

UNIVERSIDAD  
DE MURCIA

# *Documento de prácticas* *Tecnologías del Desarrollo Software*

*Juan Carlos Valera López G1.1*

*DNI:21067244A*

*Pedro José Fernández Escámez G1.2*

*DNI:58450119N*

*Profesor: José Ramón Barceló Hoyos*

*Fecha: 10/05/2025*



# ÍNDICE DE CONTENIDO

1. Introducción .....	3
2. Historias de usuario y diagrama de clases del dominio .....	4
2.1. Historias de usuario .....	4
2.2. Diagrama de clases del dominio .....	7
2.3. Diagrama de secuencia para nuevo contacto en grupo .....	7
3. Arquitectura de la aplicación y decisiones de diseño .....	8
3.1. Arquitectura General.....	8
3.2. Decisiones de Diseño Relevantes .....	8
4. Patrones de diseño utilizados.....	10
5. Manual de usuario.....	13
5.1. Registro de un Nuevo Usuario.....	13
5.2. Inicio de Sesión.....	14
5.3. Ventana Principal y Navegación .....	14
5.4. Gestión de Perfil .....	15
5.5. Gestión de Contactos y Grupos .....	16
5.6. Envío de Mensajes .....	20
5.7. Búsqueda de Mensajes.....	22
5.8. Funcionalidad Premium.....	23
6. Conclusión .....	25

# ÍNDICE DE FIGURAS

Figura 1: Login AppChat.....	13
Figura 2: Registro AppChat .....	13
Figura 3: Login AppChat (2).....	14
Figura 4: Ventana Principal AppChat .....	14
Figura 5: Botones de navegación AppChat.....	15
Figura 6: Panel usuario AppChat .....	15
Figura 7: Información usuario AppChat .....	15
Figura 8: Modificar usuario AppChat .....	16
Figura 9: Ventana contactos AppChat .....	16
Figura 10: Crear contacto AppChat .....	17
Figura 11: Contacto creado.....	17
Figura 12: Contacto aparece en Ventana Principal .....	17
Figura 13: Modificar Contacto AppChat .....	18
Figura 14: Crear Grupo AppChat (1) .....	18
Figura 15: Crear Grupo AppChat (2) .....	19
Figura 16: Crear Grupo AppChat (3) .....	19
Figura 17: Modificación de Grupo .....	20
Figura 18: Chat Ventana Principal.....	20
Figura 19: Mensaje en caja de texto .....	21
Figura 20: Mensaje enviado .....	21
Figura 21: Selección de emoticonos.....	21
Figura 22: Emoticono enviado .....	21
Figura 23: Envío de Mensaje en Grupo.....	22
Figura 24: Botón de buscar mensajes .....	22
Figura 25: Ventana Buscar Mensajes .....	22
Figura 26: Búsqueda de mensajes (1).....	23
Figura 27: Búsqueda de mensajes (2).....	23
Figura 28: Botón Activar Premium.....	23
Figura 29: Ventana Actualización Premium .....	24
Figura 30: Botón Premium Activo .....	24
Figura 31: Ventana exportación PDF .....	24

# *1. Introducción*

En el marco de la asignatura “Tecnologías de Desarrollo Software” del Grado en Ingeniería Informática de la Universidad de Murcia, se ha desarrollado el proyecto “AppChat”. Este proyecto consiste en una aplicación de escritorio de mensajería instantánea, inspirada en funcionalidades de aplicaciones populares como WhatsApp o Telegram. El objetivo principal ha sido aplicar los conocimientos adquiridos en el curso para diseñar e implementar una aplicación robusta, mantenible y funcional, que abarca desde el registro y autenticación de usuarios, la gestión de contactos y grupos, hasta el intercambio de mensajes y la oferta de características avanzadas para usuarios Premium.

La aplicación se ha desarrollado en Java, utilizando Swing para la construcción de la interfaz gráfica. La persistencia de los datos se maneja a través de una capa de acceso a datos (DAO) que interactúa con un servicio de persistencia proporcionado. La gestión de dependencias y la construcción del proyecto se han realizado con Maven, y el control de versiones con Git.

Esta memoria detalla el proceso de desarrollo, comenzando por la definición de requisitos mediante historias de usuario y el modelado del dominio. Se describe la arquitectura software adoptada, las decisiones de diseño más significativas y, de manera particular, se profundiza en los patrones de diseño implementados. Finalmente, se incluye un manual de usuario para guiar en el uso de la aplicación y se presentan unas conclusiones sobre el trabajo realizado y el aprendizaje obtenido.

## 2. Historias de usuario y diagrama de clases del dominio

### 2.1. Historias de usuario

#### 2.1.1. Actividad 1: Registro y Login de usuarios

**Objetivo:** Permitir a los usuarios acceder a la aplicación y sus funcionalidades DESPUÉS de registrarse y autenticarse.

- **Historia 1.1: Registro de Nuevo Usuario**

- Como usuario, quiero registrarme proporcionando mi nombre, fecha de nacimiento, email, imagen de perfil, número de teléfono móvil, contraseña y, opcionalmente, un mensaje de saludo, para acceder a la aplicación.
- *Criterios de Aceptación:* Todos los campos obligatorios deben de estar completos correctamente y recibir un mensaje de confirmación. Debe mostrarse un mensaje de error si algún campo es incorrecto o está incompleto.

- **Historia 1.2: Inicio de sesión**

- Como usuario, quiero acceder a mi cuenta con mi número de teléfono y mi contraseña para acceder a mi cuenta.
- *Criterios de Aceptación:* Puedo iniciar sesión si las credenciales son válidas o recibir un mensaje de error si son incorrectas. Si las credenciales son correctas debo ser redirigido a la pantalla principal de la aplicación.

#### 2.1.2. Actividad 2: Gestión de contactos

**Objetivo:** Permitir la gestión de una lista de contactos donde poder buscar otros usuarios o un grupo de contactos.

- **Historia 2.1: Agregar un contacto**

- Como usuario, quiero poder agregar usuarios a mi lista de contactos por su nombre y teléfono, para identificar a los demás usuarios.
- *Criterios de Aceptación:* El sistema debe permitir agregar un contacto introduciendo un número de teléfono válido y un nombre. Si el número de teléfono no está registrado en la base de datos, se debe notificar al usuario con un mensaje de error.

- **Historia 2.2: Agregar contacto a grupo de contactos (Crear Grupo)**

- Como usuario, quiero poder añadir contactos a un grupo de contactos para poder mandar un mensaje a varios contactos a la vez.
- *Criterios de Aceptación:* El sistema debe permitir crear como contacto un grupo de contactos, el cual estará compuesto únicamente por usuarios, nunca por otros grupos. Debe tener un nombre y opcionalmente una foto.

- **Historia 2.3: Agregar a contactos al recibir un mensaje de un número desconocido (opcional)**

- Como usuario, quiero poder agregar usuarios a mi lista de contactos por su nombre, habiendo recibido un mensaje de número sin registrar, para poder identificar otros usuarios.
- *Criterios de Aceptación:* El sistema debe permitir agregar un contacto introduciendo un número de teléfono válido y un nombre. Si el número de teléfono no está registrado en la base de datos, se debe notificar al usuario con un mensaje de error.

- **Historia 2.4: Buscar contactos**
  - Como usuario, quiero poder buscar contactos por nombre o número de teléfono para encontrar más fácilmente la persona a la que quiero enviarle un mensaje.
  - *Criterios de Aceptación:* Para permitir la escalabilidad de la aplicación, el sistema debe permitir buscar mediante subcadenas el nombre o número de un contacto.
- **Historia 2.5: Editar contactos (opcional)**
  - Como usuario, quiero editar la información de mis contactos para mantenerlos actualizados.
  - *Criterios de Aceptación:* El sistema debe de permitir cambiar el nombre de un contacto ya guardado.
- **Historia 2.6: Editar nombre del grupo (opcional)**
  - Como usuario, quiero poder editar el nombre de mis grupos para mantenerlos actualizados.
  - *Criterios de Aceptación:* El sistema debe de permitir cambiar el nombre de un grupo ya creado.
- **Historia 2.7: Editar integrantes grupos (opcional)**
  - Como usuario, quiero poder editar los integrantes de mis grupos para mantenerlos actualizados, ya sea para añadir o eliminar contactos del grupo.
  - *Criterios de Aceptación:* El sistema debe de permitir cambiar los integrantes de un grupo ya creado y asegurarse de que nunca se quede un grupo sin integrantes. Debe de haber al menos un contacto en un grupo.

