



---

GRADO EN INGENIERÍA INFORMÁTICA

3ER CURSO

**Tecnologías de Desarrollo Software**

AppChat

Pablo Ballesta Rodriguez - 1.2

Alvaro Miñarro Gil - 3.2

---



# Índice

<b>Introducción.....</b>	<b>3</b>
<b>1. User Story Mapping.....</b>	<b>4</b>
<b>2.1. Historias de Usuario.....</b>	<b>5</b>
<b>2.2. Modelo de dominio.....</b>	<b>7</b>
<b>3. Diagrama de colaboración.....</b>	<b>8</b>
<b>4. Diseño.....</b>	<b>9</b>
4.1 Arquitectura.....	9
4.2 Decisiones de diseño relevantes.....	10
4.2.1 Gestión de los mensajes.....	10
4.2.1 Uso de interfaces.....	11
4.3 Controlador Stub.....	12
<b>5. Patrones.....</b>	<b>13</b>
5.1 Patrones de Creación.....	13
5.2 Patrones de Estructural.....	13
5.3 Patrones de Comportamiento.....	13
<b>6. Manual de usuario.....</b>	<b>14</b>
6.1 Inicio de sesión y registro.....	14
6.2 Gestión de contactos.....	15
6.3 Gestión de grupos.....	16
6.4 Mensajería.....	19
6.5 Funciones premium.....	21
6.6 Perfil de usuario.....	22
<b>7. Conclusiones.....</b>	<b>23</b>



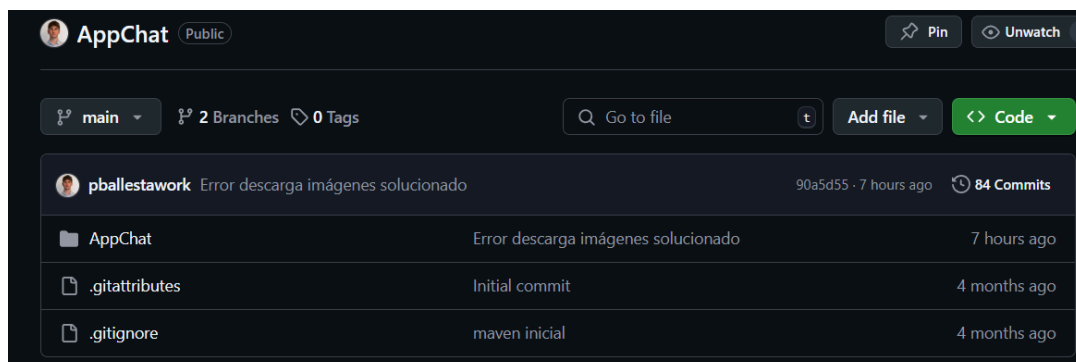
# Introducción

La aplicación AppChat es una aplicación desarrollada en JAVA con fines educativos. La intención es llegar a simular una aplicación real de intercambio de mensajes al estilo 'WhatsApp', utilizando siempre una arquitectura limpia y patrones de diseño útiles para tener un código desacoplado.

Buscaremos siempre utilizar el principio de inversión de dependencias y aplicaremos una programación orientada a interfaces para cumplir el objetivo inicial. Este tipo de programación nos ayudará además en el reparto de responsabilidades, ya que al abstraer la interfaz de la funcionalidad, podemos desarrollar un mismo caso de uso en distintas capas a la vez.

Para el desarrollo de la vista se usará Swing, una framework para construir interfaces gráficas con una gran variedad de opciones. En ocasiones se ha utilizado la herramienta 'Window Builder' para el prototipo inicial y se ha evolucionado a partir de esta versión.

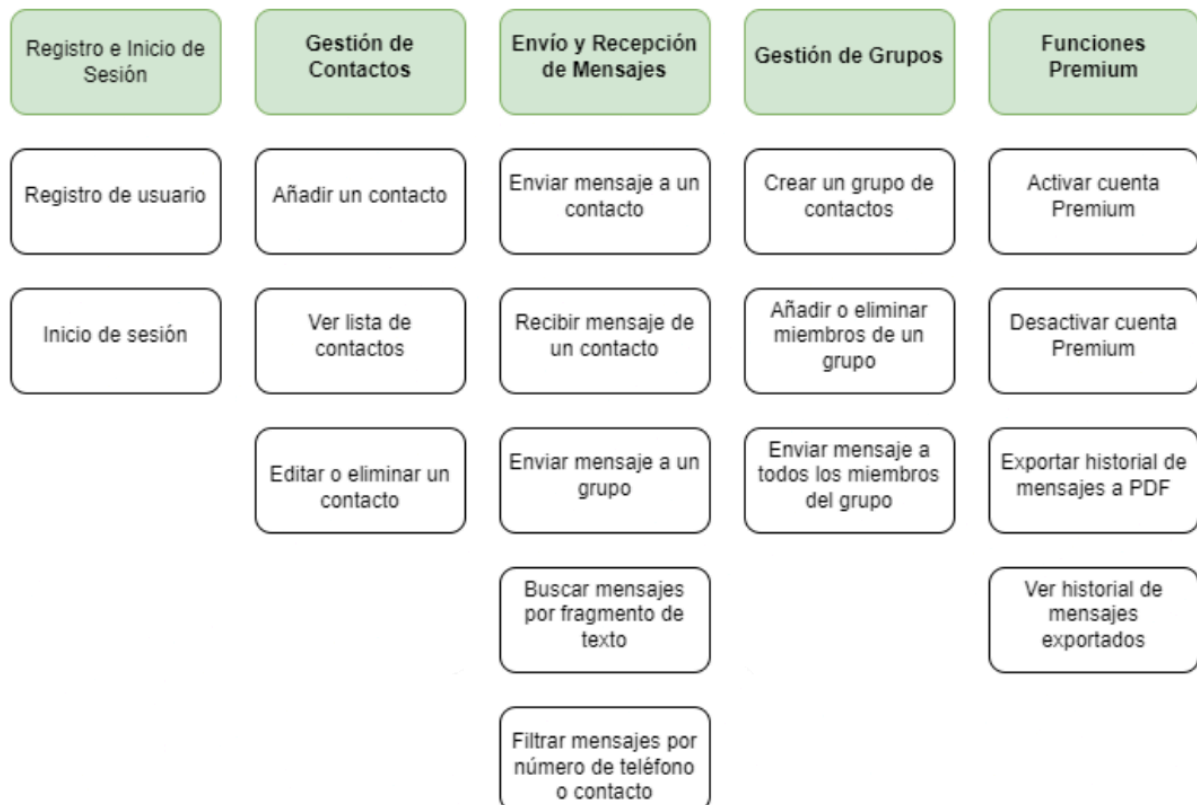
Hemos utilizado la plataforma GitHub para llevar un registro del trabajo realizado y trabajar en equipos remotos. Pincha [aquí](#) para acceder al repositorio.





