase, si en esta se ha introducido un grupo o no.

Esta vista está compuesta principalmente por dos JTable y un JTextField. En la primera tabla se encuentran todos los nombres de los ContactosIndividuales del usuario que no se encuentran como miembros en el grupo. En la segunda JTable, se encuentran los nombres de los miembros del Grupo. Los nombres se pueden mover entre tablas gracias a dos botones, que eliminan un nombre de una lista y lo introduce en la otra. Dentro del JTextField se encuentre el nombre del Grupo, que es obligatorio.

Una vez accionado el método acciónAceptar(), se obtiene la lista de nombres del JTable con los miembros del Grupo y se obtiene los ContactosIndividuales(Acciones 1-3).

Con los ContactosIndividuales, el nombre del Grupo y el Grupo, se hace llamada del ControladorAppChat, que es el controlador de la aplicación (Accion 4).

Dentro del método modificarGrupo() de ControladorAppChat, se encuentran dos bucles:

- El primer bucle recorre los ContactosIndividuales del grupo y los elimina del mismo (Acción 5-8).
- El segundo bucle recorre los ContactosIndividuales obtenidos del JTable y los añade al Grupo (Acción 9 -12).

Por último, el grupo se modificar (Acción 13) y es el resultado del método.

Posteriormente, el grupo será enviado por parámetro a la clase PanelVistaPrincipal.

3. Arquitectura de la aplicación.

La aplicación está organizada por un modelo de tres capas (Presentación, Lógica de Negocio y Almacenamiento para mejorar la extensibilidad y legibilidad del código de la aplicación.

En cuanto a la presentación, hemos utilizado la tecnología Java Swing para la implementación de la interfaz.

Se ha creado un único JFrame principal en la clase <<VentanaMain>>. Desde esta clase se hace llamada a las clases PanelRegistroUsuario, PanelLoginUsuario y PanelVistaPrincipal. Las dos primeras se encargan de la introducción de información del Usuario que quiere utilizar la aplicación. Esta última es la vista principal de la Aplicación.

Desde PanelVistaPrincipal, podemos acceder a PanelBuscar, donde encontramos el uso del componente JDateChooser. Este panel es encargado de filtrar Mensajes de un contacto específico. Cada mensaje se visualiza con PanelMensajeBuscado; PanelCambiarImagen, donde encontramos el uso de JFileChooser y encargado de cambiar la imagen de Perfil del Usuario; PanelCambiarSaludo, encargado de cambiar el Saludo de Bienvenida del Usuario; PanelChat, donde podemos ver el uso de la clase BubbleText, y encargado de la visión de mensajes de un Contacto. De esta última clase hacemos llamada a la clase PanelEmoticonos, encargada de previsualizar los emoticonos y devolver el número del elegido para visualiazarlo en el PanelChat como BubbleText. Encontramos también PanelContactos, formado por PanelContacto de cada uno de los Contactos. Este último panel tiene un JButton invisible con el que se elige el Contacto y muestra en

PanelVistaPrincipal el PanelChat inicializado con este contacto. A continuación, tenemos PanelCrearModificarContacto y PanelCrearModificarGrupo, con el que se crean y modifican los ContactosIndividuales y Grupos respectivamente. La capa de presentación tiene conocimiento de la lógica de negocio y es por ello que pueden hacer llamadas a los métodos del controlador. En estas interfaces no se desarrolla la lógica de negocio.

En Cuanto a la Lógica de Negocio, como hemos mostrado previamente en el apartado 1.Diagrama de Clases de Dominio, vemos como gira todo en torno a la clase Usuario. Es debido a esto, que se utiliza un Catálogo de Usuario para tener toda la información de cada Usuario. Encontramos junto al usuario, una lista de Contactos, que pueden ser ContactoIndividuales o Grupos. Estos últimos están compuestos por una agenda de ContactosIndividuales junto al usuario creador o administrador. En cuanto a los ContactosIndividuales, están compuestos, por un usuario y su número de teléfono. Tanto ContactosIndividuales como Grupos son herencia de una clase Contacto.No cabe decir, que todo contacto tiene una lista junto a la mensajería utilizada. Cada Usuario tiene la opción de un Estado.

Todo esto viene “protegido” por una fachada (ControladorAppChat), que junta las tres capas.