#### 2.1.3. Actividad 3: Envío de mensajes

**Objetivo:** Permitir la comunicación entre dos usuarios REGISTRADOS a través de mensajes de texto y/o emoticonos.

- **Historia 3.1: Enviar un mensaje**
  - Como usuario, quiero enviar mensajes a mis contactos y a mis grupos para poder comunicarme de manera efectiva.
  - *Criterios de Aceptación:* El sistema debe permitir el envío de mensajes a cualquier contacto o grupo en la lista de contactos del usuario. El mensaje puede contener texto o un emoticono. Para cada mensaje debe registrarse nombre del emisor, receptor y la fecha y hora de envío. Si el usuario receptor no está en la lista de contactos, se mostrará su número de teléfono en lugar del nombre.
- **Historia 3.2: Recibir mensaje (con extensión opcional)**
  - Como usuario, quiero poder recibir mensajes, ya sean de uno de mis contactos o no.
  - *Criterios de Aceptación:* El sistema debe permitir recibir mensajes de cualquier usuario de la aplicación. Si el número del emisor está guardado previamente en la lista de contactos del receptor, debe aparecer el nombre; si no, el número del emisor. En el momento en que el usuario recibe un mensaje de un número que no tiene guardado en sus contactos, este pasa a estar dentro de sus contactos, teniendo como nombre el número de teléfono, con opción de añadirle un nombre posteriormente.

#### 2.1.4. Actividad 4: Búsqueda de Mensajes

**Objetivo:** Facilitar la localización de mensajes enviados a través de diversos criterios de búsqueda.

- **Historia 4.1: Buscar mensajes por fragmentos de texto, número teléfono, nombre de contacto o combinación de estas.**
  - Como usuario, quiero poder buscar mensajes por fragmento de texto, número de teléfono, nombre de contacto o combinación de estas para poder encontrar mensajes más fácilmente.
  - *Criterios de Aceptación:* Debo poder escribir una subcadena en los campos de búsqueda y obtener una lista con todos los mensajes que contengan esa subcadena como texto, nombre del emisor/receptor o número de teléfono. El sistema debe de estar preparado para admitir nuevos filtros.

#### 2.1.5. Actividad 5: Funcionalidad Premium

**Objetivo:** Ofrecer un servicio especial para usuarios premium.

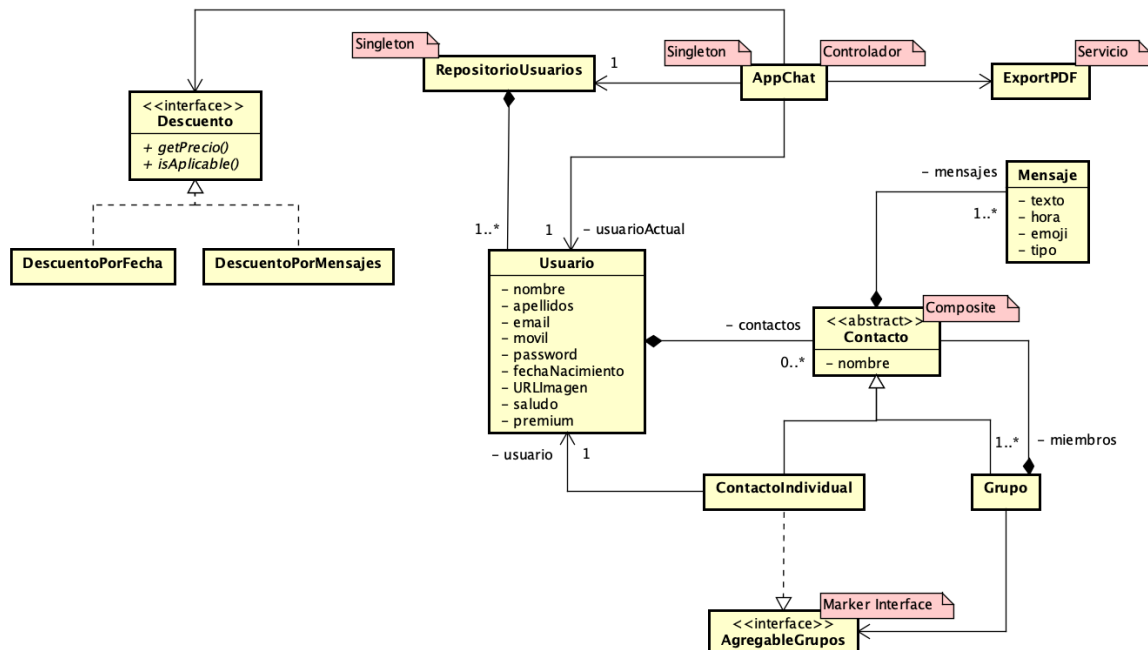
- **Historia 5.1: Generar un documento PDF con mensajes intercambiados**
  - Como usuario Premium, quiero poder exportar en formato PDF todos los mensajes intercambiados con un usuario, para poder guardar la conversación en mi almacenamiento local.
  - *Criterios de Aceptación:* El sistema debe de permitir la conversión de los mensajes a un formato PDF, ya sea una conversación con una persona, o con un grupo (en el caso de un grupo sólo aparecerán los mensajes que el usuario ha enviado a los múltiples contactos que conforman el grupo).
- **Historia 5.2: Activar/Desactivar Cuenta Premium:**
  - Como usuario, quiero poder activar mi cuenta a Premium pagando una suscripción, y también poder desactivarla.
  - *Criterios de Aceptación:* El sistema permite el pago (simulado) aplicando descuentos. El estado Premium da acceso a funcionalidades adicionales. La desactivación revierte este acceso.
- **Historia 5.3: Descuentos automáticos (Opcional):**
  - Como usuario, quiero que cuando vaya a activar mi suscripción premium, se compruebe si se me puede aplicar un descuento, y que se seleccione el que más me beneficie para pagar menos dinero.
  - *Criterios de Aceptación:* El sistema debe de comprobar las condiciones de los descuentos y aplicar el mejor descuento en cada caso.

#### 2.1.6. Actividad 6: Personalización

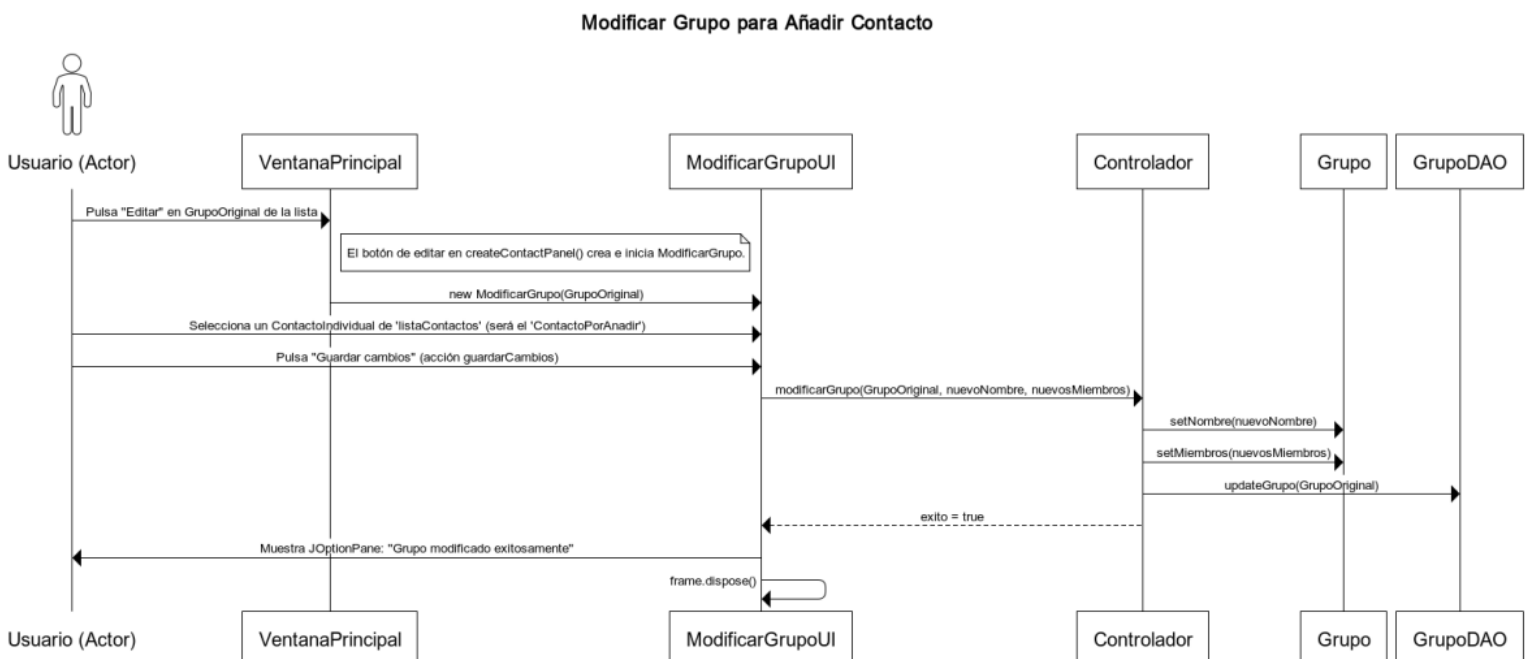
**Objetivo:** Ofrecer opciones de personalización al usuario.

