DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERIA EN SISTEMAS E INFORMÁTICA

PROGRAMACIÓN MOVIL

CHAT

INTEGRANTES:

PASPUEL YÁNEZ MAYRA ALEXANDRA

QUISTANCHALA SUNTAXI KARLA DANIELA

VILLARRUEL ARCINIEGA MICHAEL ALEJANDRO

NRC:

6112

06 DE AGOSTO DE 2020

SANGOLQUÍ – ECUADOR

MAYO – SEPTIEMBRE 2020

Contenidos

1. Introducción

Una aplicación móvil es un programa de software que puede descargar y acceder directamente usando su teléfono u otro dispositivo móvil, como una tableta o un reproductor de música.

Los dispositivos móviles se están apoderando de los equipos de escritorio: la cantidad de usuarios de dispositivos móviles y el tiempo dedicado a los dispositivos móviles están experimentando un crecimiento constante.

Brindar una experiencia fluida y atractiva en dispositivos móviles ahora es más importante que nunca, y brinda una verdadera ventaja competitiva a las empresas que lo hacen bien.

Sin embargo, la conectividad en todo momento es cada día más importante. Por eso se han impuesto los móviles como principal medio de acceso a Internet y el mundo del chat ha tenido que actualizarse para ofrecer lo mejor en aplicaciones móviles adaptadas a las necesidades de sus usuarios.

2. Objetivo

- Construir una aplicación móvil que permita emular un sistema de chat para el envio y recepción de mensajes de texto mediante el uso de la herramienta Android Studio y Firebase.

3. Marco teórico

3.1. Código nativo

El código nativo es la programación de computadora (código) que se compila para ejecutarse con un procesador en particular (como un procesador Intel x86- class) y su conjunto de instrucciones. Si el mismo programa se ejecuta en una computadora con un procesador diferente, se puede proporcionar un software para que la computadora emule el procesador original. En este caso, el programa original se ejecuta en "modo de emulación" en el nuevo procesador y casi seguramente más lentamente que en el modo nativo en el procesador original. (El programa se puede reescribir y volver a compilar para que se ejecute en el nuevo procesador en modo nativo).

El código nativo también se puede distinguir del código de bytes (a veces llamado código interpretado), una forma de código que se puede decir que se ejecuta en una máquina virtual (por ejemplo, la máquina virtual Java). La máquina virtual es un programa que convierte el bytecode generalizado de plataforma en el código nativo que se ejecutará en un procesador específico. Los compiladores .NET de Microsoft para sus lenguajes Visual Basic, C # y JavaScript producen bytecode (que Microsoft llama lenguaje intermedio). El código de bytes de Java y el lenguaje intermedio de Microsoft se pueden compilar en código nativo antes de la ejecución por un compilador justo a tiempo para un rendimiento más rápido

3.2. Android Studio

Android Studio es el entorno oficial de desarrollo integrado (IDE) para el desarrollo de aplicaciones de Android. Se basa en IntelliJ IDEA, un entorno de desarrollo integrado de Java para software, e incorpora sus herramientas de edición y desarrollo de código.

Para admitir el desarrollo de aplicaciones dentro del sistema operativo Android, Android Studio utiliza un sistema de compilación basado en Gradle, emulador, plantillas de código e integración de Github . Cada proyecto en Android Studio tiene una o más modalidades con código fuente y archivos de recursos. Estas modalidades incluyen módulos de aplicaciones de Android, módulos de biblioteca y módulos de Google App Engine.

Android Studio usa una función Instant Push para enviar cambios de código y recursos a una aplicación en ejecución. Un editor de código ayuda al desarrollador a escribir el código y ofrece la finalización, refracción y análisis del código. Las aplicaciones creadas en Android Studio se compilan en el formato APK para su envío a Google Play Store.

3.2.1. ¿Qué es gradle?

Gradle es una herramienta de automatización de compilación que parece ser más fácil que los configuradores de proyectos basados en XML tradicionales y se creó para proyectos grandes. Una ventaja es que sabe qué partes del árbol de compilación están actualizadas, por lo que no es necesario volver a ejecutarlas. Gradle está escrito en Java y Groovy, lo que hace que sea relativamente fácil hacer las cosas básicas necesarias para una aplicación. Gradle se introdujo en 2007, pero solo se ha utilizado para Android desde el lanzamiento de Android Studio. Tenga en cuenta que cada módulo en un proyecto tendrá su propio archivo Gradle. Gradle proporciona una manera fácil de configurar los detalles de la aplicación, incluida la versión de compilación y la versión del SDK.

3.2.2. Android SDK, AVD Manager y ADM

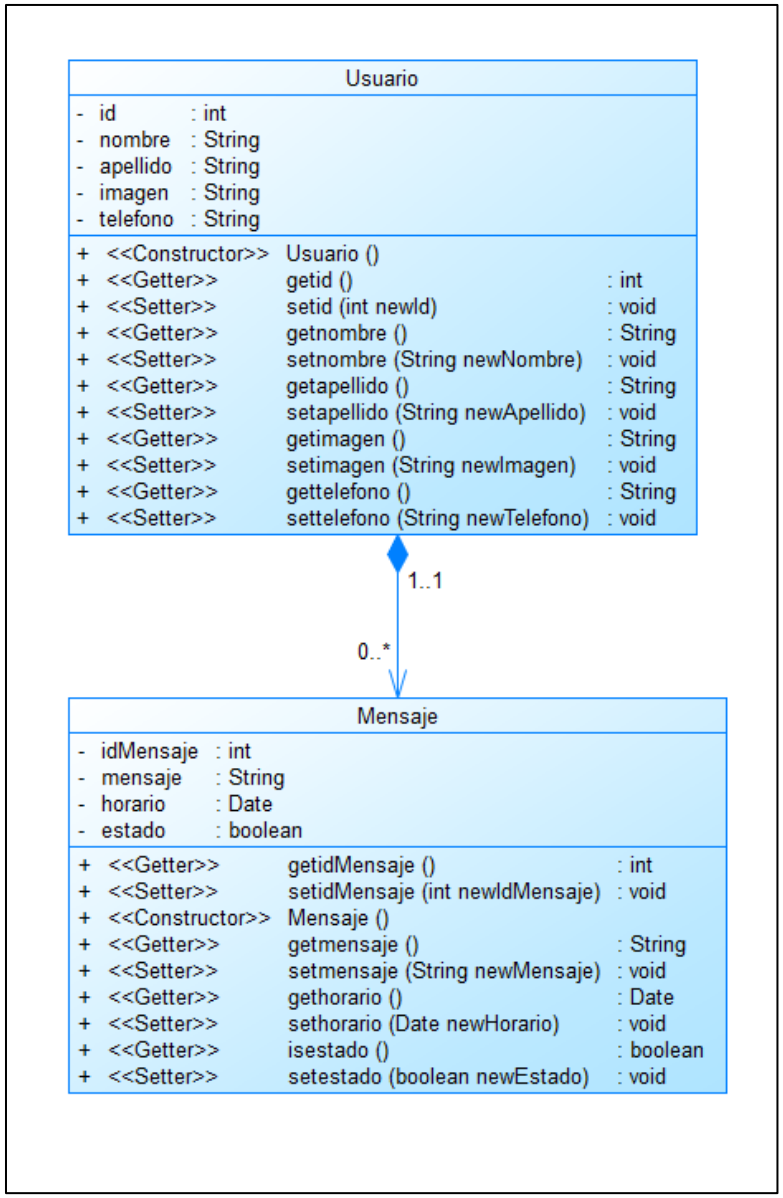
El SDK de Android incluye todas las bibliotecas y archivos necesarios para que los desarrolladores de Android puedan comenzar. Lo bueno de Android Studio es que el SDK está integrado y es de fácil acceso simplemente haciendo clic en un botón en la barra de herramientas superior. Los elementos junto al icono del SDK Manager incluyen el Administrador de dispositivo virtual de Android y el Monitor de dispositivo Android. AVD Manager le permite configurar dispositivos virtuales Android para probar aplicaciones. Puede configurar casi cualquier cosa, desde el tamaño del dispositivo hasta la arquitectura del conjunto de instrucciones. Si selecciona una arquitectura de conjunto de instrucciones Intel x86_64, puede ejecutar el AVD en algo conocido como "modo virt rápido", esto utiliza el Administrador de ejecución acelerada de hardware de Intel (HAXM) que permite una experiencia muy fluida al ejecutar un AVD.

4. Conclusión

- La aplicación móvil de chat, además de que es una aplicación totalmente intuitiva, permite establecer comunicación en todo momento y mantener conversaciones con diferentes usuarios registrados.