En cuanto al Almacenamiento, se ha utilizado un Patrón DAO. Para cada Clase de la Lógica de Negocio, se ha creado una interfaz DAO, que proporciona los métodos CRUD para crear, recuperar, modificar y borrar datos almacenados.

Un ejemplo sería el de IAdaptadorUsuario, donde se encuentran las acciones de registrarUsuario, modificarUsuario, recuperarUsuario, recuperarTodosUsuarios.

Todas estas interfaces están implementadas en clases adaptadores. En este caso se han implementado junto a una Factoria de Servicio de Persistencia y siendo todas singleton. Se utiliza una clase auxiliar PoolDAO para reconstruir los objetos que llaman a diferentes Adaptadores.

Por otro lado, se recopilan todos los adaptadores de una misma familia en una clase FactoriaDAO singleton. En este caso solo hay una familia, la que utiliza la Factoria de Servicio de Persistencia.

En resumen, un Patrón DAO, es un patrón compuesto por unos adaptadores, todos unidos en una misma Factoría, y a su vez todos singletón.

4.Patrones de diseño.

En este apartado nos vamos a centrar principalmente en la capa de Lógica de Negocio, ya que es donde se encuentran la mayoría de patrones. A su vez, en la capa de Almacenamiento, se encuentra el Patrón DAO explicado en el punto anterior.

Para la acción de Buscar en Contacto y Descuento para ser Premium en Usuario, hemos utilizado el Patrón de Estrategia, como patrón de comportamiento, junto a Composite, como patrón estructural. Es decir, permitimos que se puedan añadir más de un tipo de Descuento, y a su vez que se computen individualmente o colectivamente.

Encontramos un patrón estructural de Fachada con ControladorAppChat, ya que permite que la capa de Visión no gestione sobre la Lógica de Negocio ni Almacenamiento. A su vez la capa de Almacenamiento, tampoco puede gestionar la Lógica de Negocio.

Para la actualización de la capa de Visión, se ha creado un patrón de comportamiento con la interfaz PropertyChange. Con esto, se ha añadido una variable PropertyChangeSupport

en las clases del modelo de negocio de Usuario y Contacto. Cada vez que se creen Contactos, se cambie la imagen del Usuario, el Estado, se eliminen Contactos o se envíen o borren mensajes, los oyentes que se encuentren dentro de esta variable, serán notificados mediante un evento. De esta forma nos olvidamos de los cambios internos hechos por el usuario, que serán actualizados automáticamente. Para los cambios externos, ejemplo un mensaje escrito por el otro usuario, se ha creado una clase singleton llamada ControladorActualizarUsario, que implementa un reloj el cual, cada 5 segundos actualiza el CatalogodeUsuarios. Compara el Usuario del controlador y del catálogo. Cada cambio que encuentre, lo actualiza, de esta forma podemos actualizar la vista tanto por cambios internos como externos y a su vez tener actualizado el usuario ante cambios.

5.Componentes.

Utilizamos varios componentes, donde se encuentran:

- JFileChooser. Componente bien conocido dentro de Java Swing. Este componente nos permite buscar un fichero dentro de nuestro ordenador, devolviéndonos el path o ruta del mismo. Se ha utilizado para la obtención de la imagen del usuario y la obtención del archivo .txt para cargar mensajes. También nos permite crear filtros para sólo aceptar archivos de un tipo. Esto se ha utilizado también en la práctica.
- JCalendar. Componente también conocido dentro de Java Swing. Es un componente que nos permite mostrar un calendario del que podemos obtener una instancia de la clase Date del día seleccionado. Se ha utilizado en la capa de vista para elegir la fecha de nacimiento del Usuario o filtrar los mensajes a buscar por fecha de Inicio o Fin.

- ChatWindowLib. Este componente nos ha sido entregado como material a utilizar en la práctica. Se ha utilizado para visualizar los mensajes en las vistas de chats. Viene con un repertorio de emotoiconos también utilizados.
- ITextPdf: Este componente nos permite crear archivos pdf. Se ha utilizado para crear un pdf en los Usuarios Premium junto a su información y la de los contactos del mismo.
- XChart: Este componente nos permite crear histogramas. Se ha utilizado en la opción “Información. Creamos 2 histogramas como opción Premium.
- Luz: Este componente también ha sido proporcionado por la Asigntaura. Este Componente visualiza un botón encendido y apagado. Para estar al tanto de su actualiación se debe implementar la interfaz IEnteradoListener
- Parse: Este componente se ha creado por el estudiante. Una vez el componente luz ha sido activado, manualmente se activa este componente también, que dibuja un JFileChooser con el que obtenemos un archivo de Texto con mensajes a importar y nos da a alegir la Plataforma de donde se han exportado estos mensajes (IOS o ANDROID). Devuelve un evento con la plataforma, el fichero y el estilo de escritura.