# 1. User Story Mapping

En el entorno de las metodologías ágiles encontramos el mapa de historias de usuario. Este mapa nos ayuda a tener una visión clara y directa de los casos de uso que contendrá nuestra solución final. También nos ha ayudado a poder repartirnos las tareas con mayor profesionalidad.





## 2.1. Historias de Usuario

Tras la recogida de requisitos con el cliente o 'Enunciado de la práctica', se agruparon todos en distintas historias de usuario que albergan una funcionalidad general.

### H1. Registro de Usuario

**Como** nuevo usuario, **quiero** registrarme con mi información personal, **para** poder utilizar la aplicación.

- El usuario debe proporcionar un nombre completo, número de teléfono, contraseña, una fecha de nacimiento, una imagen de perfil, y opcionalmente un saludo.
- La contraseña debe repetirse para verificar que coincida.
- Si falta información obligatoria, se debe mostrar un mensaje de error.
- El número de teléfono debe ser único en el sistema.

### H2. Envío de Mensajes

**Como** usuario registrado, **quiero** enviar un mensaje a un contacto o a un grupo, **para** comunicarme con otros usuarios.

- El usuario debe poder seleccionar un contacto o grupo para enviar el mensaje.
- El mensaje debe incluir el texto, el número de teléfono del emisor, y la fecha y hora de envío.
- Si el receptor no tiene mi contacto agregado, debe mostrarse el nombre vacío.
- En el caso de enviar a un grupo, el mensaje debe enviarse individualmente a cada miembro.

### H3. Búsqueda de Mensajes

**Como** usuario, **quiero** poder buscar mensajes, **para** encontrar conversaciones anteriores basadas en texto, número de teléfono o nombre de contacto.

- El sistema debe permitir buscar por un fragmento de texto, número de teléfono o nombre del contacto.
- Los resultados deben mostrarse ordenados por fecha y hora.
- Se debe poder combinar los criterios de búsqueda (texto + teléfono, por ejemplo).



#### H4. Convertirse en Usuario Premium.

**Como** usuario, **quiero** convertirme en Premium, **para** acceder a funcionalidades adicionales como la exportación de mensajes.

- El usuario debe poder acceder a una opción de pago para convertirse en Premium.
- Al activar Premium, el usuario tendrá acceso a la exportación de mensajes en formato PDF.
- Deben aplicarse descuentos automáticamente si el usuario cumple con los requisitos.

#### H5. Exportación de Mensajes a PDF

**Como** usuario Premium, **quiero** exportar el historial de chat con un contacto a PDF, **para** tener un registro de las conversaciones.

- La exportación debe incluir los nombres de los contactos y sus números de teléfono.
- El archivo PDF debe contener todos los mensajes en orden cronológico.
- Debe generarse un archivo PDF accesible al usuario tras la exportación.

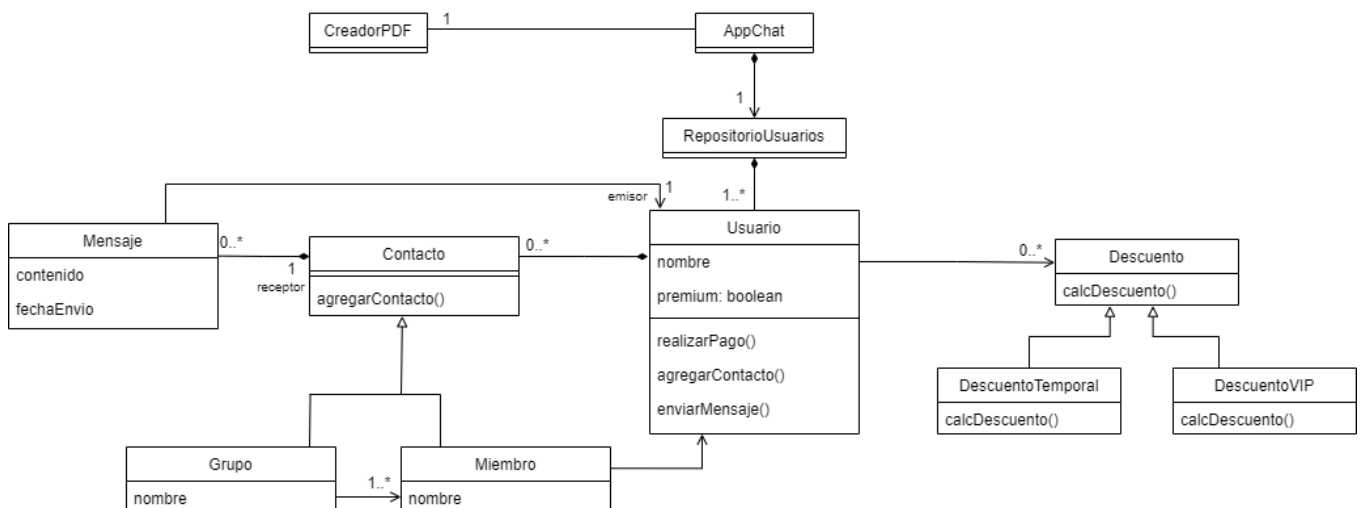


## 2.2. Modelo de dominio

La aplicación utiliza un controlador principal 'AppChat' que maneja los casos de uso mencionados. El controlador está relacionado con dos clases principalmente; RepositorioUsuarios y CreadorPDF. El controlador también conoce el usuario actual pero creemos que no tiene sentido relacionarlos en el modelo.