5. Anexos

5.1. Modelo



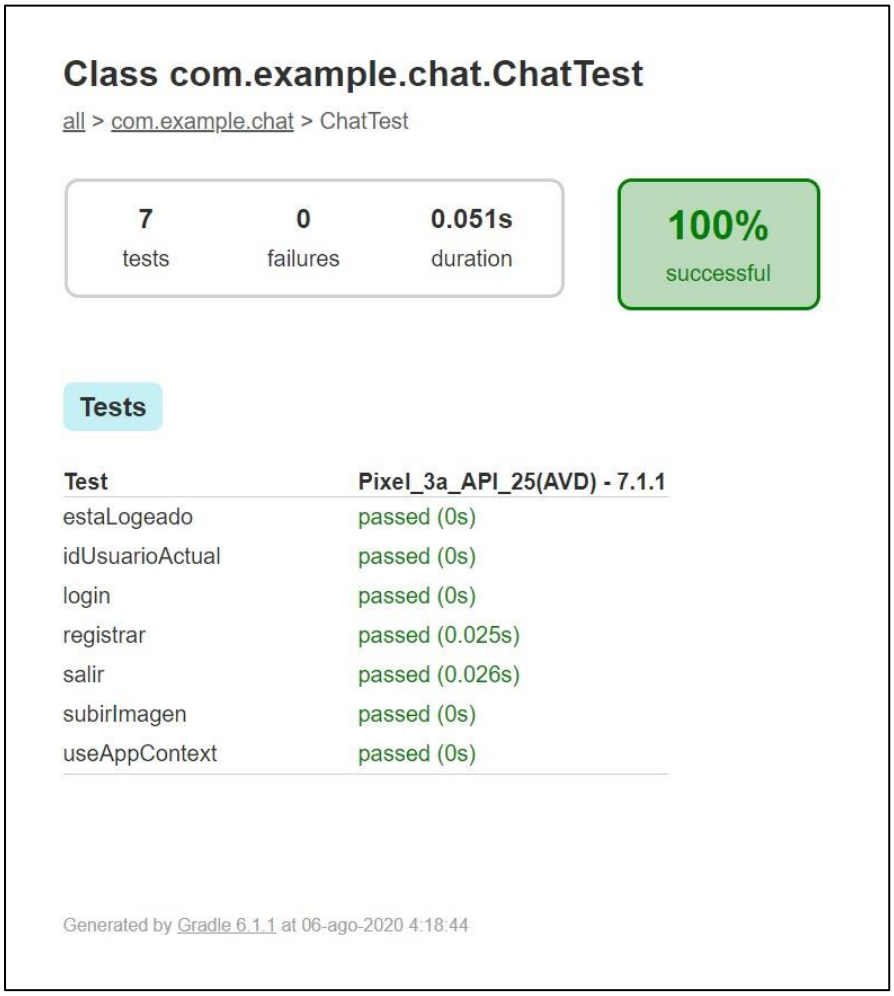
5.2. Requisitos

	Enunciado de la historia				Criterios de aceptación			
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Criterio de aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
R001	Como usuario	Necesito crear una cuenta en la aplicación de mensajería.	Con la finalidad de registrarme en el sistema.	1	Formulario de registro completado de manera correcta.	Todos los campos del formulario deben ser llenados con datos válidos.	Cuando se presiona el botón aceptar	Se registra correctamente en el sistema.
				2	Formulario de registro con datos incorrectos.	El formulario contiene campos vacios o datos incorrectos.	Cuando se presiona el botón aceptar	Se muestra un mensaje con el error correspondiente.
R002	Como usuario	Necesito iniciar sesión en la aplicación de mensajería.	Con la finalidad de acceder a la aplicación.	1	Autenticación de usuario y contraseña válida.	Todos los campos del formulario de inicio de sesión deben ser válidos.	Cuando se presiona el botón Ingresar	Se despliega la pantalla de inicio de la aplicación.
				2	Autenticación de usuario y contraseña inválida.	Los campos del formulario de inicio de sesión contienen datos erroneos.	Cuando se presiona el botón Ingresar	Se muestra un mensaje con el error correspondiente.
R003	Como usuario	Necesito enviar mensajes de texto.	Con la finalidad de comunicarme con otro	1	Mensaje enviado y no recibido.	El mensaje de texto fue enviado con éxito pero no	Cuando se presiona el botón Enviar	Se mostrará el mensaje en el chat con el estado correspodiente.

			usuario registrado en la aplicación.			recibido por el destinatario.		
				2	Mensaje enviado y recibido.	El mensaje de texto fue enviado y recibido exitosamente por el destinatario.	Cuando se presiona el botón Enviar	Se mostrará el mensaje en el chat con el estado correspodiente.
				3	Mensaje no enviado	El mensaje de texto no fue enviado al destinatario.	Cuando se presiona el botón Enviar	Se mostrará el mensaje en el chat con el estado correspodiente.
R004	Como usuario	Necesito recibir mensajes de texto.	Con la finalidad de comunicarme con otro usuario registrado en la aplicación.	1	Mensaje recibido.	El mensaje de texto fue recibido.	Cuando ingrese a la aplicación.	Se mostrará el mensaje en el chat.
R005	Como usuario	Necesito ver mi listado de chats.	Con la finalidad de ver los mensajes que he recibido.	1	Lista de chats.	Todos los chats serán mostrados en la página de inicio.	Cuando ingrese a la aplicación.	Se mostrará la lista de chats.
R006	Como programador	Necesito almacenar un mensaje con sus datos asociados.	Con la finalidad de obtener persistencia en los datos.	1	Datos guardados.	El mensaje contiene todos los datos.	Cuando envie o reciba un mensaje	Se almacenará el mensaje con su información en la base de datos.
R007	Como programador	Necesito encriptar las contraseñas de los usuarios registrados.	Con la finalidad de proporcionar seguridad a los usuarios.	1	Contraseña encriptada	La contraseña es válida.	Cuando un usuario se registre en la aplicación.	Se almacenará la contraseña encriptada en la base de datos.

5.3. Test

5.3.1. Reporte



Class com.example.chat.ChatTest

all > com.example.chat > ChatTest

7	0	0.051s	100%
tests	failures	duration	successful

Tests

Test	Pixel_3a_API_25(AVD) - 7.1.1
estaLogeado	passed (0s)
idUsuarioActual	passed (0s)
login	passed (0s)
registrar	passed (0.025s)
salir	passed (0.026s)
subirImagen	passed (0s)
useAppContext	passed (0s)