- **Historia 6.1: Personalizar imagen de perfil.**
  - Como usuario, quiero poder cambiar mi imagen de perfil con una URL, para asignar una identidad simbólica a mi cuenta.
  - *Criterios de aceptación:* El sistema debe de permitir al usuario editar su imagen de perfil mediante una URL.

## 2.2. Diagrama de clases del dominio



## 2.3. Diagrama de secuencia para nuevo contacto en grupo





## 3. Arquitectura de la aplicación y decisiones de diseño

### 3.1. Arquitectura General

AppChat se ha desarrollado siguiendo una arquitectura multicapa, implementando el patrón arquitectónico Modelo-Vista-Controlador, para promover la separación de responsabilidades, la cohesión dentro de cada capa y un bajo acoplamiento entre ellas. Las capas principales son:

1. **Capa de Presentación (UI):** Implementada con Java Swing, es responsable de toda la interacción con el usuario. Contiene las ventanas, diálogos y controles gráficos. Delega las acciones del usuario al **Controlador** y muestra los datos recibidos de este.
2. **Capa de Controlador (Aplicación):** El Controlador (incorporando un patrón Singleton) actúa como intermediario entre la UI y las capas inferiores. Orquesta las operaciones, maneja el flujo de la aplicación, gestiona la sesión del usuario actual y traduce las acciones del usuario en operaciones del dominio o de persistencia.
3. **Capa de Dominio (Lógica de Negocio):** Contiene las clases que modelan los conceptos fundamentales de AppChat (Usuario, Contacto, Mensaje, etc.) y la lógica de negocio asociada. RepositorioUsuarios actúa como una caché en memoria para los usuarios. Esta capa es independiente de la UI y de los detalles de persistencia.
4. **Capa de Acceso a Datos (Persistencia):** Abstrae el almacenamiento y recuperación de datos mediante el patrón DAO. Se utiliza una FactoriaDAO para obtener instancias de los DAOs específicos (UsuarioDAO, MensajeDAO, etc.), que interactúan con el servicio de persistencia subyacente.

Esta arquitectura facilita la mantenibilidad, ya que los cambios en una capa tienen un impacto limitado en las otras, y la testabilidad, permitiendo probar cada capa de forma más aislada.

### 3.2. Decisiones de Diseño Relevantes

- **Modelo de Dominio Simplificado:** Siguiendo las directrices, se adoptó un modelo de dominio donde Contacto es la clase base abstracta para ContactoIndividual y Grupo. Esto permite un manejo polimórfico de los contactos del usuario.
- **Grupos como Listas de Difusión:** Un Grupo no es un chat grupal con historial compartido entre todos los miembros, sino una herramienta para enviar un mensaje a múltiples ContactoIndividuales simultáneamente. Cada mensaje enviado a un grupo se replica en el historial individual de cada miembro. Un grupo no podrá quedarse sin ContactosIndividuales asociados.
- **AgregableGrupos:** Se utiliza esta interfaz marcadora para definir explícitamente qué tipos de contactos pueden ser miembros de un grupo, mejorando la seguridad de tipos y dejando la puerta abierta a una futura ampliación de nuevos integrantes de grupos o características de estos.

- **Inmutabilidad Parcial:** Campos como `Mensaje.fecha` o `Usuario.fechaRegistro` son final. Para las colecciones devueltas por getters en las clases de dominio, se ha optado por devolver nuevas instancias de listas (ej. `new LinkedList<...>`), proporcionando una copia defensiva que, si bien permite la modificación de la copia, protege la colección interna original del objeto.
- **Singletons para Componentes Centrales:** `Controlador`, `RepositorioUsuarios` y `FactoriaDescuentos` son Singletons (implementados con `enum`), asegurando un único punto de acceso global y una gestión controlada de su estado. También lo son las ventanas de la *UI*, para controlar la gestión de estas cuando se intenta abrir más de una misma venta a la vez.
- **RepositorioUsuarios como Caché:** Este repositorio carga todos los usuarios al inicio de la aplicación, sirviendo como una caché rápida en memoria. Las operaciones de modificación (añadir/eliminar usuario) en el repositorio actual solo afectan esta caché; la persistencia de estos cambios es coordinada por el `Controlador` a través de los DAOs.
- **Desacoplamiento de Descuentos:** El uso del patrón `Strategy` y una `Factoría` para los descuentos permite una gran flexibilidad para añadir o modificar reglas de promoción sin impactar el resto del sistema. Decidimos eliminar el atributo `Descuento` del usuario, de modo que esta clase para actuar como un servicio para el `Controlador` para conseguir la aplicación del mejor descuento automáticamente para cada `Usuario`.
- **Manejo de Contraseñas:** Por simplicidad y siguiendo las indicaciones para esta fase del proyecto, las contraseñas se manejan en texto plano. Se reconoce que esto no es una práctica segura para producción y se ha documentado en el código.
- **Flujo de Mensajes:** El `Controlador` gestiona el envío de mensajes: registra el mensaje enviado, lo añade al historial del contacto del emisor (a través de `Usuario.enviarMensaje`), y luego se encarga de la lógica para que el receptor (o receptores, en caso de un grupo) también “reciba” el mensaje, actualizando sus respectivos historiales y persistiendo los cambios.
- **Mensajes de números no registrados como contactos:** Se decidió realizar la ampliación para manejar el recibo de mensajes por parte de contactos que no tenemos añadidos, añadiendo a estos de forma automática la lista de contactos del usuario receptor, estableciendo como nombre el número de teléfono del emisor. Este nombre puede ser editado por el receptor.
- **DTO para Contextualizar Mensajes Buscados:** Con el objetivo de mejorar la separación de responsabilidades y optimizar la presentación de los resultados en la funcionalidad de búsqueda de mensajes, se introdujo el `Data Transfer Object (DTO) MensajeContextualizado`. Este objeto es construido por la clase `Usuario` (capa de dominio) durante el proceso de búsqueda. Su función principal es encapsular cada objeto `Mensaje` encontrado junto con información contextual necesaria para la interfaz gráfica, como el nombre del interlocutor directo (ya sea el contacto o el propio usuario) y el nombre del usuario actual. Esta decisión de diseño permite que la capa de vista (específicamente, la clase `Buscador`) reciba una estructura de datos completa y lista para ser presentada, sin necesidad de realizar consultas adicionales o lógica compleja para inferir el contexto de cada mensaje.

## 4. Patrones de diseño utilizados

En el desarrollo de AppChat, se han aplicado conscientemente varios patrones de diseño para abordar problemas comunes y mejorar la calidad general del software. A continuación, se detallan los más significativos:

### 1. Strategy Pattern:

- **Contexto:** Sistema de cálculo de precios para la suscripción Premium, donde pueden existir diversas políticas de descuento.
- **Implementación:**
  - dominio.Descuento: Interfaz que define el contrato (getPrecio, isAplicable).
  - DescuentoPorFecha, DescuentoPorMensaje, DescuentoSinDescuento: Clases concretas que implementan diferentes algoritmos de descuento.
  - dominio.FactoriaDescuentos: Actúa como el contexto que selecciona la estrategia adecuada.
- **Ventajas:** Permite intercambiar o añadir nuevas políticas de descuento dinámicamente sin modificar el código cliente que las utiliza, promoviendo la extensibilidad.

### 2. Factory (Método de Fábrica implícito en FactoriaDescuentos):

- **Contexto:** Creación/selección centralizada de objetos Descuento.
- **Implementación:** El método FactoriaDescuentos.getMejorDescuento() encapsula la lógica para decidir qué instancia de Descuento es la más apropiada para un usuario y precio dados.
- **Ventajas:** Desacopla al Controlador de las clases concretas de Descuento y de la lógica de selección, simplificando su código y centralizando la creación de descuentos.

### 3. Singleton Pattern:

- **Contexto:** Necesidad de una única instancia global para ciertos componentes críticos.
- **Implementación:**
  - controlador.Controlador: Gestiona el flujo principal de la aplicación.
  - dominio.RepositorioUsuarios: Mantiene la caché de usuarios.
  - dominio.FactoriaDescuentos: Proporciona acceso a las estrategias de descuento.
  - Todas las ventanas de la GUI: Garantiza que no se puedan abrir, simultáneamente, dos ventanas iguales.
  - Todas implementadas utilizando enum de Java, que es la forma recomendada y más segura.
- **Ventajas:** Proporciona un punto de acceso global y controlado, previene la creación múltiple de objetos que deben ser únicos, y simplifica la gestión de su estado.