Un usuario contiene una lista de contactos generales. Los contactos pueden ser de varios tipos; ContactoIndividual y Grupo. El contacto individual se puede ver en el diagrama como 'Miembro' dando a entender que los grupos contienen contactos individuales. Independientemente del tipo, todos los contactos tienen asociada una lista con los mensajes que se han enviado.

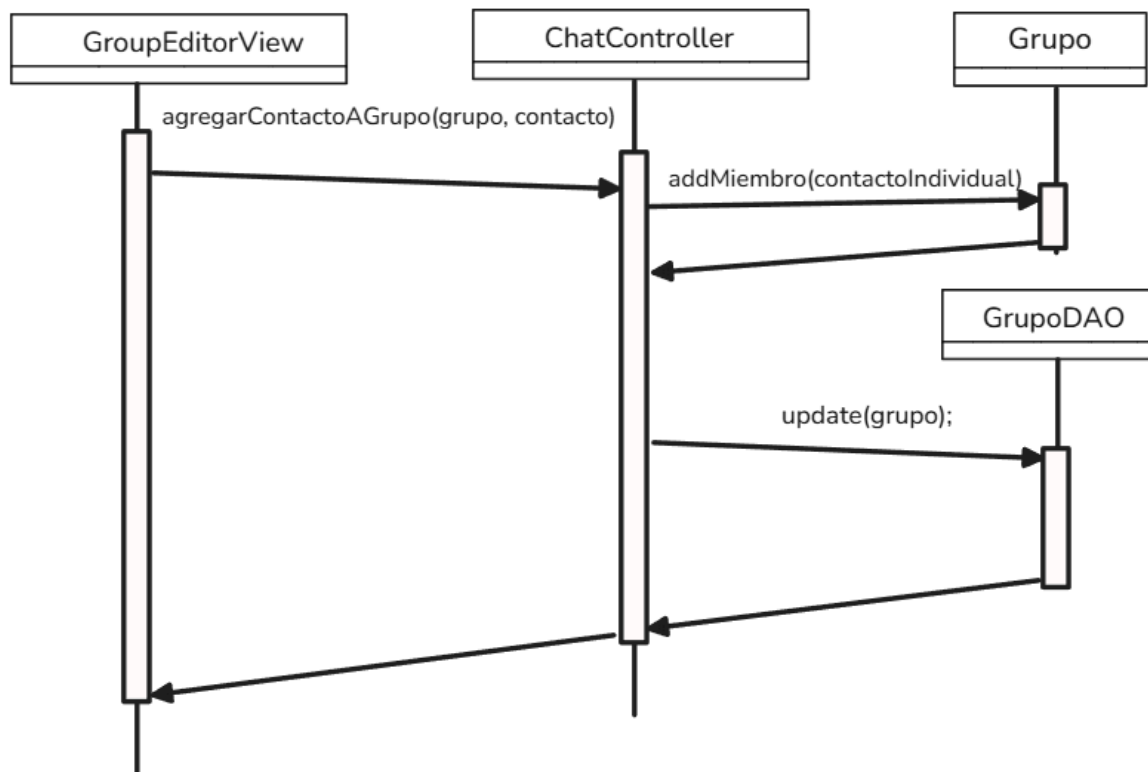
Cada mensaje conoce el usuario que ha enviado el mensaje y el contacto al que se lo ha enviado. Esto ayuda, por ejemplo, a indicar quien envió el mensaje en cada instante o si el usuario que recibe el mensaje tiene un contacto asociado al usuario que envía.





### 3. Diagrama de colaboración

El siguiente diagrama muestra las clases y metodos involucrados en la inserción de un contacto existente a un grupo existente.



En primer lugar, la clase 'GroupEditorView' de la vista indica al controlador 'ChatController' que quiere agregar un contacto 'contacto' al grupo 'grupo'.

El controlador le indica a la clase 'Grupo' que introduzca el contacto en su lista de miembros.

Por último, actualizamos el grupo en la base de datos con el método 'update' de la clase 'GrupoDAO'.



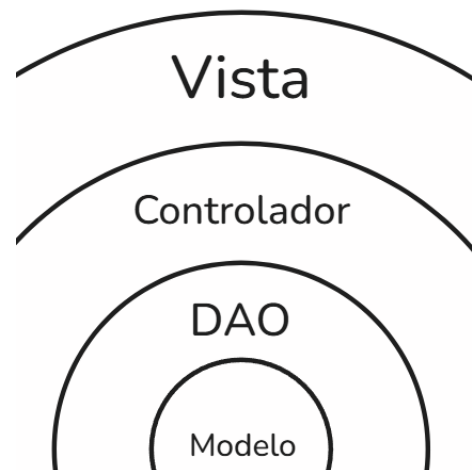
## 4. Diseño

En esta sección indicaremos el diseño estructural de la aplicación y algunas decisiones del diseño que ayudarán a la comprensión del desarrollo.

### 4.1 Arquitectura

Hemos intentado seguir los principios de la arquitectura limpia, concretamente utilizando un modelo por capas como el que vemos en la imagen. El objetivo es abstraer el conocimiento de las capas inferiores a las capas superiores. Con este enfoque evitaremos romper los patrones GRAPS y el código estará desacoplado.

En ocasiones excepcionales puede romperse este diseño por capas por cuestiones de eficiencia o por limitaciones técnicas.