Generated by Gradle 6.1.1 at 06-ago-2020 4:18:44

5.3.2. Código

- **Clase ChatTest**

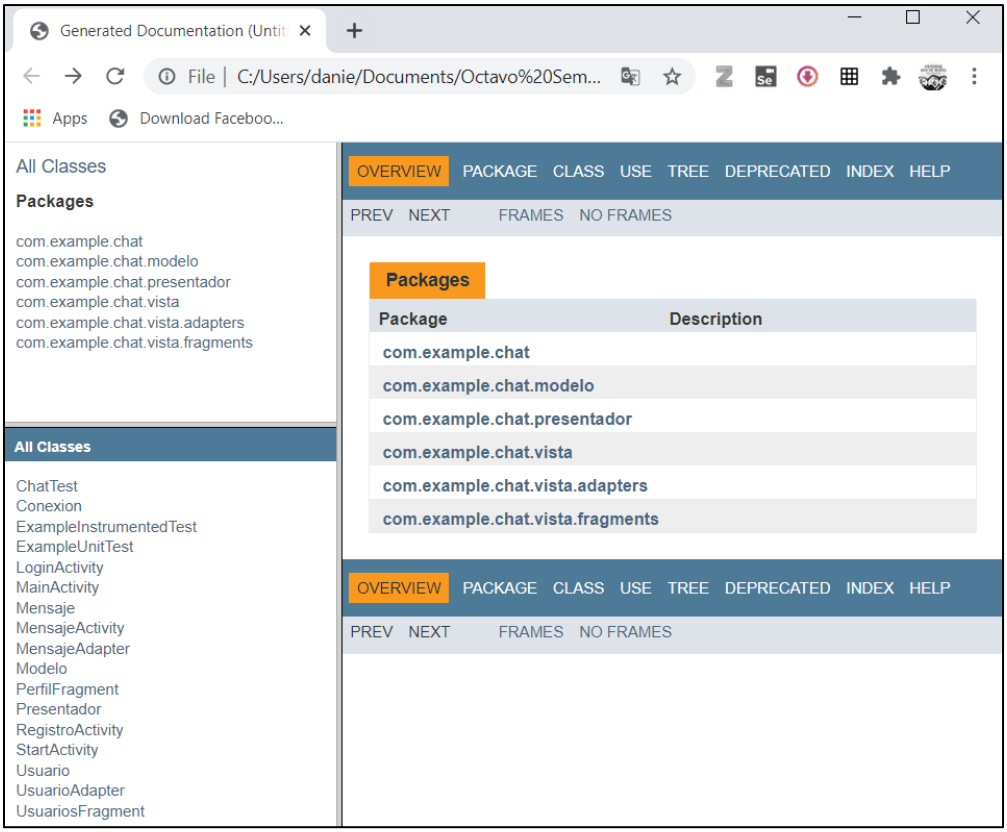
```
package com.example.chat;

import android.content.Context;
import android.net.Uri;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.platform.app.InstrumentationRegistry;

import com.example.chat.presentador.Presentador;

import org.junit.Test;
import org.junit.runner.RunWith;
```

```
import static org.junit.Assert.assertEquals;

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ChatTest {
    private Presentador presentador = new Presentador();

    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext =
InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals("com.example.chat", appContext.getPackageName());
    }

    @Test
    public void login() {
        Context appContext =
InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals(false, presentador.login(appContext,"",""));
    }

    @Test
    public void salir() {
        assertEquals(true, presentador.salir());
    }

    @Test
    public void estaLogeado() {
        assertEquals(false, presentador.estaLogeado());
    }

    @Test
    public void registrar() {
        Context appContext =
InstrumentationRegistry.getInstrumentation().getTargetContext();

assertEquals(true,presentador.registrar(appContext,"jvega","jvega@gma.com","1234567
8"));
    }

    @Test
    public void subirImagen() {
        Context appContext =
InstrumentationRegistry.getInstrumentation().getTargetContext();
        presentador.subirImagen(Uri.parse("https://q-
cf.bstatic.com/images/hotel/max500/161/161016875.jpg"),appContext);
    }

    @Test
    public void idUsuarioActual() {
```

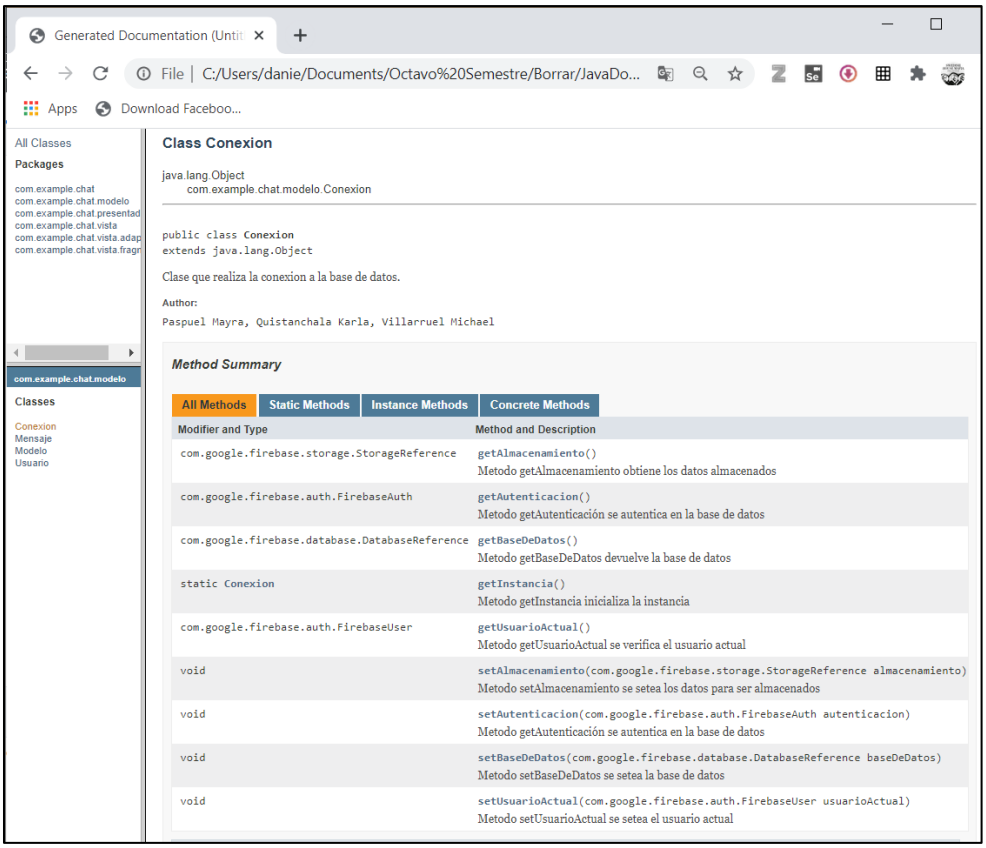
```
        assertEquals(null,presentador.idUsuarioActual());
    }

}
```

5.4. JavaDocs

- **Index.html**



- **Modelo Package**
  - **Conexión**



Generated Documentation (Untit

File | C:/Users/danie/Documents/Octavo%20Semestre/Borrar/JavaDo...

Apps Download Faceboo...

All Classes

Packages

com.example.chat
com.example.chat.modelo
com.example.chat.presentad
com.example.chat.vista
com.example.chat.vista.adap
com.example.chat.vista.fragr

com.example.chat.modelo

Classes

Conexion
Mensaje
Modelo
Usuario

Class Conexion

java.lang.Object
com.example.chat.modelo.Conexion

```
public class Conexion
extends java.lang.Object
```

Clase que realiza la conexion a la base de datos.

Author:

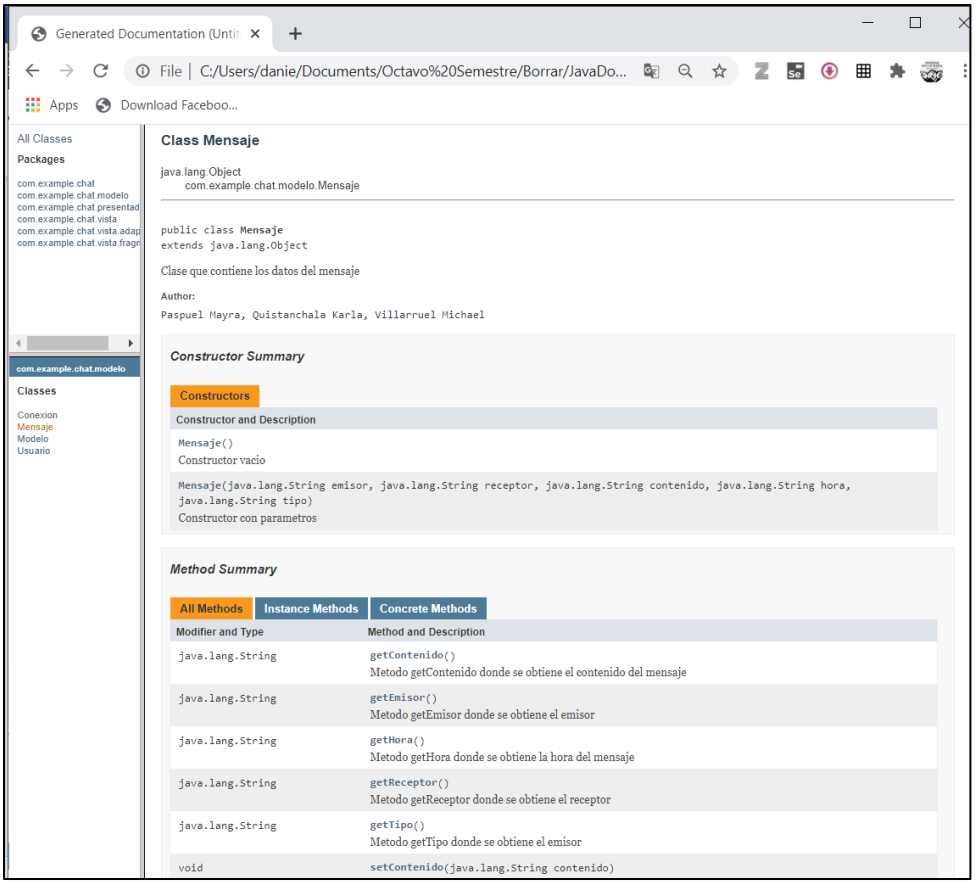
Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
com.google.firebase.storage.StorageReference	getAlmacenamiento() Metodo getAlmacenamiento obtiene los datos almacenados
com.google.firebase.auth.FirebaseAuth	getAutenticacion() Metodo getAutenticación se autentica en la base de datos
com.google.firebase.database.DatabaseReference	getBaseDeDatos() Metodo getBaseDeDatos devuelve la base de datos
static Conexion	getInstancia() Metodo getInstancia inicializa la instancia
com.google.firebase.auth.FirebaseUser	getUsuarioActual() Metodo getUsuarioActual se verifica el usuario actual
void	setAlmacenamiento(com.google.firebase.storage.StorageReference almacenamiento) Metodo setAlmacenamiento se setea los datos para ser almacenados
void	setAutenticacion(com.google.firebase.auth.FirebaseAuth autenticacion) Metodo getAutenticación se autentica en la base de datos
void	setBaseDeDatos(com.google.firebase.database.DatabaseReference baseDeDatos) Metodo setBaseDeDatos se setea la base de datos
void	setUsuarioActual(com.google.firebase.auth.FirebaseUser usuarioActual) Metodo setUsuarioActual se setea el usuario actual

  - **Mensaje**



Generated Documentation (Untit

File | C:/Users/danie/Documents/Octavo%20Semestre/Borrar/JavaDo...