### 4. Builder Pattern:

- **Contexto:** Construcción de objetos Usuario, que tienen múltiples atributos, algunos obligatorios y otros opcionales.
- **Implementación:** dominio.Usuario.Builder como una clase interna estática.
- **Ventajas:** Ofrece una API fluida y legible para crear instancias de Usuario, mejora la claridad en comparación con constructores con muchos parámetros (telescopicos), y facilita el manejo de atributos opcionales con valores por defecto.

### 5. Marker Interface Pattern:

- **Contexto:** Necesidad de categorizar ciertos tipos de Contacto como elegibles para ser miembros de un Grupo.
- **Implementación:** La interfaz dominio.AgregableGrupos no declara métodos; ContactoIndividual la implementa.

- **Ventajas:** Permite una forma de categorización y restricción de tipos en tiempo de compilación (a través del tipado de parámetros) o en tiempo de ejecución (usando instanceof), sin imponer una jerarquía de clases más compleja que la necesaria.
6. **Data Access Object (DAO) Pattern:**
- **Contexto:** Abstraer el acceso a la fuente de datos persistente.
  - **Implementación:** El Controlador y RepositorioUsuarios utilizan una FactoriaDAO para obtener instancias de interfaces DAO (ej. UsuarioDAO, MensajeDAO). Estas interfaces definen las operaciones de persistencia, y sus implementaciones concretas manejan la interacción con el servicio de persistencia.
  - **Ventajas:** Desacopla la lógica de negocio de los detalles específicos del almacenamiento de datos, permitiendo cambiar la tecnología de persistencia con un impacto mínimo en el resto de la aplicación.
7. **Template Method Pattern (Implícito):**
- **Contexto:** Definir un esqueleto de algoritmo en una superclase, permitiendo a las subclasses redefinir ciertos pasos.
  - **Implementación:** La clase abstracta Contacto define métodos como getURLImagen() (abstracto) que las subclasses ContactoIndividual y Grupo deben implementar. El método addMensaje en Contacto es invocado por super.addMensaje() en Grupo, que luego añade su comportamiento específico.
  - **Ventajas:** Promueve la reutilización de código y permite a las subclasses especializar partes de un algoritmo sin cambiar su estructura general.
8. **Composite Pattern (controlado por interfaz marcadora):**
- **Contexto:** Manejar una jerarquía de objetos donde tanto los objetos individuales (hojas) como las agrupaciones de estos objetos (compuestos) necesitan ser tratados de manera uniforme por el cliente en ciertos aspectos.
  - **Implementación:** La estructura de 'Contacto' y sus subclasses 'ContactoIndividual' (actuando como hoja) y 'Grupo' (actuando como compuesto) permiten que un grupo esté formado por elementos 'Contacto', eso sí, solo aquellos que implementen la interfaz AgregableGrupo.
  - **Ventajas:** Permite tratar a los objetos individuales y composiciones de objetos de manera uniforme a través de la interfaz común 'Contacto', evitando la necesidad de distinguir constantemente entre 'ContactoIndividual' y 'Grupo'.

## Patrones de Diseño Implícitos en Java Swing Utilizados por AppChat

AppChat se beneficia de patrones de diseño robustos inherentes al framework Java Swing, sobre el cual está construida su interfaz gráfica:

- **Model-View-Controller (MVC) y sus Variantes:** Swing separa el modelo de datos (ej. ListModel) de la vista (ej. JList) y el control de eventos. AppChat adopta esta separación en su arquitectura general.
- **Observer (Observador):** Fundamental en el manejo de eventos de Swing. Componentes como JButton notifican a los ActionListener registrados sobre interacciones del usuario, un mecanismo que AppChat usa en toda su GUI.
- **Decorator (Decorador):** Componentes como JScrollPane añaden funcionalidad (desplazamiento) a otros componentes (ej. listas o áreas de texto en AppChat) sin alterar su comportamiento base.

- **Composite (Compuesto):** La jerarquía Component/Container de Swing permite tratar contenedores (como JPanel) y componentes individuales de manera uniforme, facilitando la construcción de interfaces complejas como las de AppChat.
- **Strategy (Estrategia):**
  - Los LayoutManager (ej. BorderLayout, FlowLayout) definen cómo se organizan los componentes en un contenedor, permitiendo cambiar la estrategia de disposición fácilmente.
  - Los Renderers (ej. ListCellRenderer) personalizan cómo se muestran los datos en componentes como JList, estrategia usada en AppChat para mostrar contactos.
- **Command (Comando):** La interfaz Action de Swing encapsula una acción y sus propiedades (texto, icono), permitiendo que la misma acción sea invocada desde diferentes elementos UI (ej. un botón y un ítem de menú).
- **Adapter (Adaptador):** Clases como MouseAdapter o WindowAdapter simplifican la creación de listeners de eventos al permitir implementar solo los métodos de interés de una interfaz EventListener.
- **Factory Method (Método de Fábrica) / Abstract Factory (Fábrica Abstracta):** Swing utiliza fábricas internamente, como UIManager para gestionar el Look and Feel, o BorderFactory para crear bordes, desacoplando la creación de objetos de su uso.

La aplicación de estos patrones ha sido fundamental para lograr una base de código organizada, flexible y alineada con las buenas prácticas de diseño de software.

## 5. Manual de usuario

Este manual de usuario proporciona una guía detallada sobre cómo utilizar la aplicación AppChat, desde el registro inicial hasta las funcionalidades más avanzadas como la gestión de contactos, grupos y la función premium.

### 5.1. Registro de un Nuevo Usuario

Para comenzar a utilizar AppChat, primero debe registrarse como nuevo usuario. Al abrir la aplicación, se encontrará con la siguiente ventana de inicio de sesión:

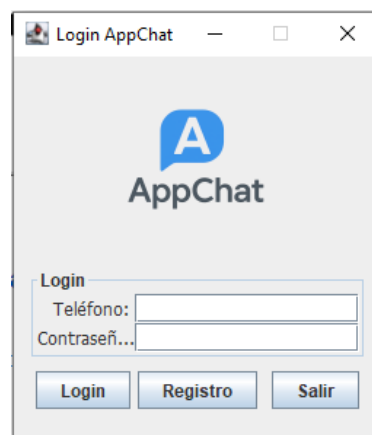


Figura 1: Login AppChat

Para proceder con el registro, haga clic en el botón "Registro". Esto abrirá la ventana de registro de usuario.

A screenshot of a web browser window titled "Registro Usuario". The window has a light gray background. Below the title bar is a section titled "Datos de Registro". This section contains several input fields: "Nombre:" with the value "Gregorio", "Apellidos:" with the value "Quesadilla Morrón", "Email:" with the value "gqm@gmail.com", "Teléfono:" with the value "626698425", "Password:" with a masked value "...", "Otra vez:" with a masked value "...", "Fecha de Nacimiento:" with the value "11/05/1978" and a calendar icon, "Saludo:" with the value "¡Hola soy Gregorio!", and "Imagen de ..." with a URL. At the bottom of the form are two buttons: "Registrar" and "Cancelar".

Figura 2: Registro AppChat

En esta pantalla, deberá completar los campos requeridos: Nombre, Apellidos, Email, Teléfono, Contraseña, confirmación de Contraseña, Fecha de Nacimiento, un Saludo (opcional) y la ruta a una Imagen de perfil (opcional). Una vez que haya rellenado todos los datos, haga clic en el botón "Registrar".

## 5.2. Inicio de Sesión

Una vez que su cuenta ha sido registrada exitosamente, puede iniciar sesión en la aplicación. Regrese a la ventana principal de inicio de sesión. Introduzca el número de teléfono y la contraseña con los que se registró en los campos correspondientes y haga clic en el botón "Login".

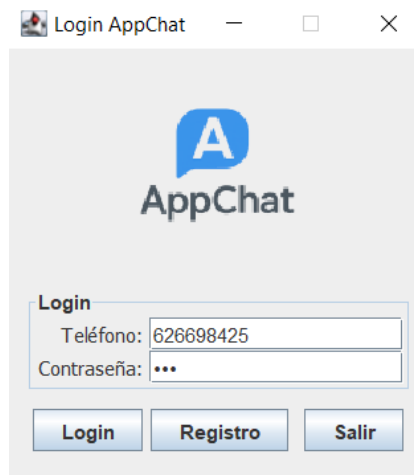


Figura 3: Login AppChat (2)

## 5.3. Ventana Principal y Navegación

Tras iniciar sesión, accederá a la ventana principal de la aplicación.