- **Capa de presentación (Vista)**
  - Ubicada en el paquete `vista`.
  - Contiene todas las clases relacionadas con la interfaz gráfica de usuario.
  - Implementada con Swing para crear una interfaz de escritorio.
  - Las principales clases incluyen: MainView, ChatPanel, LoginView, RegisterView, etc.
- **Capa de lógica de negocio (Controlador)**
  - Ubicada en el paquete `dominio.controlador`.
  - Implementa la lógica de negocio y coordina las operaciones entre la vista y el modelo.
  - La clase principal es ChatController, que sigue el patrón Singleton.
- **Capa de persistencia**
  - Ubicada en el paquete `persistencia`.
  - Gestiona el almacenamiento y recuperación de datos.
  - Utiliza el patrón DAO (Data Access Object) para abstraer la implementación de la base de datos.
- **Capa de modelo (Dominio)**
  - Ubicada en el paquete `dominio.modelo`.
  - Define las entidades del sistema: Usuario, Contacto, ContactoIndividual, Grupo, Mensaje.
  - Implementa la lógica específica de cada entidad.

## 4.2 Decisiones de diseño relevantes

En este apartado mostraremos algunas de cuestiones que al principio no surgieron y durante el desarrollo han supuesto un problema al que encontrarle una solución.

### 4.2.1 Gestión de los mensajes

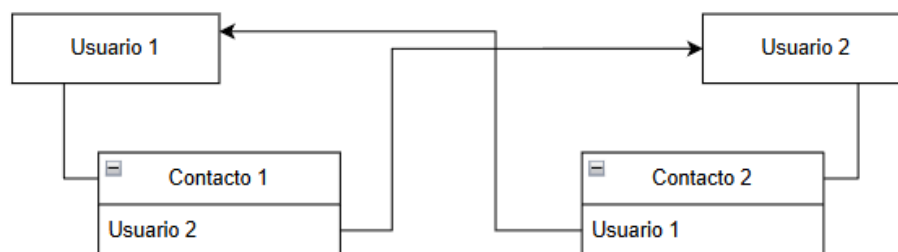
Durante el desarrollo planteamos distintas formas de gestionar cómo se deberían guardar los mensajes en la base de datos.

La primera opción era crear un mensaje en cada contacto con la misma información pero cambiando el campo 'Tipo' según si era el emisor o el receptor. Se nos presentó una complicación a altas etapas del desarrollo, incluso ya implementada la gestión del DAO. El problema que mencionamos ocurre cuando filtramos mensajes y queremos mostrar quien recibe el mensaje, produciendo un recorrido excesivo de las listas por cada mensaje para hallar los nombres y propietarios de cada mensaje enviado.

La segunda opción solucionaba dos problemas; el comentado en la opción uno, y una optimización que hacía que ambos contactos referenciara a los mismos mensajes. Esto se realizó quitando el campo 'Tipo' y añadiendo el campo 'Receptor', que sería el 'Contacto' que recibe el mensaje. De esta forma podemos indicar en cada momento quien recibe el mensaje y quien lo envía, y además, reduce las entradas en la base de datos a la mitad.

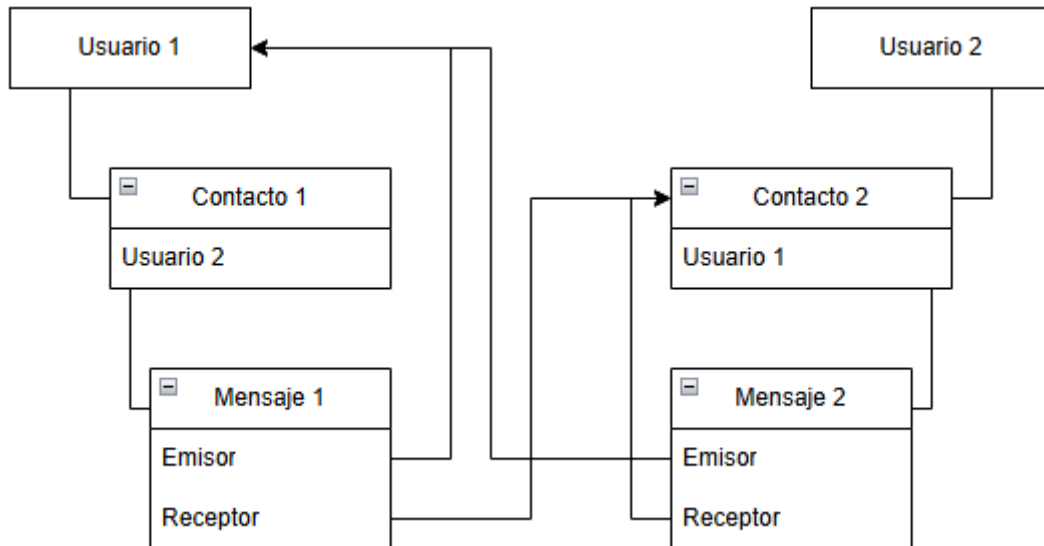
Al hacer pruebas sobre la eliminación de contactos que ya tenían un historial de mensajes comprobamos que se producía un error al eliminar mensajes de un contacto, ya que el DAO comprendía que al querer borrar un contacto se deben borrar sus entidades asociadas, borrando también las del otro contacto.

Para entender el problema, en la siguiente imagen podemos observar la representación de un chat en la aplicación. Un contacto no es más que el objeto que permite ver el chat, por lo que cada usuario necesita su propia visión individual del chat. De esta forma también podemos agregar a las personas por distintos nombres sin que afecte a su usuario.



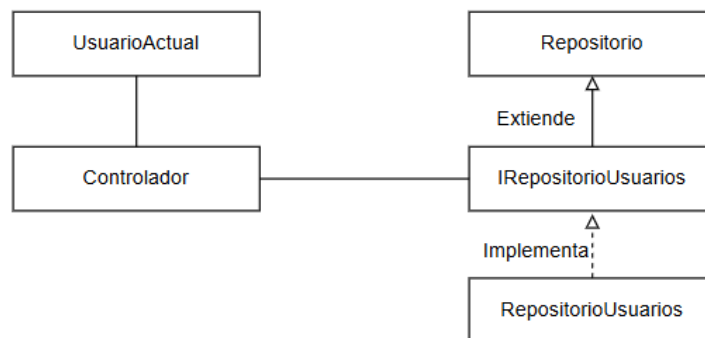


A continuación mostramos la solución final al problema anterior, una implementación híbrida entre la solución uno y dos. El mensaje ahora está duplicado para cada contacto y ambos conocen quien es el usuario emisor y el contacto receptor.