Apps Download Faceboo...

All Classes

Packages

com.example.chat
com.example.chat.modelo
com.example.chat.presentad
com.example.chat.vista
com.example.chat.vista.adap
com.example.chat.vista.fragr

com.example.chat.modelo

Classes

Conexion
Mensaje
Modelo
Usuario

Class Mensaje

java.lang.Object
com.example.chat.modelo.Mensaje

```
public class Mensaje
extends java.lang.Object
```

Clase que contiene los datos del mensaje

Author:

Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Constructor Summary

Constructors

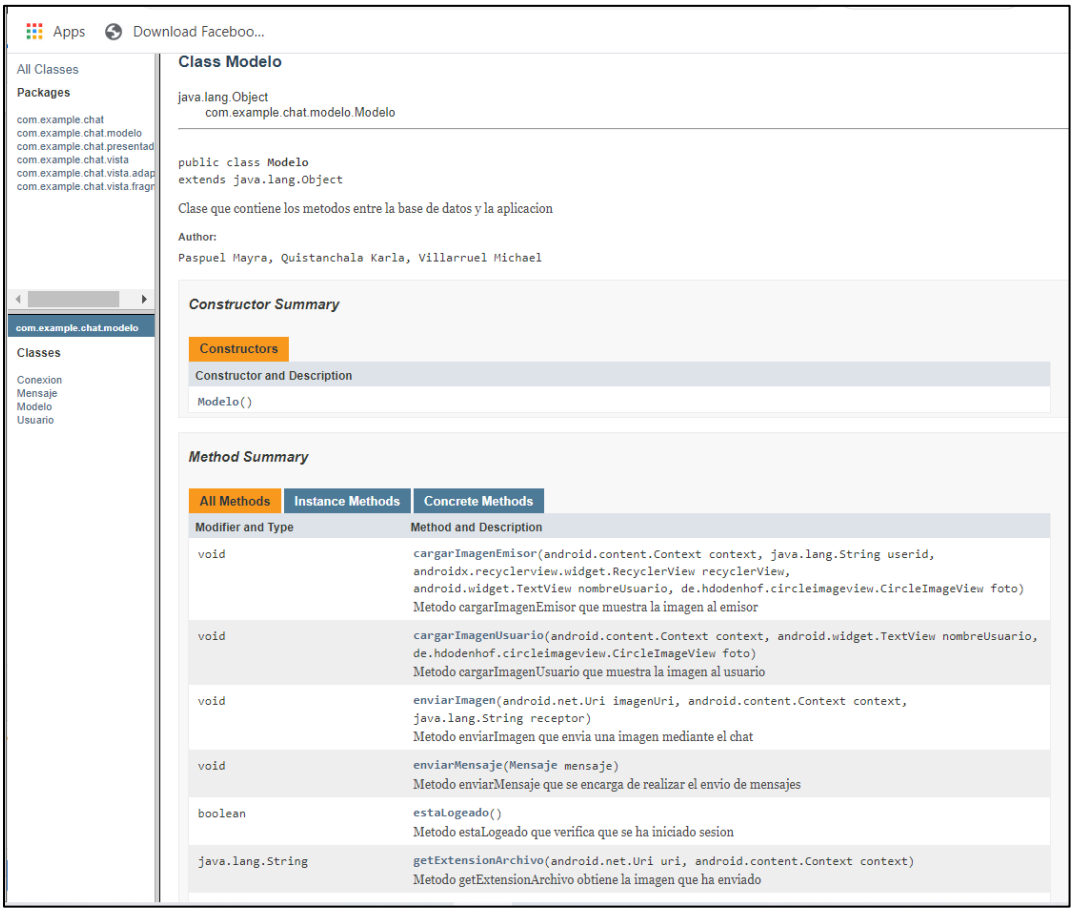
Constructor and Description
Mensaje() Constructor vacio
Mensaje(java.lang.String emisor, java.lang.String receptor, java.lang.String contenido, java.lang.String hora, java.lang.String tipo) Constructor con parametros

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getContenido() Metodo getContenido donde se obtiene el contenido del mensaje
java.lang.String	getEmisor() Metodo getEmisor donde se obtiene el emisor
java.lang.String	getHora() Metodo getHora donde se obtiene la hora del mensaje
java.lang.String	getReceptor() Metodo getReceptor donde se obtiene el receptor
java.lang.String	getTipo() Metodo getTipo donde se obtiene el emisor
void	setContenido(java.lang.String contenido)

- **Modelo**



Apps Download Faceboo...

All Classes

Packages

com.example.chat
com.example.chat.modelo
com.example.chat.presentad
com.example.chat.vista
com.example.chat.vista.adap
com.example.chat.vista.fragr

com.example.chat.modelo

Classes

Conexion
Mensaje
Modelo
Usuario

Class Modelo

java.lang.Object
com.example.chat.modelo.Modelo

public class **Modelo**
extends java.lang.Object

Clase que contiene los metodos entre la base de datos y la aplicacion

Author:

Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Constructor Summary

Constructors

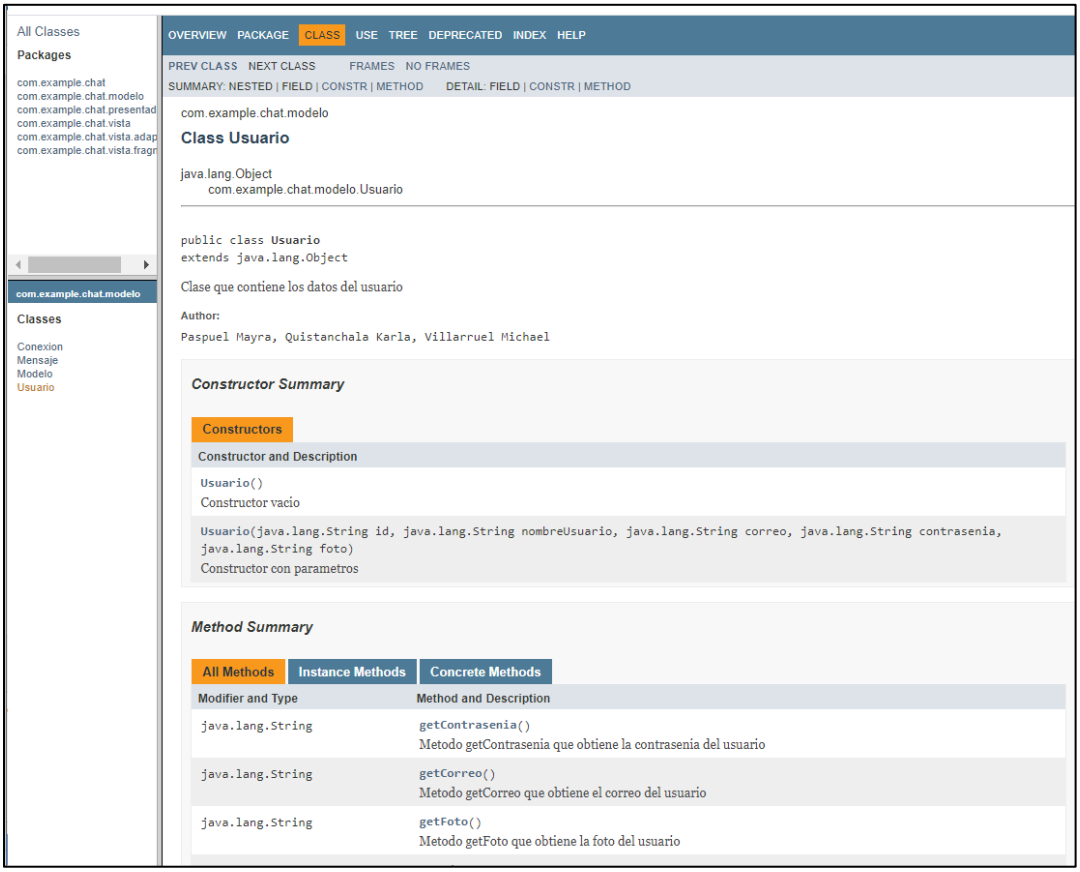
Constructor and Description
Modelo()