6.Tests unitarios

Principalmente se ha hecho unos pocos tests, sobre algunos métodos de ControladorAppChat. Se han creado tests Unitarios con la líbreia JUnit4 para los métodos de creación y modiciafición de Contactos y envío de mensajes. Se ha comparado lo que devuelve el método y lo que se obtiene de la capa de almacenamiento.

Los 7 tests se han pasado sin problemas. Hemos utilizado la etiqueta @Before para crear previamente unas instancias.

7.Manual de usuario

Al inicializar el programa, se encuentra un panel donde se debe introducir nombre y contraseña. Si no se está registrado, acceder a la ventana de Registro mediante el botón “Registrar”.

Una vez introducidos los datos necesarios para acceder a la aplicación, encontramos la Ventana Principal. Esta venta está compuesta por una barra superior con varios botones. La parte derecha, compuesta por los Contactos del Usuario y la parte Derecha que, de momento se encuentra vacía.

En cuanto a la barra Superior, de izquierda a derecha, encontramos los siguientes botones:

- Opciones de Usuario. En las dos primeras columnas se encuentran el nombre del Usuario y el saludo. Las dos siguientes columnas nos permiten cambiar el Saludo y la Imagen del Usuario.
- Opciones de Contactos. En esta pestaña podemos crear un ContactoIndividual nuevo, un Grupo o convertirnos en Premium. Una vez convertidos en Premium nos permitirá crear un archivo PDF junto a la información de nuestro Usuario o mostrar información de Uso de la aplicación.

    - Al seleccionar la opción Crear Contacto e introducir unos datos válidos. En el lado derecho de la vista aparecerá nuestro nuevo Contacto, y en el derecho nuestro Chat junto a más Opciones.
    - Al seleccionar la Opción Crear Grupo e introducir los datos válidos. En el lado derecho de la vista aparecerá nuestro nuevo Grupo, y en el derecho nuestro Chat junto a más Opciones.
- Opción de Estado. En esta pestaña podemos crear y ver Estados de nuestros contactos.

En cuanto a la lista de Contactos que encontramos en la izquierda, a cada mensaje enviado de cada Chat o la actualización de imágenes del otro usuario, se verán reflejadas en el correspondiente contacto. Si queremos abrir cualquier chat solo tenemos que pulsar en el contacto y se mostrará una ventana idéntica a la que se muestra cuando un Contacto es recién creado. A no ser que hubiera de antes mensajes enviados. Tras seleccionar un Contacto, vemos que se nos presentan nuevas Opciones:

- Si pulsamos sobre la imagen del Contacto nos aparecerá el nombre dado al Contacto y su número de teléfono. En el caso de un Grupo estas funciones no existen.
- Si pulsamos sobre el icono de la lupa, se nos mostrará en el lado derecho de la Vista unos filtros para poder buscar unos mensajes específicos. Todos los parámetros son opcionales. Si el Contacto en el que queremos buscar es un Grupo, tendremos un apartado más que busca entre Usuarios del Grupo.
- Si pulsamos sobre el icono de tres barras negras, accedemos a las opciones del Contacto. Podemos:

- Modificar Contacto: Ventana idéntica a la de creación de Contacto, pero con la información del mismo rellenado. Se puede modificar todo.
- Eliminar Contacto: Se eliminará el Contacto. Si es un grupo tienes que ser el administrador
- Eliminar Mensajes: Se eliminarán los mensajes del Contacto. Si es un grupo tienes que ser el aministrador. Si es un ContactoIndividual, el otro Usuario mantendrá los mensajes.

- Si pulsamos sobre el icono que parece un botón, tendremos la posibilidad de importar mensajes directamente desde la aplicación WhatsApp.

En cuanto a la Vista del Chat, podemos escribir cualquier mensaje y enviarlo presionando la tecla ENTER. Si queremos enviar un emoticono, sólo tenemos que pulsar a la imagen del emoticono y elegir el que queramos.

Todas estas vistas se pueden ver en el Anexo de esta documentación.