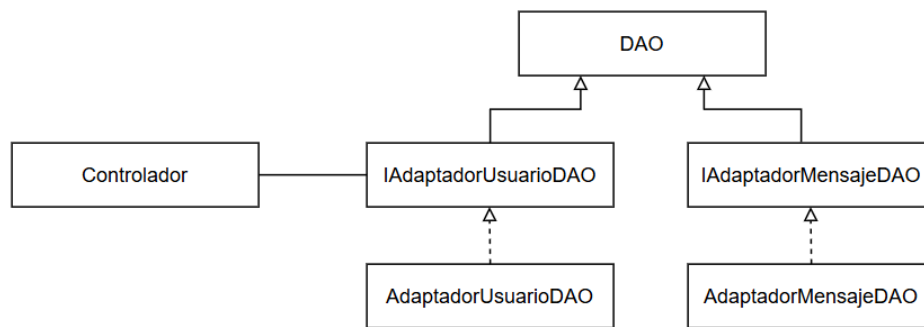


#### 4.2.1 Uso de interfaces

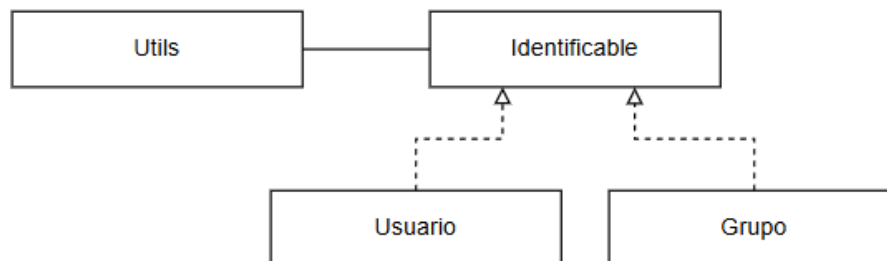
La mayoría del desarrollo ha sido orientado a interfaces, aquí mostraremos tres escenarios en los que se ha seguido este principio.



En primer lugar, creimos conveniente utilizar una interfaz padre llamada 'Repositorio' que obligará a las demás interfaces a implementar los mismo métodos. En este caso solo usamos el repositorio de usuarios, pero en un futuro podría ser un repositorio de cualquier otra cosa. El controlador usa la interfaz 'IRepositorioUsuarios' en lugar de la clase 'RepositorioUsuarios' para desacoplar el código al máximo.



En la implementación del patrón DAO también creimos necesario implementar una interfaz 'DAO' con las operaciones CRUD. Estas interfaces primarias utilizan generalización para dar libertad a las clases que las heredan. El controlador nuevamente vuelve a utilizar las interfaces en lugar de la implementación real para desacoplar el código.



Por último, creamos una interfaz 'Identificable' que obliga a implementar los métodos 'getId' y 'setId'. Esta interfaz nos es de gran ayuda para crear métodos que sean generales para todos los objetos que deban ser introducidos en la base de datos. Por ejemplo, en la clase 'Utils' se utiliza esta interfaz para un método que simplemente concatena los id en un único String separado por comas. En caso de no usar la interfaz, cada objeto debería tener su propio método para concatenar los id.

```
//
public static String concatenarIds(List<? extends Identificable> objetosIdentificables) {
    return objetosIdentificables.stream()
        .map(obj -> String.valueOf(obj.getId()))
        .collect(Collectors.joining(","));
}
```

### 4.3 Controlador Stub

Mientras se estaba desarrollando el controlador real, se estuvo trabajando con una clase imitadora 'Stub' que permitía crear datos estáticos para poder seguir el desarrollo de la interfaz visual.



## 5. Patrones

La aplicación implementa una serie de patrones de diseño para mejorar su estructura, mantenimiento y desarrollo por ello vamos a separarlos en Patrones de diseño de Creación, Estructurales y de Comportamiento.

### 5.1 Patrones de Creación

- **Patrón Singleton**

- Implementado en la clase **ChatController**.
- Garantiza una única instancia del controlador en toda la aplicación.
- Proporciona un punto centralizado para la lógica de negocio.

- **Patrón Factory Method**

- Implementado en la clase **FactoriaDAO**.
- Crea e inicializa los objetos DAO adecuados sin exponer la lógica de creación.

### 5.2 Patrones de Estructural

- **Patrón Composite**

- Implementado en las clases **Contacto**, **ContactoIndividual** y **Grupo**.
- Permite tratar de manera uniforme a contactos individuales y grupos.

- **Patrón Adapter**

- Implementado en las clases **AdaptadorUsuarioDAO**, **AdaptadorMensajeDAO**, **AdaptadorGrupoDAO** y **AdaptadorContactoIndividualDAO**.
- Convierte los objetos de dominio en un formato adecuado para el almacenamiento y viceversa.
- Proporciona una abstracción para operar con el sistema de persistencia subyacente.

### 5.3 Patrones de Comportamiento

- **Patrón Observer** (implícito en **Swing**)

- Utilizado en la interfaz gráfica para actualizar la vista cuando cambian los datos.
- Los listeners de eventos en componentes Swing implementan este patrón.

- **Patrón Strategy**

- Implementado en las clases de descuento (**Descuento**, **DescuentoBasico**, **DescuentoPorEdad**, **DescuentoPorFecha**).
- Permite seleccionar dinámicamente diferentes algoritmos de descuento para usuarios premium.



## 6. Manual de usuario

### 6.1 Inicio de sesión y registro

- **Iniciar la aplicación**

- Ejecuta el archivo **Launcher** para lanzar la aplicación .