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	cargarImagenEmisor(android.content.Context context, java.lang.String userid, androidx.recyclerview.widget.RecyclerView recyclerView, android.widget.TextView nombreUsuario, de.hdodenhof.circleimageview.CircleImageView foto) Metodo cargarImagenEmisor que muestra la imagen al emisor
void	cargarImagenUsuario(android.content.Context context, android.widget.TextView nombreUsuario, de.hdodenhof.circleimageview.CircleImageView foto) Metodo cargarImagenUsuario que muestra la imagen al usuario
void	enviarImagen(android.net.Uri imagenUri, android.content.Context context, java.lang.String receptor) Metodo enviarImagen que envia una imagen mediante el chat
void	enviarMensaje(Mensaje mensaje) Metodo enviarMensaje que se encarga de realizar el envio de mensajes
boolean	estaLogeado() Metodo estaLogeado que verifica que se ha iniciado sesion
java.lang.String	getExtensionArchivo(android.net.Uri uri, android.content.Context context) Metodo getExtensionArchivo obtiene la imagen que ha enviado

- **Usuario**



All Classes

Packages

com.example.chat
com.example.chat.modelo
com.example.chat.presentad
com.example.chat.vista
com.example.chat.vista.adap
com.example.chat.vista.fragr

com.example.chat.modelo

Classes

Conexion
Mensaje
Modelo
Usuario

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.example.chat.modelo

Class Usuario

java.lang.Object
com.example.chat.modelo.Usuario

public class **Usuario**
extends java.lang.Object

Clase que contiene los datos del usuario

Author:

Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Constructor Summary

Constructors

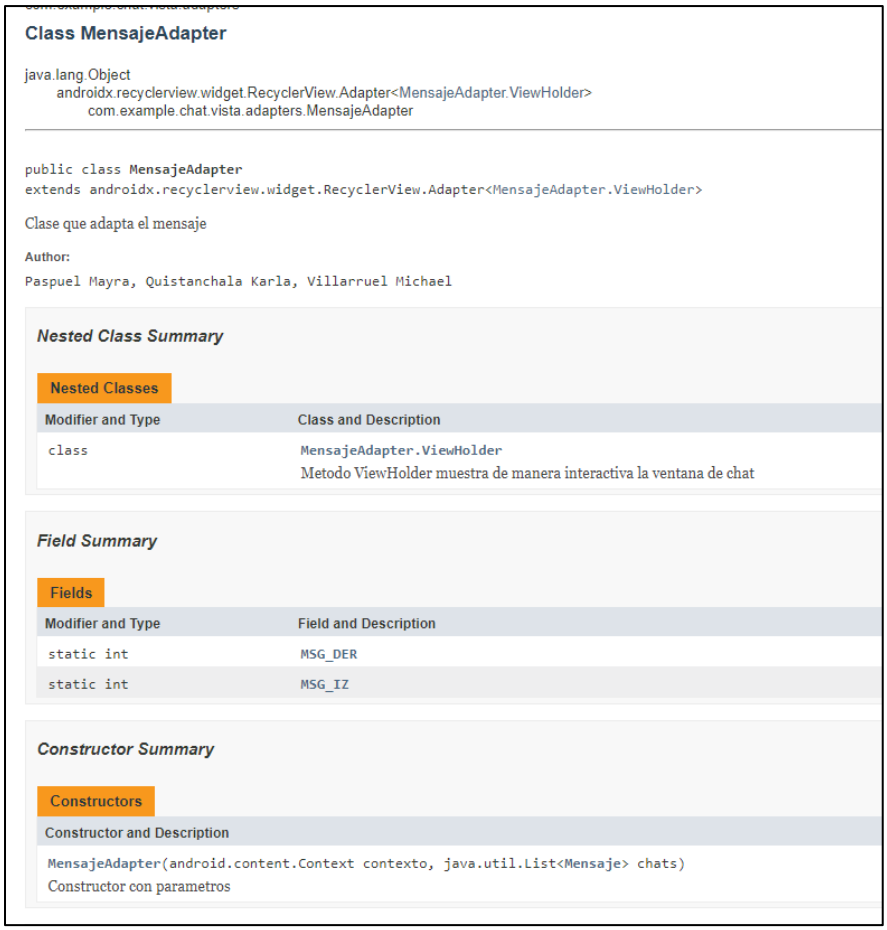
Constructor and Description
Usuario() Constructor vacio
Usuario(java.lang.String id, java.lang.String nombreUsuario, java.lang.String correo, java.lang.String contrasenia, java.lang.String foto) Constructor con parametros

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getContrasenia() Metodo getContrasenia que obtiene la contrasenia del usuario
java.lang.String	getCorreo() Metodo getCorreo que obtiene el correo del usuario
java.lang.String	getFoto() Metodo getFoto que obtiene la foto del usuario

- **Adapters Package**
  - **MensajeAdapter**



Class MensajeAdapter

java.lang.Object
androidx.recyclerview.widget.RecyclerView.Adapter<MensajeAdapter.ViewHolder>
com.example.chat.vista.adapters.MensajeAdapter

```
public class MensajeAdapter
extends androidx.recyclerview.widget.RecyclerView.Adapter<MensajeAdapter.ViewHolder>
```

Clase que adapta el mensaje

Author:

Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
class	MensajeAdapter.ViewHolder Metodo ViewHolder muestra de manera interactiva la ventana de chat

Field Summary

Fields

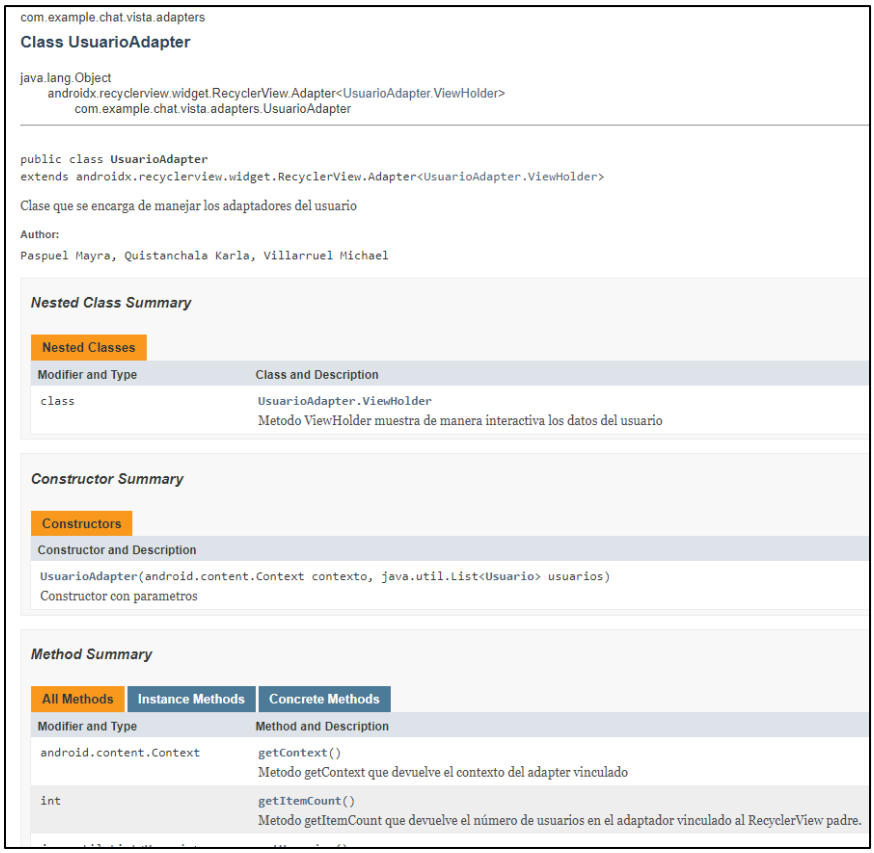
Modifier and Type	Field and Description
static int	MSG_DER
static int	MSG_IZ