8.Observaciones finales

Esta ha sido una práctica muy completa y que se acerca bastante a la formación de un proyecto laboral. Hemos utilizado desde la herramienta Maven para dependencias sin necesidad de buscar archivos jar directamente, sino utilizar enlaces a repositorios y añadirlos al archivo pom.xml, hasta repositorios compartidos para poder trabajar con más personas y tener el trabajo guardado junto a sus versiones o actualizaciones. Esto lo he utilizado bastante ya que es muy sencillo retroceder si ha habido algún error.

Se ha trabajado con componentes y a su vez se han creado por lo que hemos podido aprender tanto Java Swing desde una vista de diseño o gráfica como de escritura de código.

También hemos aprendido un modelo de 3 capas desarrollado completamente por el estudiante.

He podido aprender patrones de creación, estructurales y comportamiento.

A su vez ha sido muy desafiante crear un modelo de negocio en el que sus modificaciones se mostraran incluso cuando no ha sido modificada por el Usuario. Es decir, lo más difícil desde mi punto de perspectiva ha sido crea un programa “NO asíncrono”.

Al crear este trabajo yo sólo, ha creado una carga temporal grande. Se ha necesitado más de 160 horas para desarrollar este proyecto.

9. Anexo

Ilustración 1. Ventana Login

Ilustración 2. Ventana Registrar

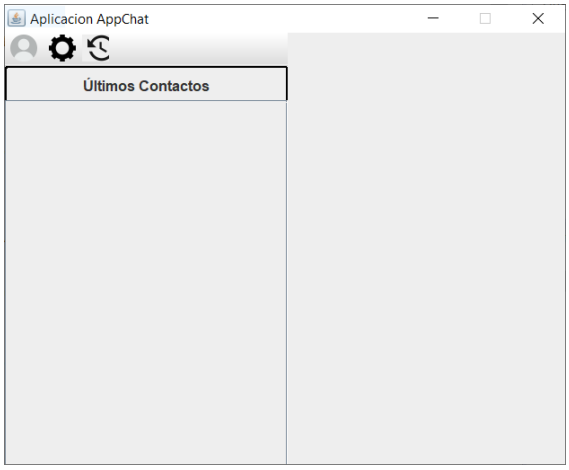


Ilustración 3. Ventana Principal

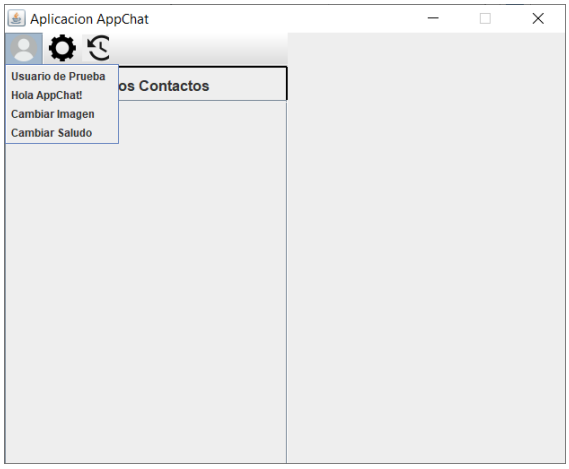


Ilustración 4. Opciones de Usuario

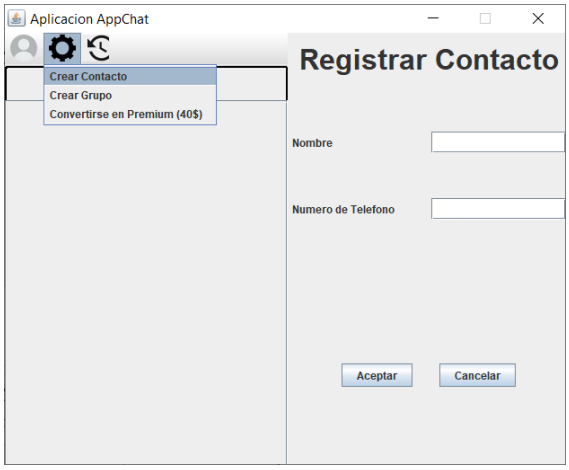


Ilustración 5. Vista Opciones de contacto junto a Crear Contacto

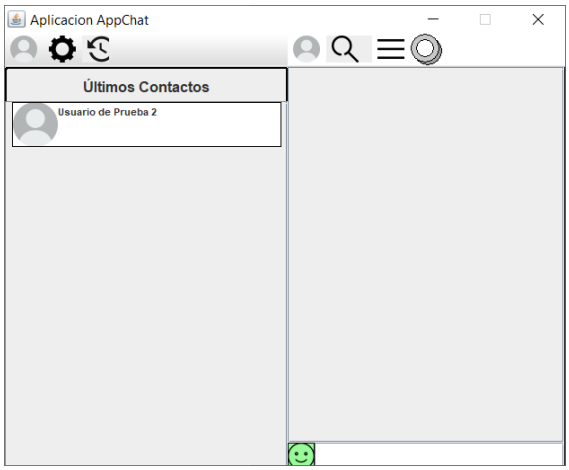


Ilustración 6. Vista tras Crear un Contacto

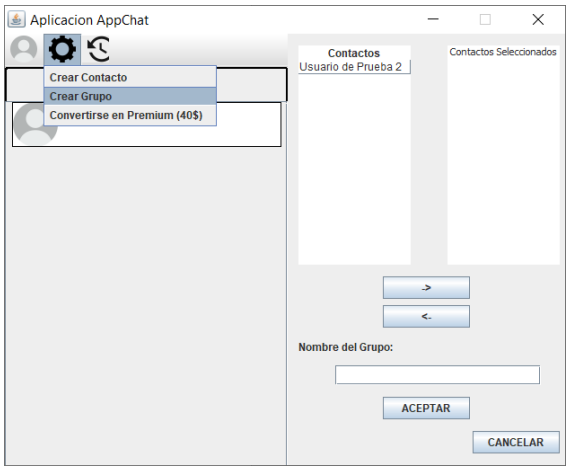


Ilustración 7. Vista Crear Grupo