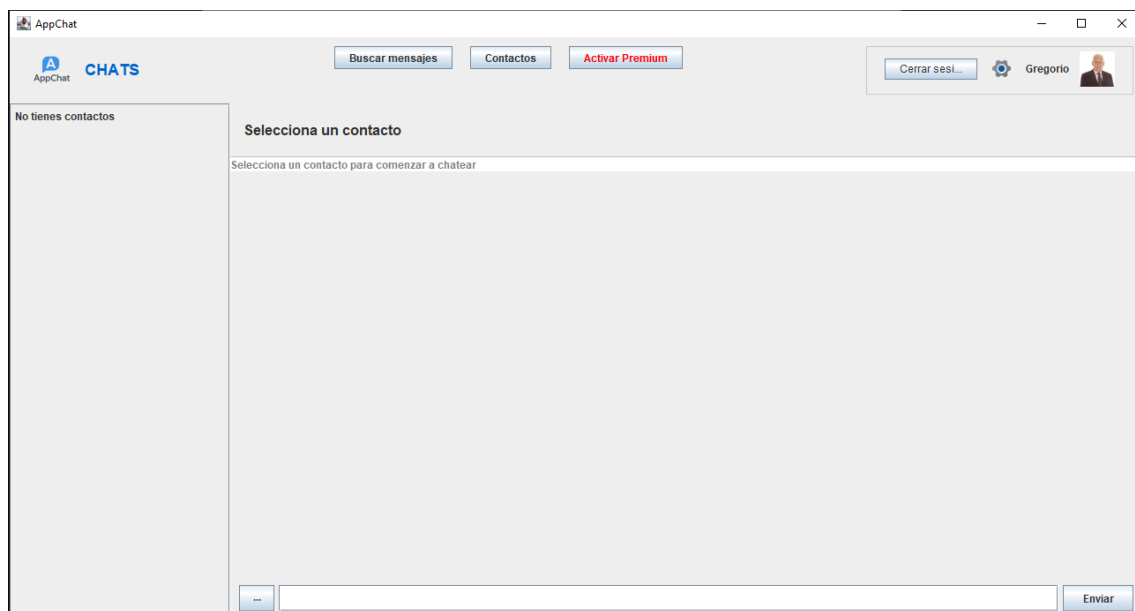


Figura 4: Ventana Principal AppChat

Esta es la interfaz central desde la que podrá acceder a las diferentes funcionalidades de AppChat. En el panel izquierdo, verá la lista de sus chats activos. El área central mostrará el contenido de la conversación seleccionada.

En la parte superior de la ventana principal, encontrará varios botones de navegación: "Buscar mensajes", "Contactos" y "Activar Premium". Junto a esta sección, a la derecha, se mostrará un panel para el usuario.

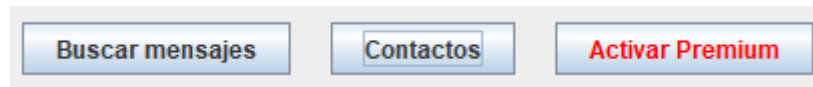


Figura 5: Botones de navegación AppChat

## 5.4. Gestión de Perfil

En la esquina superior derecha de la ventana principal, encontrará la sección de usuario con su nombre y una imagen de perfil.

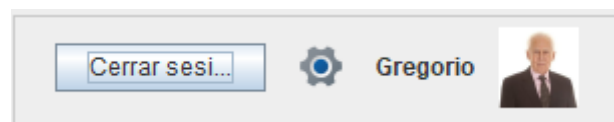


Figura 6: Panel usuario AppChat

Al interactuar con esta sección, tiene acceso a tres opciones principales:

- **Cerrar sesión:** Para salir de su cuenta.
- **Configuración del perfil:** Permite modificar sus datos de usuario.
- **Información del perfil:** Al pulsar sobre su imagen de perfil, se mostrará una ventana con la información de su perfil.



Figura 7: Información usuario AppChat

Desde la opción de "Configuración del perfil", podrá actualizar su nombre, apellidos, email, número de teléfono, cambiar su contraseña, modificar su fecha de nacimiento, actualizar su saludo y cambiar su imagen de perfil.



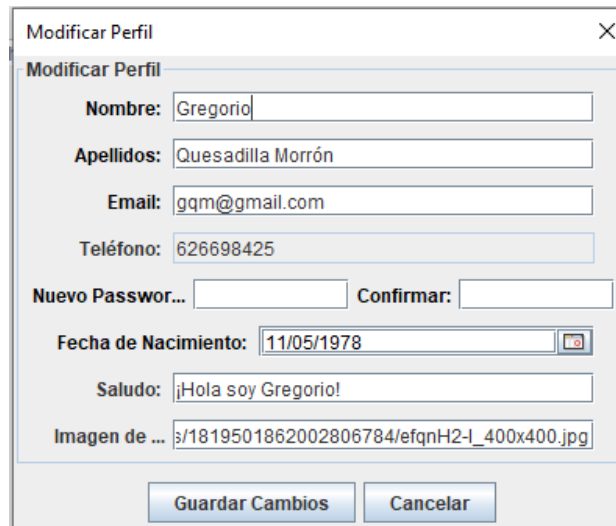


Figura 8: Modificar usuario AppChat

Después de realizar los cambios deseados, haga clic en "Guardar Cambios".

## 5.5. Gestión de Contactos y Grupos

Para gestionar sus contactos y crear grupos, haga clic en el botón "Contactos" en la ventana principal. Esto abrirá la ventana de gestión de contactos y grupos.

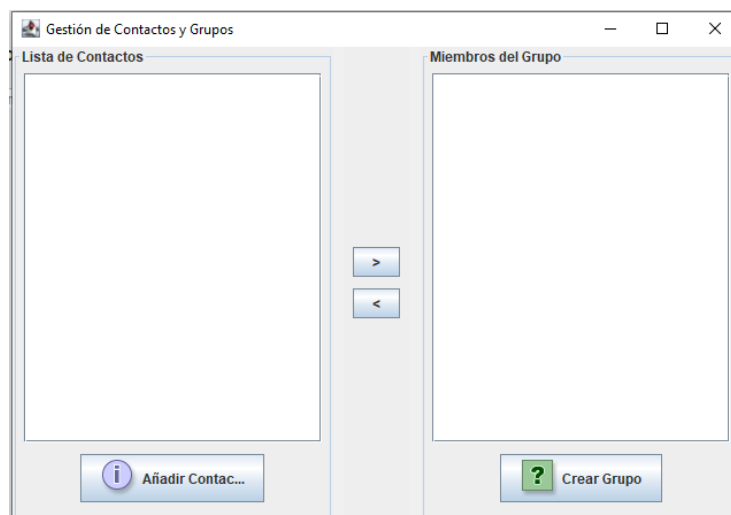
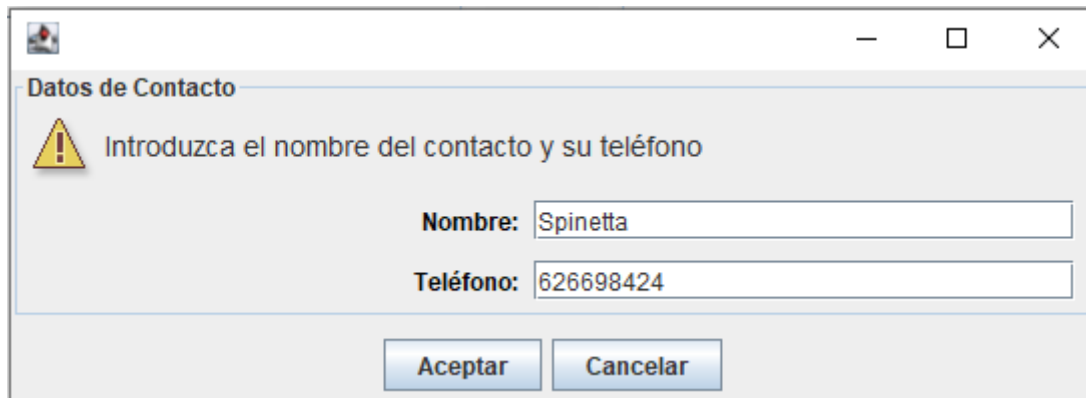


Figura 9: Ventana contactos AppChat

En esta ventana, verá su "Lista de Contactos" en el panel izquierdo y los "Miembros del Grupo" que está creando o editando en el panel derecho.

### Añadir un Contacto:

Pulse el botón "Añadir Contacto...". Se abrirá una ventana donde deberá introducir el nombre y el número de teléfono del contacto que desea agregar.



Datos de Contacto

Introduzca el nombre del contacto y su teléfono

Nombre: Spinetta

Teléfono: 626698424

Aceptar Cancelar

Figura 10: Crear contacto AppChat

**Importante:** Debe existir un usuario registrado en AppChat con el número de teléfono que introduzca. Haga clic en "Aceptar" para añadir el contacto. El nuevo contacto aparecerá en su "Lista de Contactos".