Constructor Summary

Constructors

Constructor and Description
MensajeAdapter(android.content.Context contexto, java.util.List<Mensaje> chats) Constructor con parametros

  - **UsuarioAdapter**



com.example.chat.vista.adapters

Class UsuarioAdapter

java.lang.Object
androidx.recyclerview.widget.RecyclerView.Adapter<UsuarioAdapter.ViewHolder>
com.example.chat.vista.adapters.UsuarioAdapter

```
public class UsuarioAdapter
extends androidx.recyclerview.widget.RecyclerView.Adapter<UsuarioAdapter.ViewHolder>
```

Clase que se encarga de manejar los adaptadores del usuario

Author:

Paspuel Mayra, Quistanchala Karla, Villarruel Michael

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
class	UsuarioAdapter.ViewHolder Metodo ViewHolder muestra de manera interactiva los datos del usuario

Constructor Summary

Constructors

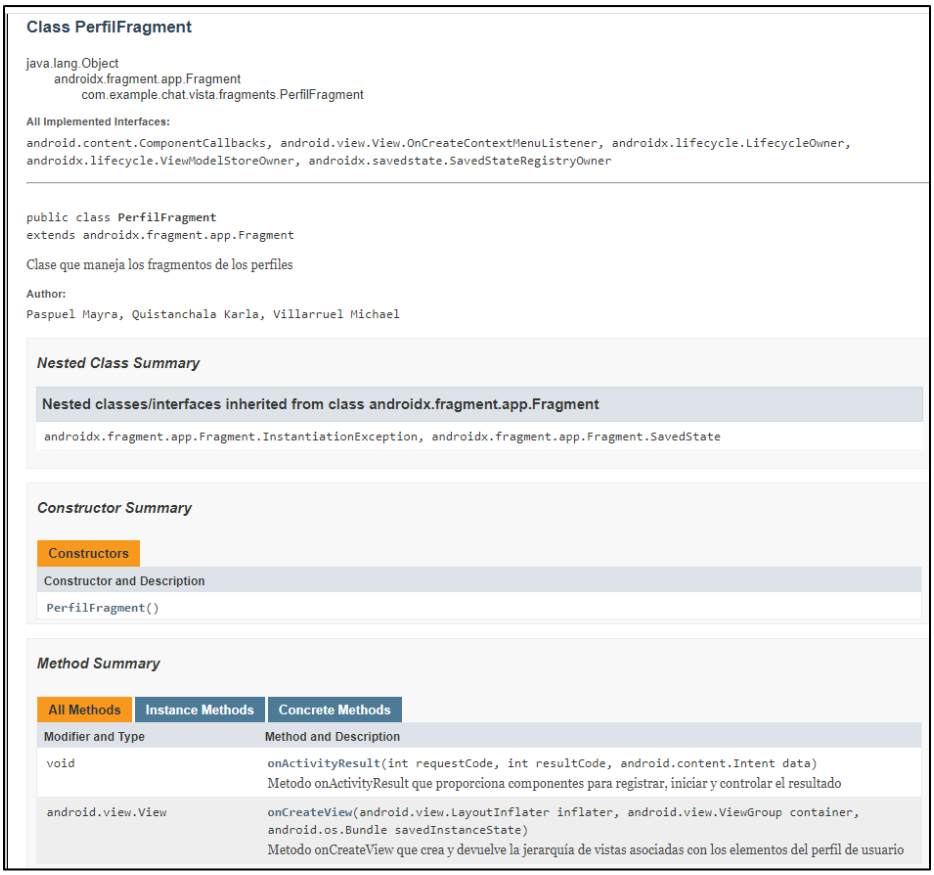
Constructor and Description
UsuarioAdapter(android.content.Context contexto, java.util.List<Usuario> usuarios) Constructor con parametros

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
android.content.Context	getContext() Metodo getContext que devuelve el contexto del adapter vinculado
int	getItemCount() Metodo getItemCount que devuelve el número de usuarios en el adaptador vinculado al RecyclerView padre.

- **Fragments Package**
  - **PerfilFragment**



Class PerfilFragment

java.lang.Object
  androidx.fragment.app.Fragment
    com.example.chat.vista.fragments.PerfilFragment

All Implemented Interfaces:

`android.content.ComponentCallbacks, android.view.View.OnCreateContextMenuListener, androidx.lifecycle.LifecycleOwner, androidx.lifecycle.ViewModelStoreOwner, androidx.savedstate.SavedStateRegistryOwner`

```
public class PerfilFragment
extends androidx.fragment.app.Fragment
```

Clase que maneja los fragmentos de los perfiles

Author:

`Paspuel Mayra, Quistanchala Karla, Villarruel Michael`

Nested Class Summary

Nested classes/interfaces inherited from class androidx.fragment.app.Fragment
`androidx.fragment.app.Fragment.InstantiationException, androidx.fragment.app.Fragment.SavedState`

Constructor Summary

Constructors

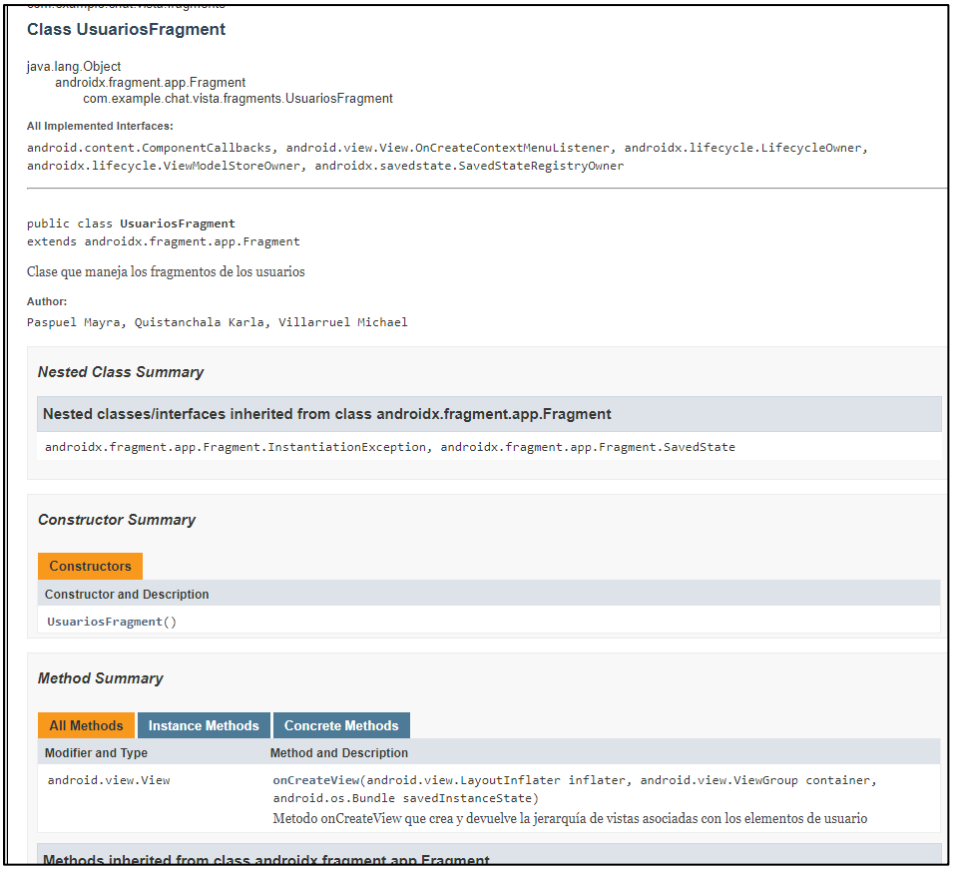
Constructor and Description
`PerfilFragment()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`void`	`onActivityResult(int requestCode, int resultCode, android.content.Intent data)` Metodo onActivityResult que proporciona componentes para registrar, iniciar y controlar el resultado
`android.view.View`	`onCreateView(android.view.LayoutInflater inflater, android.view.ViewGroup container, android.os.Bundle savedInstanceState)` Metodo onCreateView que crea y devuelve la jerarquía de vistas asociadas con los elementos del perfil de usuario

  - **UsuariosFragment**



Class UsuariosFragment

java.lang.Object
  androidx.fragment.app.Fragment
    com.example.chat.vista.fragments.UsuariosFragment

All Implemented Interfaces:

`android.content.ComponentCallbacks, android.view.View.OnCreateContextMenuListener, androidx.lifecycle.LifecycleOwner, androidx.lifecycle.ViewModelStoreOwner, androidx.savedstate.SavedStateRegistryOwner`

```
public class UsuariosFragment
extends androidx.fragment.app.Fragment
```