- **Registrarse como nuevo usuario**

- Si no tiene cuenta, haga clic en "**Registrarse**".
- Complete el formulario con su nombre, fecha de nacimiento, correo electrónico, teléfono, contraseña, saludo e imagen de perfil, está desde URL o local.
- Haga clic en "**Registrar**" para crear su cuenta.

The image shows two side-by-side screenshots of the AppChat web application. The left screenshot is the 'AppChat - Registro' page, titled 'Crear nueva cuenta'. It contains a registration form with fields for: Nombre, Correo Electrónico, Teléfono, Contraseña (with a 'Repetir:' field), Fecha de nacimiento (with a calendar icon), Saludo inicial, and Foto de perfil. Below the form is a 'Seleccionar imagen' button. At the bottom are 'Cancelar' and 'Registrar' buttons. The right screenshot is the 'Seleccionar imagen' dialog box. It has the title 'Seleccionar imagen' and the text 'Puedes arrastrar una imagen aquí, seleccionarla desde tu ordenador...'. It features a large dashed box for dragging an image, with the text 'Arrastra una imagen aquí' inside. Below the box are buttons for 'Seleccionar de tu ordenador', 'Cargar URL', 'Cancelar', and 'Aceptar'.

- **Iniciar sesión**

- Introduzca su número de teléfono y contraseña.
- Haga clic en "**Iniciar sesión**" para acceder a su cuenta.



The image shows the AppChat login interface. At the top, there's a blue header with the 'AppChat' logo. Below it, a welcome message 'Bienvenido/a a AppChat' is displayed. There are two input fields: 'Número de teléfono' (Phone Number) and 'Contraseña' (Password). A blue 'Iniciar Sesión' (Login) button is positioned below the password field. A link '¿No tienes cuenta? Regístrate' (Don't you have an account? Register) is located below the login button. At the bottom, a copyright notice '© 2025 AppChat - Todos los derechos reservados' is visible.

## 6.2 Gestión de contactos

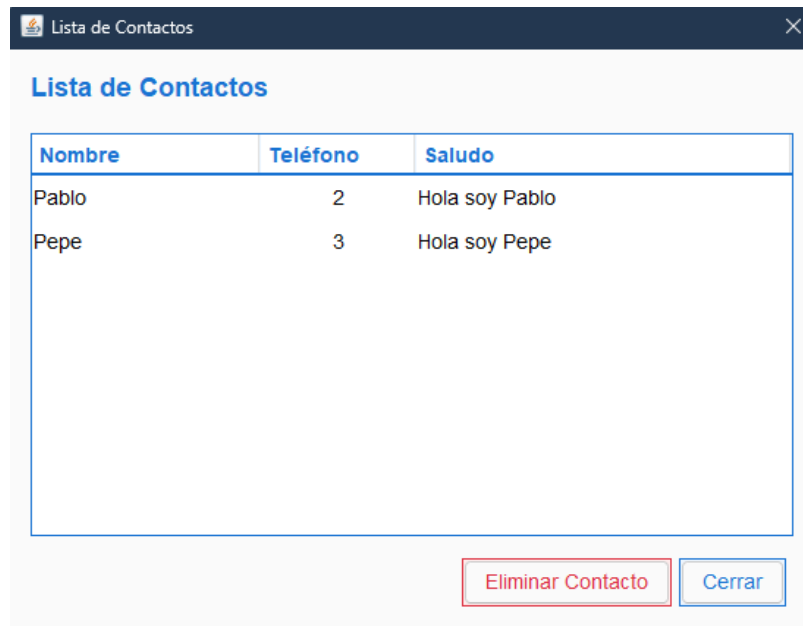
- **Añadir un contacto**

- En la pantalla principal, haga clic en el botón "+C" (Agregar Contacto).
- Introduzca el número de teléfono del contacto que desea añadir.
- Asigne un nombre a este contacto en su lista.
- Haga clic en "Añadir" para agregar el contacto.

The image shows a dialog box titled 'Agregar Nuevo Contacto'. It has a blue header with a close button. Below the header, there are two input fields: 'Nombre:' (Name) and 'Teléfono:' (Phone Number). At the bottom, there are two buttons: 'Cancelar' (Cancel) and 'Agregar Contacto' (Add Contact).

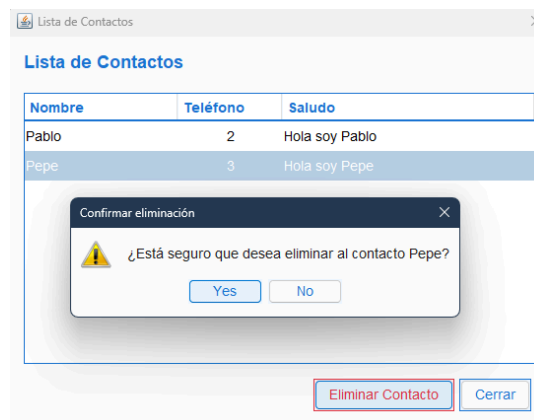
- **Ver lista de contactos**

- Haga clic en el botón "Contactos" para ver todos sus contactos.
- Esta lista muestra el nombre, número de teléfono y saludo de cada contacto.



- **Eliminar un contacto**

- En la lista de contactos, seleccione el contacto que desea eliminar.
- Haga clic en "Eliminar Contacto" y confirme la acción.



## 6.3 Gestión de grupos

- **Crear un grupo**


- Haga clic en el botón "+G" (Agregar Grupo).
- Asigne un nombre al grupo.
- Seleccione los contactos que desea incluir en el grupo.
- Puede añadir una imagen para el grupo.
- Haga clic en "Crear grupo" para finalizar.