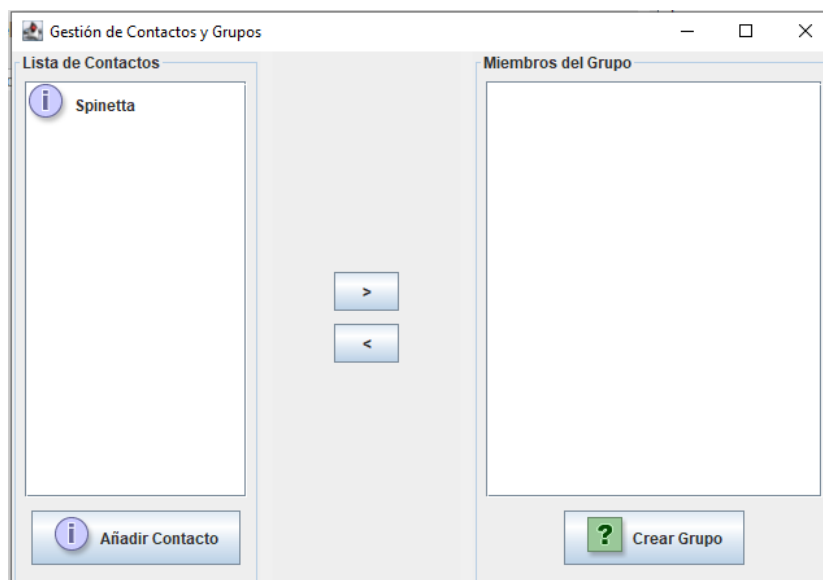


Figura 11: Contacto creado

Además, el contacto también se añadirá a su panel de chats en la ventana principal.

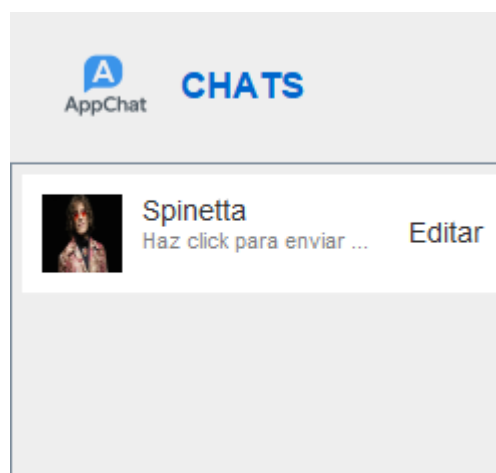


Figura 12: Contacto aparece en Ventana Principal

### Editar un Contacto:

Para editar un contacto existente, selecciónelo en la "Lista de Contactos" y luego pulse el botón "Editar" que aparece junto a su nombre en el panel de chats principal. Se abrirá una ventana que le permitirá modificar el nombre del contacto.

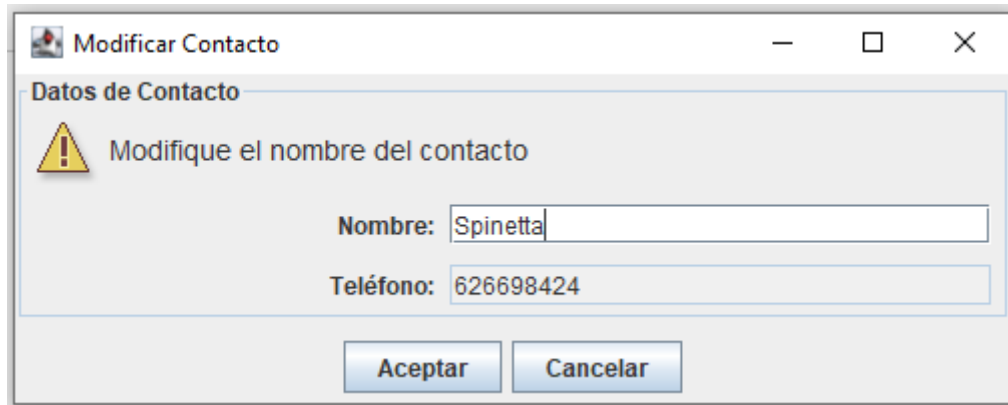


Figura 13: Modificar Contacto AppChat

### Crear un Grupo:

En la ventana de gestión de contactos y grupos, seleccione los contactos de su "Lista de Contactos" que desea añadir al grupo y pulse el botón ">".

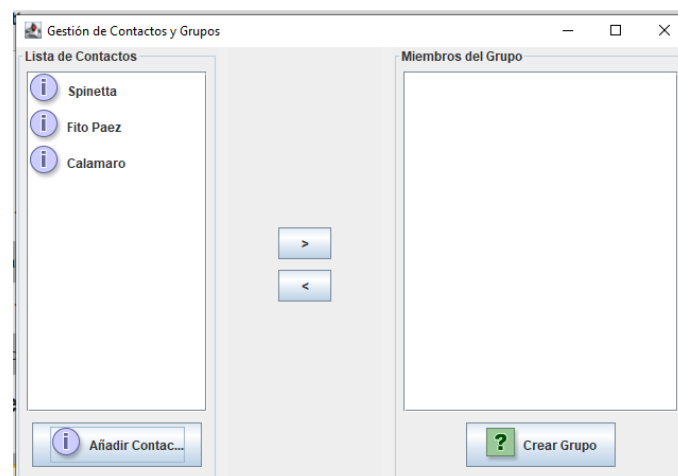


Figura 14: Crear Grupo AppChat (1)

Los contactos seleccionados se moverán al panel "Miembros del Grupo".

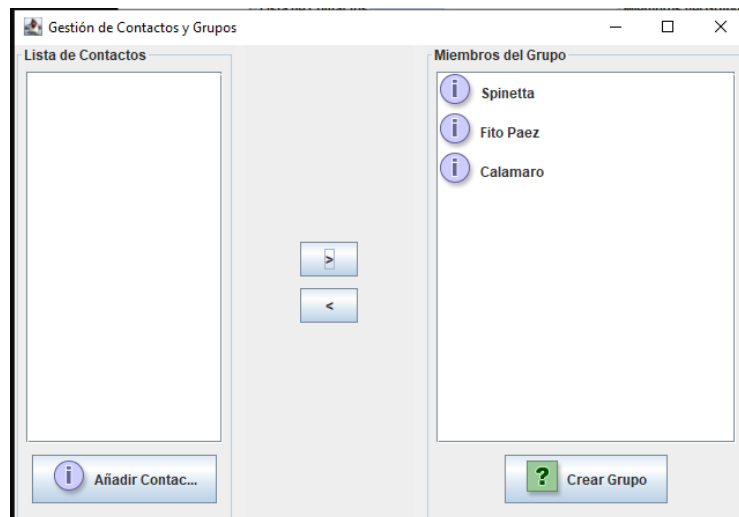


Figura 15: Crear Grupo AppChat (2)

Una vez que haya añadido a todos los miembros deseados, pulse el botón "Crear Grupo". Se le pedirá que introduzca un nombre para el grupo y luego el grupo será creado.

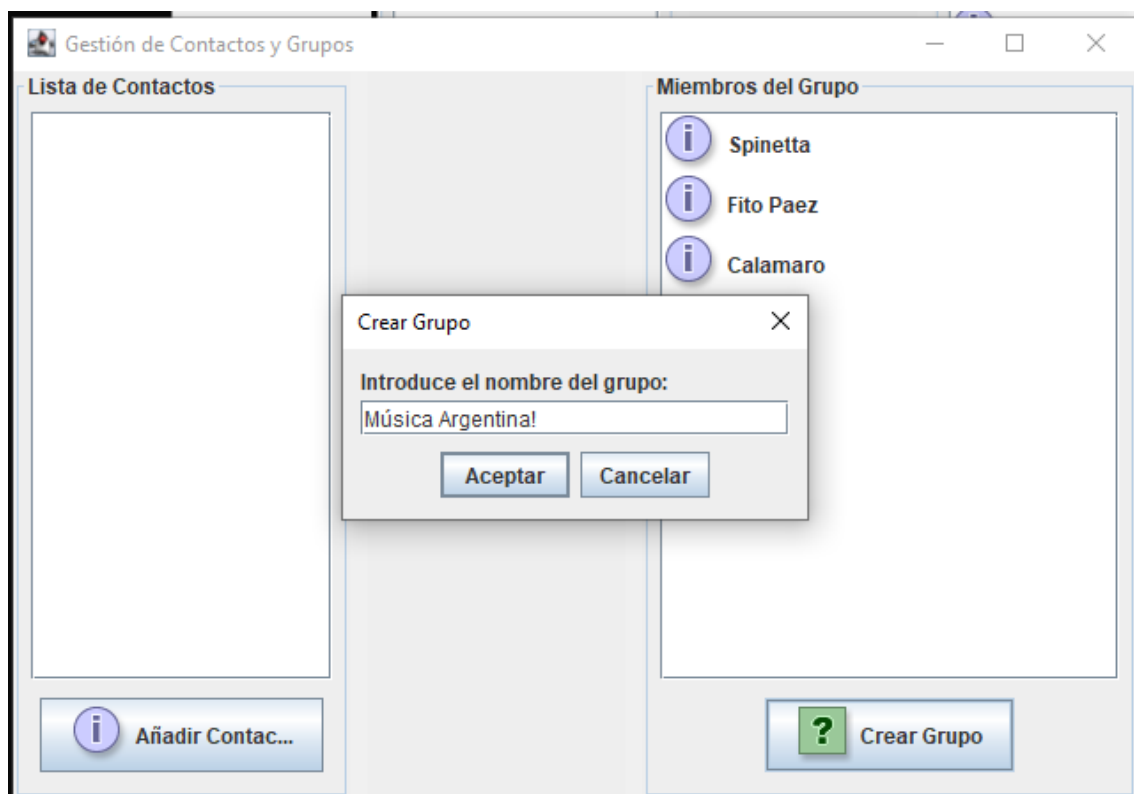


Figura 16: Crear Grupo AppChat (3)

El nuevo grupo aparecerá en su lista de chats en la ventana principal, junto con sus contactos individuales.