Clase que maneja los fragmentos de los usuarios

Author:

`Paspuel Mayra, Quistanchala Karla, Villarruel Michael`

Nested Class Summary

Nested classes/interfaces inherited from class androidx.fragment.app.Fragment
`androidx.fragment.app.Fragment.InstantiationException, androidx.fragment.app.Fragment.SavedState`

Constructor Summary

Constructors

Constructor and Description
`UsuariosFragment()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`android.view.View`	`onCreateView(android.view.LayoutInflater inflater, android.view.ViewGroup container, android.os.Bundle savedInstanceState)` Metodo onCreateView que crea y devuelve la jerarquía de vistas asociadas con los elementos de usuario

Methods inherited from class androidx.fragment.app.Fragment

5.5. Código

Modelo

- **Clase Conexión**

```java
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.modelo;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
/**
 * Clase que realiza la conexion a la base de datos.
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Conexion {

    private static Conexion instancia = null;

    private FirebaseAuth autenticacion;
    private FirebaseUser usuarioActual;
    private DatabaseReference baseDeDatos;
    private StorageReference almacenamiento;
    /**
     * Constructor
     */
    private Conexion() {
        this.autenticacion = FirebaseAuth.getInstance();
        this.usuarioActual = FirebaseAuth.getInstance().getCurrentUser();
        this.baseDeDatos = FirebaseDatabase.getInstance().getReference();
        this.almacenamiento = FirebaseStorage.getInstance().getReference();
    }

    /**
     * Metodo getInstancia inicializa la instancia
     * @return instancia.
     */
```

```java
    public static Conexion getInstancia() {
        instancia = new Conexion();
        return instancia;
    }
    /**
     * Metodo getAutenticación se autentica en la base de datos
     * @return autenticación en firebase.
     */
    public FirebaseAuth getAutenticacion() {
        return autenticacion;
    }

    /**
     * Metodo getAutenticación se autentica en la base de datos
     * @param autenticacion
     */
    public void setAutenticacion(FirebaseAuth autenticacion) {
        this.autenticacion = autenticacion;
    }

    /**
     * Metodo getUsuarioActual se verifica el usuario actual
     * @return usuarioActual.
     */
    public FirebaseUser getUsuarioActual() {
        return usuarioActual;
    }

    /**
     * Metodo setUsuarioActual se setea el usuario actual
     * @param usuarioActual
     */
    public void setUsuarioActual(FirebaseUser usuarioActual) {
        this.usuarioActual = usuarioActual;
    }

    /**
     * Metodo getBaseDeDatos devuelve la base de datos
     * @return baseDeDatos
     */
    public DatabaseReference getBaseDeDatos() {
        return baseDeDatos;
    }

    /**
     * Metodo setBaseDeDatos se setea la base de datos
     * @param baseDeDatos
     */
    public void setBaseDeDatos(DatabaseReference baseDeDatos) {
        this.baseDeDatos = baseDeDatos;
    }

    /**
     * Metodo getAlmacenamiento obtiene los datos almacenados
     * @return almacenamiento
```

```
     */
    public StorageReference getAlmacenamiento() {
        return almacenamiento;
    }

    /**
     * Metodo setAlmacenamiento se setea los datos para ser almacenados
     * @param almacenamiento
     */
    public void setAlmacenamiento(StorageReference almacenamiento) {
        this.almacenamiento = almacenamiento;
    }
}
```

- **Clase Mensaje**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.modelo;
/**
 * Clase que contiene los datos del mensaje
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Mensaje {

    private String emisor;
    private String receptor;
    private String contenido;
    private String hora;
    private String tipo;
    /**
     * Constructor vacio
     */
    public Mensaje() {
    }
    /**
     * Constructor con parametros
     * @param emisor
     * @param receptor
     * @param contenido
     * @param hora
     * @param tipo
```

```
     */
    public Mensaje(String emisor, String receptor, String contenido, String hora,
String tipo) {
        this.emisor = emisor;
        this.receptor = receptor;
        this.contenido = contenido;
        this.hora = hora;
        this.tipo = tipo;
    }
    /**
     * Metodo getEmisor donde se obtiene el emisor
     * @return emisor
     */
    public String getEmisor() {
        return emisor;
    }
    /**
     * Metodo setEmisor donde se setea el emisor
     * @param  emisor
     */
    public void setEmisor(String emisor) {
        this.emisor = emisor;
    }
    /**
     * Metodo getReceptor donde se obtiene el receptor
     * @return receptor
     */
    public String getReceptor() {
        return receptor;
    }
    /**
     * Metodo setReceptor donde se setea el receptor
     * @param  receptor
     */
    public void setReceptor(String receptor) {
        this.receptor = receptor;
    }
    /**
     * Metodo getContenido donde se obtiene el contenido del mensaje
     * @return contenido
     */
    public String getContenido() {
        return contenido;
    }
    /**
     * Metodo setContenido donde se setea el contenido del mensaje
     * @param  contenido
     */
    public void setContenido(String contenido) {
        this.contenido = contenido;
    }
    /**
     * Metodo getHora donde se obtiene la hora del mensaje
     * @return hora
     */
```