Crear Grupo

### Crear nuevo grupo

**Contactos disponibles**




**Pablo**  
Tel: 2  
Hola soy Pablo



**Pepe**  
Tel: 3  
Hola soy Pepe

[Agregar al grupo ▶](#)

**Miembros del grupo**




**Pepe**  
Tel: 3  
Hola soy Pepe

[◀ Quitar del grupo](#)

Nombre del grupo:

Imagen del grupo:



[Seleccionar Imagen](#)

[Cancelar](#) [Crear Grupo](#)

- **Editar un grupo existente**

- Seleccione el grupo en su lista de contactos.
- Haga doble clic con el botón izquierdo sobre el grupo seleccionado.
- Puede editar la imagen y el nombre del grupo.
- Haga clic en "**Guardar cambios**".

Editar Grupo

### Editar Grupo



[Cambiar Imagen](#)

Nombre del grupo:

[Gestionar Miembros del Grupo](#)

[Eliminar Grupo](#) [Cancelar](#) [Guardar Cambios](#)



- **Añadir y eliminar contacto a un grupo existente**

- Seleccione el grupo en su lista de contactos.
- Haga doble clic con el botón izquierdo sobre el grupo seleccionado.
- Seleccione "**Gestionar Miembros del Grupo**".
- En la ventana emergente, seleccione los contactos que desea añadir o eliminar.
- Haga clic en "**Cerrar**" y posteriormente "**Guardar cambios**".



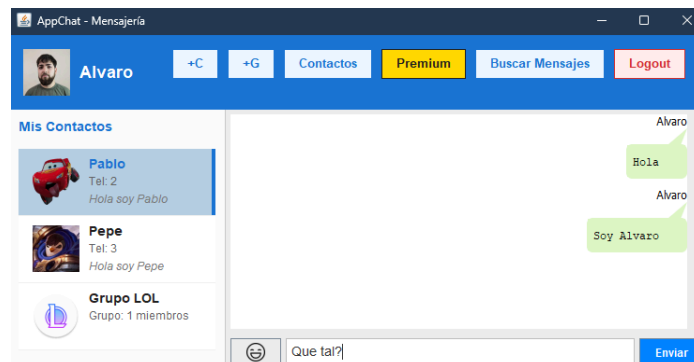
- **Eliminar un grupo**

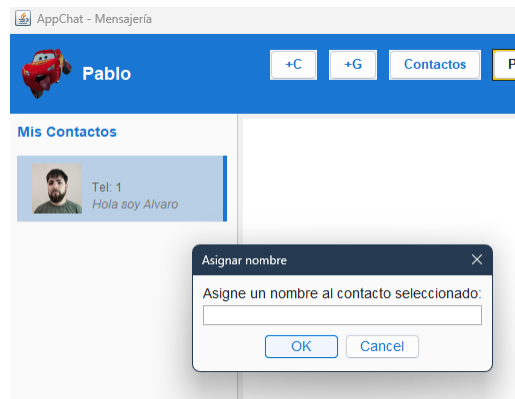
- Seleccione el grupo en su lista de contactos.
- Haga doble clic con el botón izquierdo sobre el grupo seleccionado.
- Seleccione "**Gestionar Miembros del Grupo**".
- Haga clic en "**Eliminar Grupo**".



## 6.4 Mensajería

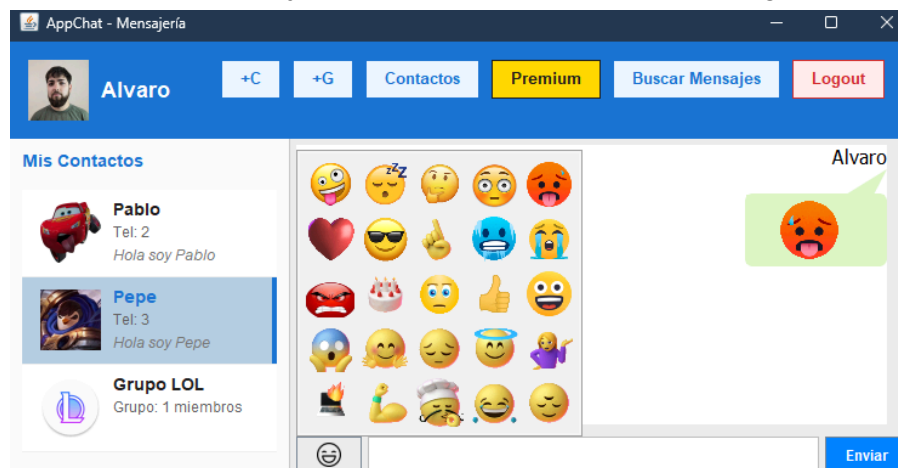
- **Enviar un mensaje a un contacto o grupo**
  - Seleccione el contacto o grupo en su lista de contactos.
  - Escriba su mensaje en el área de texto en la parte inferior.
  - Presione Enter o haga clic en el botón "Enviar".
  - Si el destinatario no te tiene agregado, puedes elegir su nombre.





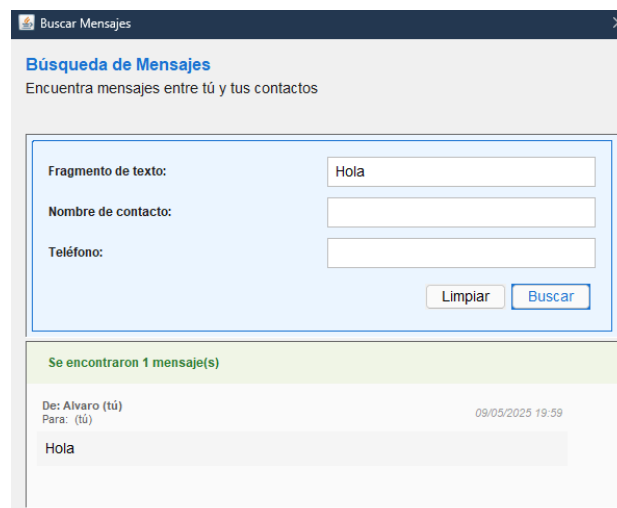
- **Enviar un emoji**

- Haga clic en el botón de emoji junto al área de texto.
- Seleccione el emoji que desea enviar de la lista desplegable.