### Editar un Grupo:

De igual manera que con un contacto individual, se puede modificar un grupo. En este caso, al accionar "Editar" se nos mostrará la siguiente pantalla:

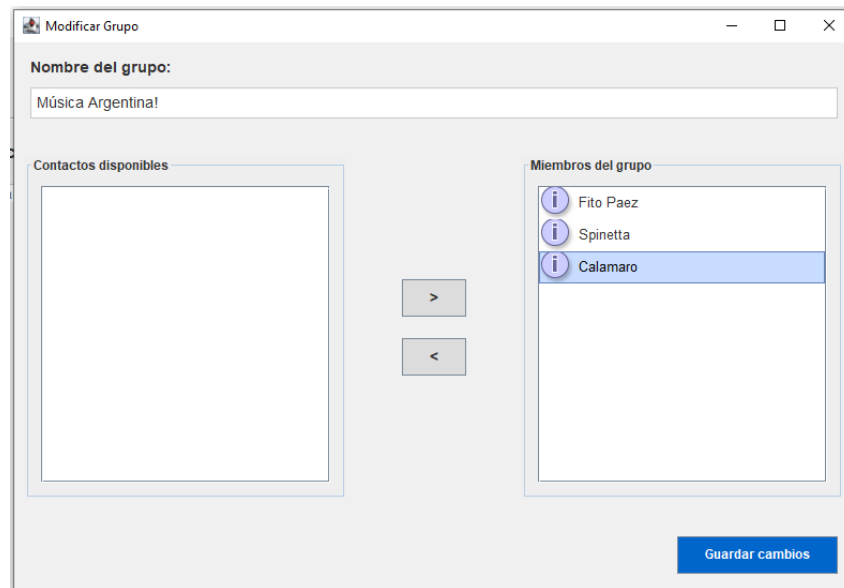


Figura 17: Modificación de Grupo

Podrá eliminar o añadir miembros al grupo de igual manera que al crearlos. También se puede modificar el nombre del grupo. Finalmente, los cambios se hayan hecho, se pulsa “Guardar Cambios” y el grupo se habrá modificado correctamente.

## 5.6. Envío de Mensajes

Para enviar un mensaje a un contacto o grupo, selecciónelo en su lista de chats en la ventana principal. El chat relacionado se mostrará en el área central.

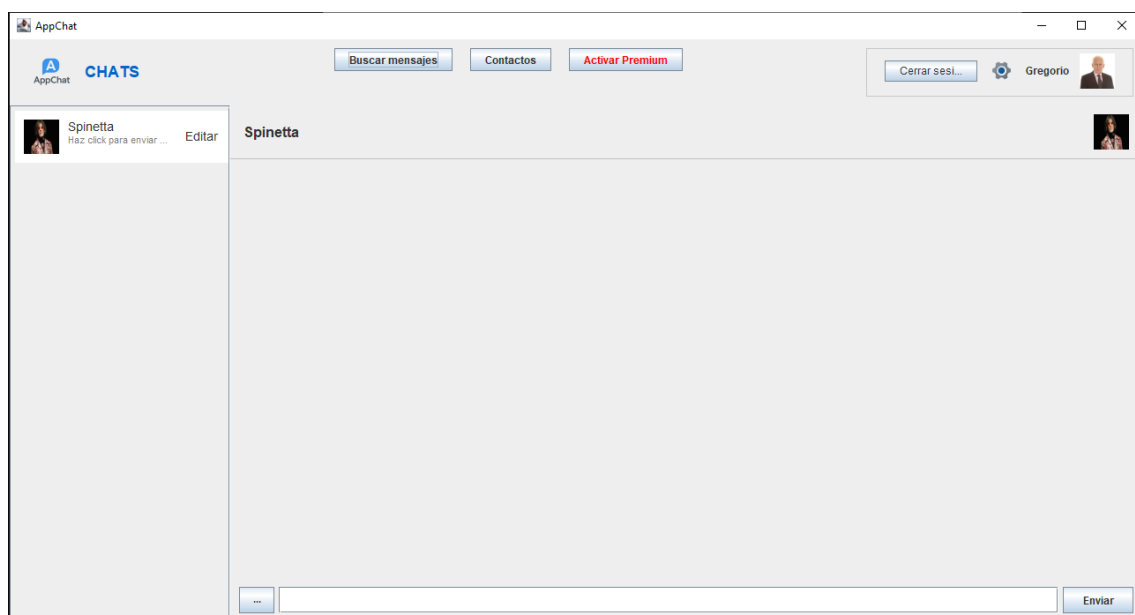


Figura 18: Chat Ventana Principal

Escriba su mensaje en la caja de texto situada en la parte inferior de la ventana y pulse la tecla Enter o haga clic en el botón "Enviar". El mensaje será enviado.

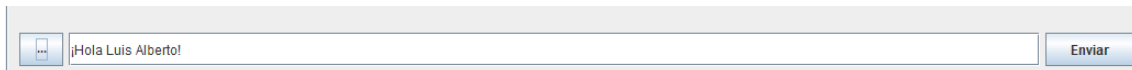


Figura 19: Mensaje en caja de texto

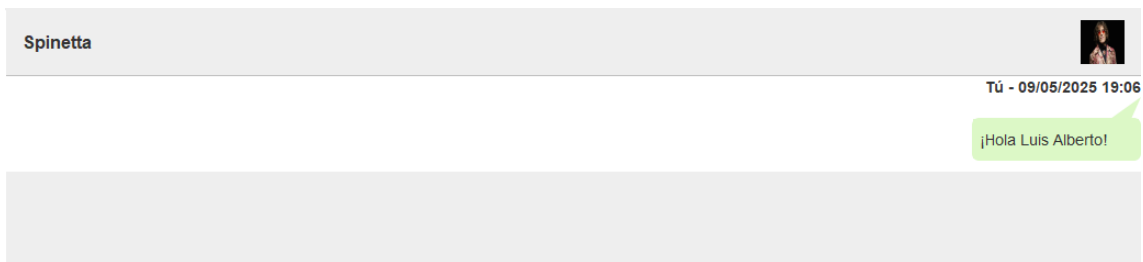


Figura 20: Mensaje enviado

Si desea incluir emoticonos en su mensaje, pulse el botón a la izquierda de la caja de texto del mensaje para ver los emoticonos disponibles.

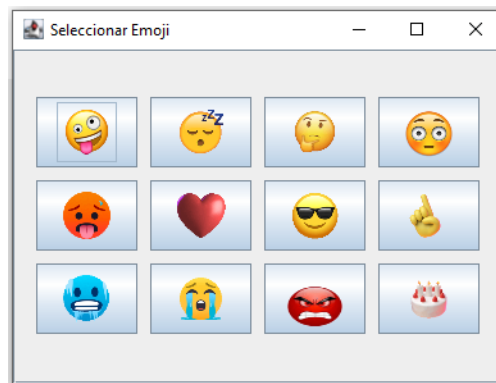


Figura 21: Selección de emoticonos

Seleccione el emoticono deseado y se insertará en su mensaje.

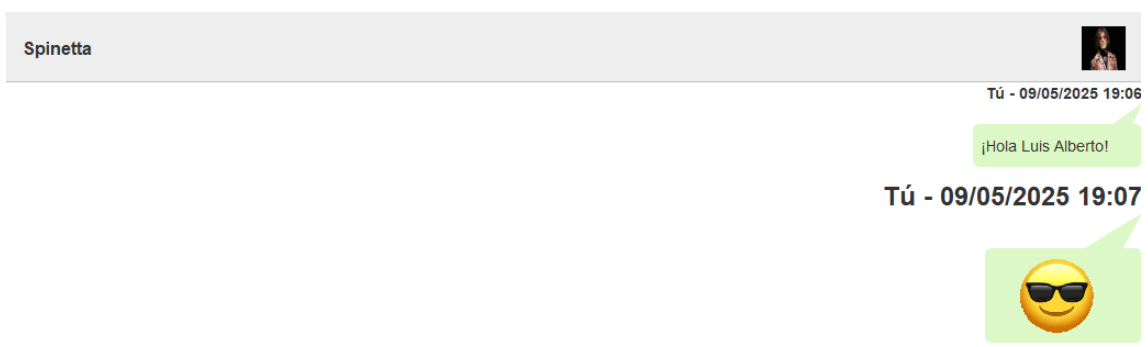


Figura 22: Emoticono enviado

Cuando envía un mensaje a un grupo, este mensaje se envía a cada contacto individual que forma parte de ese grupo.