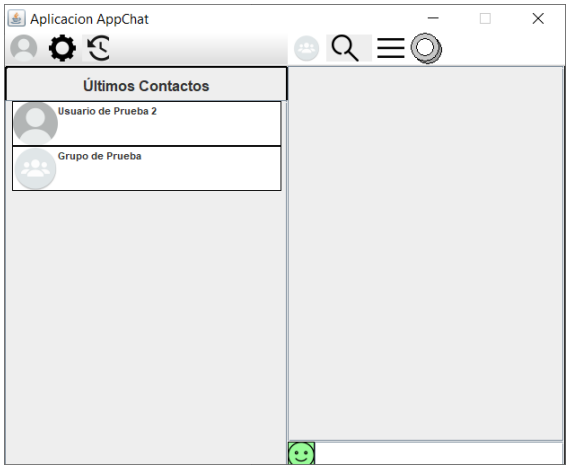


Ilustración 8. Vista tras Crear un Grupo

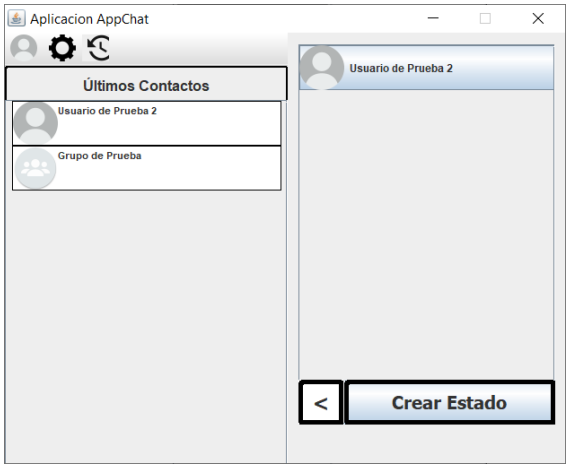


Ilustración 9. Vista Estados

Ilustración 10. Vista de un Estado

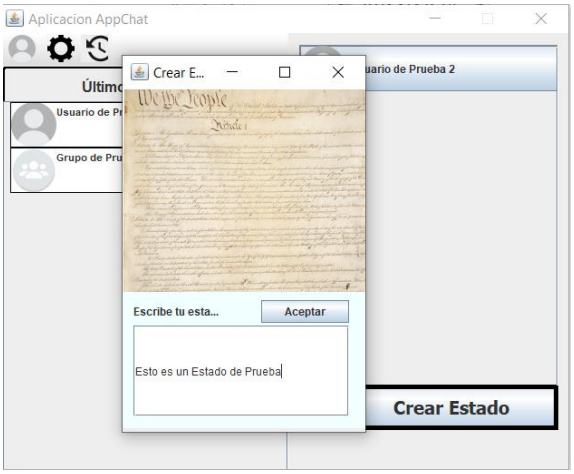


Ilustración 11. Vista de la creación de un Estado

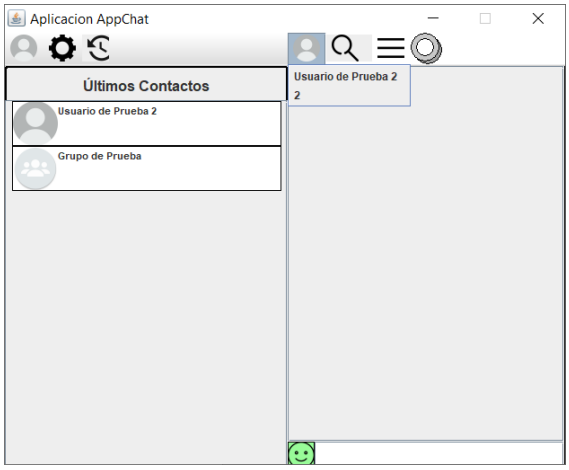


Ilustración 12. Información de Contacto

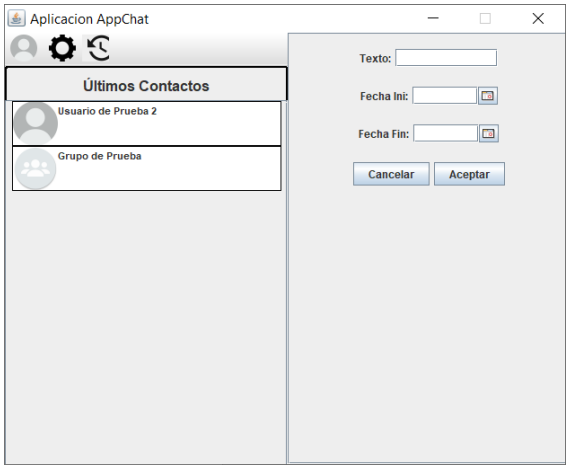


Ilustración 13. Ventana Buscar

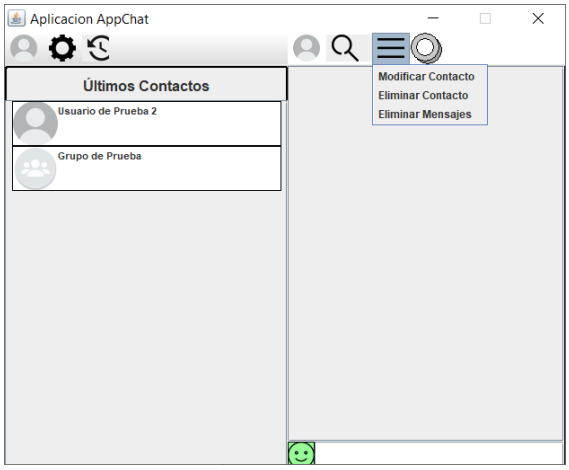


Ilustración 14. Opciones de Contacto

Ilustración 15. Vista Modificar Contacto

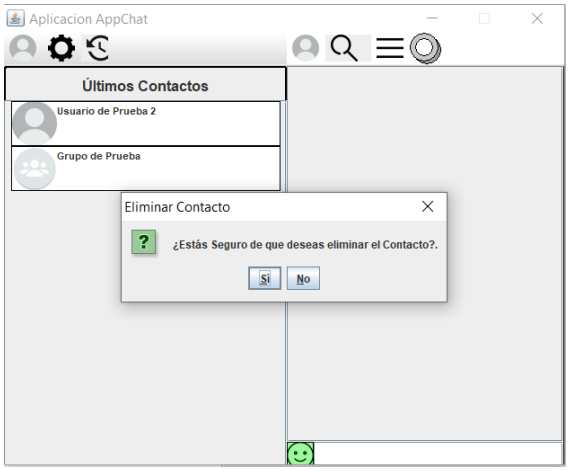


Ilustración 16. Advertencia tras pulsar "Eliminar Contacto"

Ilustración 17. Advertencia tras pulsar "Eliminar Mensajes"

Ilustración 18. Vista después del Envío de un Mensaje

Ilustración 19. Vista de Selección de Emoticonos

Ilustración 20. Vista después de Enviar un Emoticono