- **Buscar mensajes**

- Haga clic en el botón de "Buscar Mensajes".
- Introduzca el texto que desea buscar o filtrar por contacto o número de teléfono.
- Los resultados mostrarán todos los mensajes que coincidan con su búsqueda.





## 6.5 Funciones premium

- **Activar cuenta premium**
  - Haga clic en el botón "Premium" en la barra superior.
  - Seleccione la opción de suscripción que desee.
  - Complete el proceso de activación siguiendo las instrucciones.

Actualizar a Premium

**¡Actualiza a Premium!**  
Disfruta de ventajas exclusivas

Beneficios Premium

- Generar documento PDF con contactos y grupos  
Genera un PDF con tus contactos, grupos y sus mensajes. Incluye información detallada de cada conversación.

Seleccionar descuento

<b>Precio original:</b>	9,99 €
<b>Descuento:</b>	Descuento por estudia...
<b>Descuento aplica...</b>	2,50 €
<b>Precio final:</b>	7,49 €

Cancelar Actualizar a Premium

- **Generar informes PDF**
  - Con una cuenta premium, haga clic en "Generar PDF" en el menú premium.
  - Seleccione la ubicación donde guardar el archivo.
  - El informe incluirá estadísticas de uso y resúmenes de conversaciones.

Opciones Premium

**Acceso Premium**  
¡Gracias por ser usuario Premium!

Generar PDF de contactos y mensajes

Genera un documento PDF con todos tus contactos, grupos y mensajes organizados en un formato elegante. Ideal para tener un respaldo de tus conversaciones.

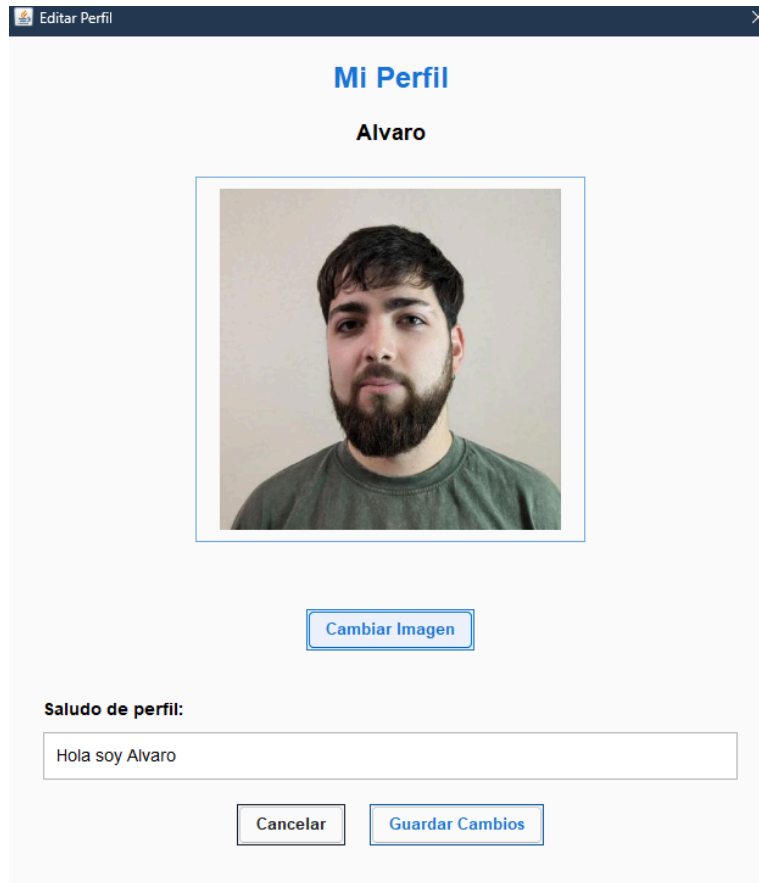
Generar PDF

Cerrar

## 6.6 Perfil de usuario

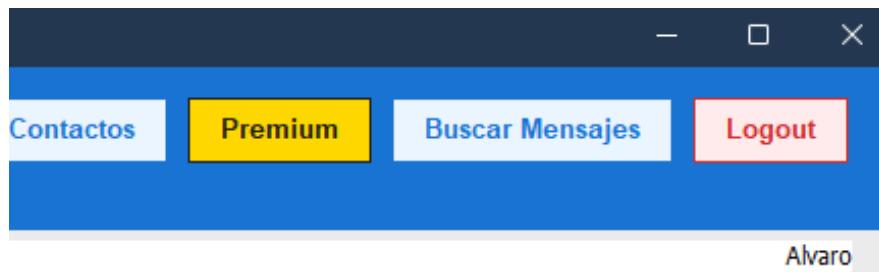
- **Editar perfil**

- Haga clic en su foto de perfil en la esquina superior izquierda.
- Modifique su saludo personal o foto de perfil.
- Haga clic en "**Guardar cambios**" para actualizar su perfil.



- **Cerrar sesión**

- Haga clic en el botón "**Logout**" en la barra superior.
- Confirme que desea salir de la aplicación.





## 7. Conclusiones

Este proyecto nos acerca al desarrollo de una aplicación que podría ser realmente lanzada. Sin duda aquello con lo que más nos quedamos de esta prueba es con la importancia de las interfaces. Extraer lo que hace una clase del como lo hace es realmente útil para la extensión y modificación de cualquier aplicación. Además fomenta la producción en equipo, ya que un compañero puede estar desarrollando la implementación concreta mientras que otro puede estar utilizando la interfaz para otra capa del desarrollo.

El tiempo que hemos empleado para conseguir el resultado final ha sido exageradamente alto. Al principio perdimos muchas horas al empezar sin planificación, posteriormente retrocedimos para empezar de nuevo con la planificación y distribución de responsabilidades necesaria. Hemos repasado y encontrado muchos fallos que han provocado rehacer algunas secciones y eso nos ha frustrado en muchas ocasiones. También contamos el tiempo de investigación para la comprensión de algunos de los patrones que han sido utilizados.

El total de horas aproximadas son 150h cada uno.