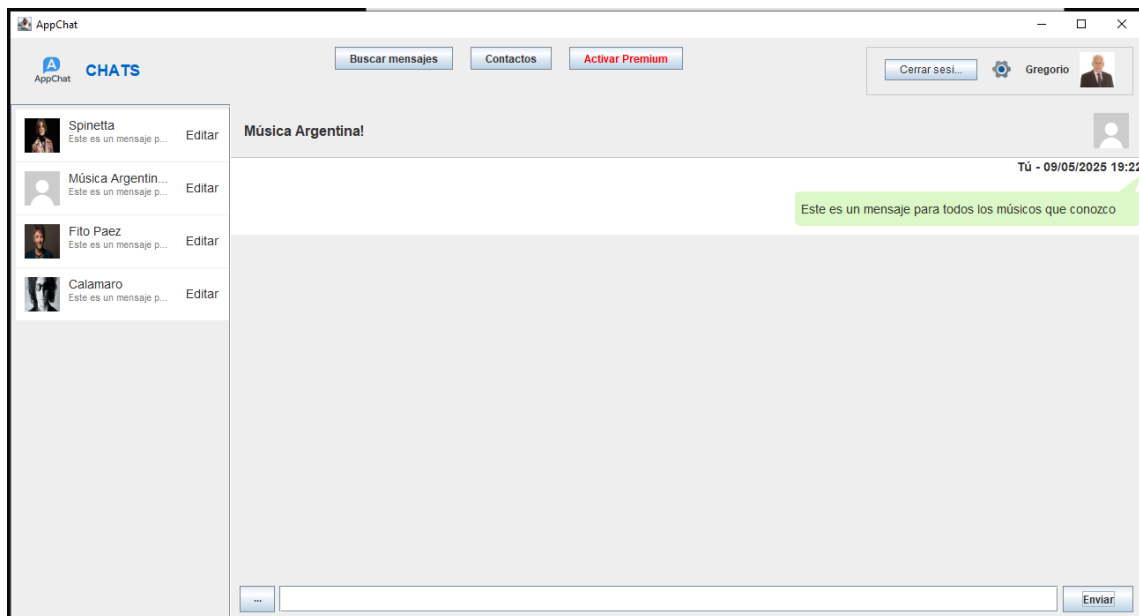


Figura 23: Envío de Mensaje en Grupo

## 5.7. Búsqueda de Mensajes

Para buscar mensajes específicos, haga clic en el botón "Buscar mensajes" en la parte superior de la ventana principal.

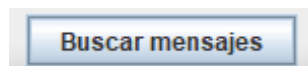


Figura 24: Botón de buscar mensajes

Esto abrirá la ventana de búsqueda de mensajes.

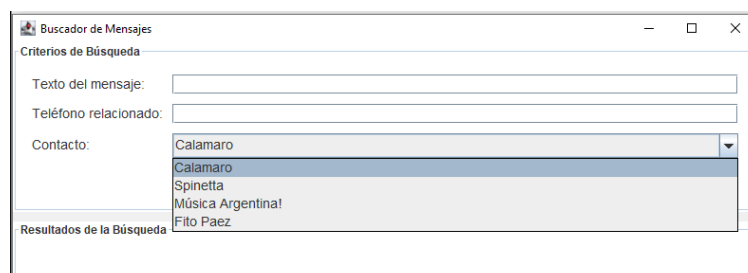


Figura 25: Ventana Buscar Mensajes

Puede buscar mensajes por texto, número de teléfono o contacto. Si selecciona un contacto y luego introduce texto en el campo de búsqueda, la aplicación solo mostrará los mensajes que haya intercambiado con ese contacto y que contengan el texto especificado.

The screenshot shows a window titled 'Buscador de Mensajes'. It has a section 'Criterios de Búsqueda' with three input fields: 'Texto del mensaje:', 'Teléfono relacionado:', and 'Contacto:'. The 'Contacto:' field has a dropdown menu with 'Spinetta' selected. Below these fields is a blue button labeled 'Buscar Mensajes'. The 'Resultados de la Búsqueda' section displays a list of messages from 'Spinetta' with timestamps and content: '2025-05-09T19:06: ¡Hola Luis Alberto!', '2025-05-09T19:07: Emoticono 6', '2025-05-09T19:09: Genial amigo!!', '2025-05-09T19:10: Todo piola :)', '2025-05-09T19:10: Emoticono 0', and '2025-05-09T19:22: Este es un mensaje para todos los músicos que conozco'.

Figura 26: Búsqueda de mensajes (1)

The screenshot shows the same 'Buscador de Mensajes' window. In this instance, the 'Teléfono relacionado:' field contains the number '666777666'. The 'Contacto:' dropdown still shows 'Spinetta'. The 'Buscar Mensajes' button is present. The 'Resultados de la Búsqueda' section now only displays one message: 'Calamaro 2025-05-09T19:22: Este es un mensaje para todos los músicos que conozco'.

Figura 27: Búsqueda de mensajes (2)

## 5.8. Funcionalidad Premium

AppChat ofrece una funcionalidad premium con características adicionales. Para activar el modo premium, haga clic en el botón "Activar Premium" en la ventana principal.

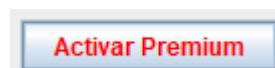


Figura 28: Botón Activar Premium

Se le presentará información sobre la funcionalidad premium, incluyendo el precio con el descuento más favorable aplicado, y un botón para convertirse en usuario premium.



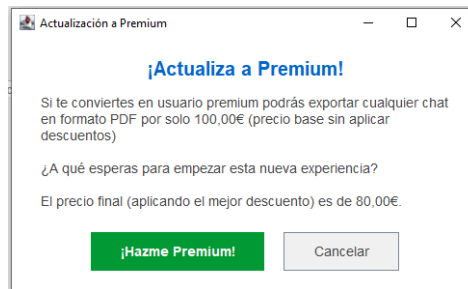


Figura 29: Ventana Actualización Premium

Como usuario premium, tendrá acceso a la función de exportación de chats. Podrá acceder a esta función pulsando el botón que indicará "Premium Activo".



Figura 30: Botón Premium Activo

Desde la ventana de funcionalidad premium activa, podrá seleccionar un contacto o grupo, pulsar "Exportar", elegir una ruta en su explorador de archivos y exportar el chat correspondiente correctamente.

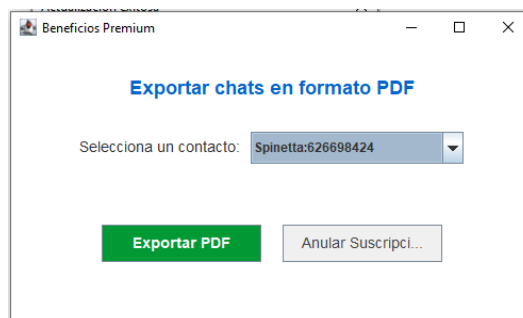


Figura 31: Ventana exportación PDF

## 6. Conclusión

El desarrollo del proyecto AppChat ha representado un ejercicio práctico integral, permitiendo aplicar los conceptos y tecnologías fundamentales de la ingeniería del software moderno. A lo largo de su implementación, se ha puesto énfasis en la creación de un dominio robusto y bien estructurado, la aplicación de patrones de diseño para resolver problemas comunes de forma elegante, y la consideración de una arquitectura en capas que promueva la modularidad y la separación de intereses.

La refactorización del código de dominio ha sido un paso importante para mejorar la claridad y mantenibilidad, prestando atención a la correcta encapsulación, el manejo de excepciones y el uso efectivo de las características del lenguaje Java, como los streams.

Se han identificado áreas de mejora continua como el desacoplamiento entre el dominio y las librerías específicas de la interfaz de usuario.

Consideramos que el proyecto cumple con los objetivos funcionales principales definidos en las historias de usuario y demuestra la aplicación de los principios de diseño y patrones estudiados. La experiencia ha sido valiosa para consolidar conocimientos en el diseño orientado a objetos, la arquitectura de software y las buenas prácticas de programación.

**Estimación del Tiempo Dedicado:** Aproximadamente, hemos dedicado unas 125 horas cada uno, lo que hace un total de 250 horas.