```
    public String getHora() {
        return hora;
    }
    /**
     * Metodo setHora donde se setea la hora del mensaje
     * @param hora
     */
    public void setHora(String hora) {
        this.hora = hora;
    }
    /**
     * Metodo getTipo donde se obtiene el emisor
     * @return tipo
     */
    public String getTipo() {
        return tipo;
    }
    /**
     * Metodo setTipo donde se obtiene el emisor
     * @param tipo
     */
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }
}
```

- **Clase Modelo**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.modelo;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.webkit.MimeTypeMap;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;
```

```java
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.chat.R;
import com.example.chat.vista.MainActivity;
import com.example.chat.vista.MensajeActivity;
import com.example.chat.vista.adapters.MensajeAdapter;
import com.example.chat.vista.adapters.UsuarioAdapter;
import com.google.android.gms.tasks.Continuation;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.StorageTask;
import com.google.firebase.storage.UploadTask;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;

import static android.content.Context.NOTIFICATION_SERVICE;
/**
 * Clase que contiene los metodos entre la base de datos y la aplicacion
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Modelo {

    Conexion conexion = Conexion.getInstancia();

    /**
     * Metodo login que realiza el inicio de sesion
     * @param context
     * @param txtContrasenia
     * @param txtEmail
     */
    public void login(final Context context, String txtEmail, String
txtContrasenia) {
        conexion.getAutenticacion().signInWithEmailAndPassword(txtEmail,
txtContrasenia).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Intent intent = new Intent(context, MainActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
                    context.startActivity(intent);
                } else {
                    Toast.makeText(context, "Datos Incorrectos",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    /**
     * Metodo salir que finaliza la sesion abierta
     */
    public void salir() {
        conexion.getAutenticacion().signOut();
    }

    /**
     * Metodo estaLogeado que verifica que se ha iniciado sesion
     */
    public boolean estaLogeado() {
        if (conexion.getUsuarioActual() != null) {
            return true;
        } else {
            return false;
        }
    }
    /**
     * Metodo enviarMensaje que se encarga de realizar el envio de mensajes
     * @param mensaje
     */
    public void enviarMensaje(Mensaje mensaje) {

        Calendar calendario = Calendar.getInstance();
        String hora = "" + calendario.get(Calendar.HOUR_OF_DAY);
        if (Integer.parseInt(hora) < 10) {
            hora = "0" + hora;
        }
        String minutos = "" + calendario.get(Calendar.MINUTE);
        if (Integer.parseInt(minutos) < 10) {
            minutos = "0" + minutos;
        }

        HashMap<String, Object> hashMap = new HashMap<>();
        hashMap.put("emisor", mensaje.getEmisor());
        hashMap.put("receptor", mensaje.getReceptor());
        hashMap.put("contenido", mensaje.getContenido());
        hashMap.put("hora", hora + ":" + minutos);
        hashMap.put("tipo", mensaje.getTipo());

        conexion.getBaseDeDatos().child("Mensajes").push().setValue(hashMap);

    }
    /**
     * Metodo registrar que registra un nuevo usuario
     * @param context
```

```
     * @param contrasenia
     * @param email
     * @param usuario
     */
    public void registrar(final Context context, final String usuario, String
email, String contrasenia) {
        conexion.getAutenticacion().createUserWithEmailAndPassword(email,
contrasenia).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    String idUsuario = conexion.getAutenticacion().getUid();
                    HashMap<String, String> hashMap = new HashMap<>();
                    hashMap.put("id", idUsuario);
                    hashMap.put("nombreUsuario", usuario);
                    hashMap.put("foto", "default");


conexion.getBaseDeDatos().child("Usuarios").child(idUsuario).setValue(hashMap).addO
nCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                Intent intent = new Intent(context,
MainActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                                context.startActivity(intent);
                            } else {
                                Toast.makeText(context, "No se puede registrar con
el email o contraseña ingresados", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });

                } else {
                    Toast.makeText(context, "No se puede registrar con el email o
contraseña ingresados", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
    /**
     * Metodo leerMensaje en el cual muestra los mensajes que se le han enviado al
usuario
     * @param context
     * @param recyclerView
     * @param usuarioId
     */
    public void leerMensajes(final RecyclerView recyclerView, final Context
context, final String usuarioId) {
        final ArrayList<Mensaje> mensajes = new ArrayList<>();
        final String miId = conexion.getAutenticacion().getUid();
        conexion.getBaseDeDatos().child("Mensajes").addValueEventListener(new
ValueEventListener() {
```

```
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                mensajes.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Mensaje mensaje = snapshot.getValue(Mensaje.class);
                    if (mensaje.getReceptor().equals(miId) &&
mensaje.getEmisor().equals(usuarioId) || mensaje.getReceptor().equals(usuarioId) &&
mensaje.getEmisor().equals(miId)) {
                        mensajes.add(mensaje);
                    }
                    MensajeAdapter messageAdapter = new MensajeAdapter(context,
mensajes);
                    recyclerView.setAdapter(messageAdapter);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }
    /**
     * Metodo cargarImagenEmisor que muestra la imagen al emisor
     * @param recyclerView
     * @param context
     * @param foto
     * @param nombreUsuario
     * @param userid
     */
    public void cargarImagenEmisor(final Context context, final String userid,
final RecyclerView recyclerView, final TextView nombreUsuario, final
CircleImageView foto) {


conexion.getBaseDeDatos().child("Usuarios").child(userid).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Usuario user = dataSnapshot.getValue(Usuario.class);
                nombreUsuario.setText(user.getNombreUsuario());
                if (user.getFoto().equals("default")) {
                    foto.setImageResource(R.mipmap.ic_launcher);
                } else {

Glide.with(context.getApplicationContext()).load(user.getFoto()).into(foto);
                }
                leerMensajes(recyclerView, context, userid);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
```

```
    }
    /**
     * Metodo cargarImagenUsuario que muestra la imagen al usuario
     * @param nombreUsuario
     * @param foto
     * @param context
     */
    public void cargarImagenUsuario(final Context context, final TextView
nombreUsuario, final CircleImageView foto) {


conexion.getBaseDeDatos().child("Usuarios").child(conexion.getUsuarioActual().getUi
d()).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Usuario usuario = dataSnapshot.getValue(Usuario.class);
                nombreUsuario.setText(usuario.getNombreUsuario());
                if (usuario.getFoto().equals("default")) {
                    foto.setImageResource(R.mipmap.ic_launcher);
                } else {

Glide.with(context.getApplicationContext()).load(usuario.getFoto()).into(foto);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });

    }
    /**
     * Metodo leerUsuarios muestra la lista de usuarios registrados
     * @param recyclerView
     * @param usuarioAdapter
     * @param usuarios
     */
    public void leerUsuarios(final List<Usuario> usuarios, final UsuarioAdapter
usuarioAdapter, final RecyclerView recyclerView) {

        conexion.getBaseDeDatos().child("Usuarios").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                usuarios.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Usuario user = snapshot.getValue(Usuario.class);
                    if (!user.getId().equals(conexion.getUsuarioActual().getUid()))
{
                        usuarios.add(user);
                    }
                }
                usuarioAdapter.setUsuarios(usuarios);
```

```
                recyclerView.setAdapter(usuarioAdapter);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });

    }
    /**
     * Metodo getExtensionArchivo obtiene la imagen que ha enviado
     * @param context
     * @param uri
     * @return uri
     */
    public String getExtensionArchivo(Uri uri, Context context) {
        ContentResolver contentResolver = context.getContentResolver();
        MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();
        return mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri));
    }
    /**
     * Metodo subirImagen que guarda en la base de datos la imagen enviada
     * @param context
     * @param imagenUri
     */
    public void subirImagen(Uri imagenUri, final Context context) {
        StorageTask storageTask;
        final StorageReference fileReference;
        final ProgressDialog progressDialog = new ProgressDialog(context);

        progressDialog.setMessage("Cargando...");
        progressDialog.show();

        if (imagenUri != null) {

            fileReference =
conexion.getAlmacenamiento().child("Archivos").child(conexion.getUsuarioActual().ge
tUid() + "." + getExtensionArchivo(imagenUri, context));
            storageTask = fileReference.putFile(imagenUri);

            storageTask.continueWithTask(new Continuation<UploadTask.TaskSnapshot,
Task<Uri>>() {
                @Override
                public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task)
throws Exception {
                    if (!task.isSuccessful()) {
                        throw task.getException();
                    }
                    return fileReference.getDownloadUrl();
                }
            }).addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    if (task.isSuccessful()) {
```

```
                        Uri downloadUri = task.getResult();
                        String uriBD = downloadUri.toString();
                        HashMap<String, Object> nuevaUriFoto = new HashMap<>();
                        nuevaUriFoto.put("foto", "" + uriBD);

conexion.getBaseDeDatos().child("Usuarios").child(conexion.getUsuarioActual().getUi
d()).updateChildren(nuevaUriFoto);
                    } else {
                        Toast.makeText(context, "Error!",
Toast.LENGTH_SHORT).show();
                    }
                    progressDialog.dismiss();

                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(context, e.getMessage(),
Toast.LENGTH_SHORT).show();
                    progressDialog.dismiss();
                }
            });
        } else {
            Toast.makeText(context, "No se ha seleccionado ninguna imagen",
Toast.LENGTH_SHORT).show();
        }
    }
    /**
     * Metodo idUsuarioActual que ubica el usuario que esta ingresando
     * @return usuarioId
     */
    public String idUsuarioActual() {
        return conexion.getUsuarioActual().getUid();
    }
    /**
     * Metodo enviarImagen que envia una imagen mediante el chat
     * @param context
     * @param imagenUri
     * @param receptor
     */
    public void enviarImagen(Uri imagenUri, final Context context, final String
receptor) {
        StorageTask storageTask;
        final StorageReference fileReference;
        final ProgressDialog progressDialog = new ProgressDialog(context);

        progressDialog.setMessage("Enviando...");
        progressDialog.show();

        if (imagenUri != null) {

            fileReference =
conexion.getAlmacenamiento().child("Archivos").child(System.currentTimeMillis() +
"." + getExtensionArchivo(imagenUri, context));
            storageTask = fileReference.putFile(imagenUri);
```

```
            storageTask.continueWithTask(new Continuation<UploadTask.TaskSnapshot,
Task<Uri>>() {
                @Override
                public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task)
throws Exception {
                    if (!task.isSuccessful()) {
                        throw task.getException();
                    }
                    return fileReference.getDownloadUrl();
                }
            }).addOnCompleteListener(new OnCompleteListener<Uri>() {
                @Override
                public void onComplete(@NonNull Task<Uri> task) {
                    if (task.isSuccessful()) {
                        Uri downloadUri = task.getResult();

                        Mensaje mensaje = new Mensaje();
                        mensaje.setEmisor(idUsuarioActual());
                        mensaje.setReceptor(receptor);
                        mensaje.setContenido(downloadUri.toString());
                        mensaje.setTipo("img");

                        enviarMensaje(mensaje);
                    } else {
                        Toast.makeText(context, "Error!",
Toast.LENGTH_SHORT).show();
                    }
                    progressDialog.dismiss();

                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(context, e.getMessage(),
Toast.LENGTH_SHORT).show();
                    progressDialog.dismiss();
                }
            });
        } else {
            Toast.makeText(context, "No se ha seleccionado ninguna imagen",
Toast.LENGTH_SHORT).show();
        }
    }
    int ban=0;
    /**
     * Metodo leer que muestra el char completo entre dos personas
     * @param context
     */
    public void leer(final Context context) {

        final ArrayList<Mensaje> mensajes = new ArrayList<>();

        conexion.getBaseDeDatos().child("Mensajes").addValueEventListener(new
ValueEventListener() {
```

```
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                mensajes.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Mensaje objMensaje = snapshot.getValue(Mensaje.class);
                    mensajes.add(objMensaje);
                }
                if(ban!=0) {
                    Mensaje miMensaje = null;
                    try {
                        miMensaje = mensajes.get(mensajes.size() - 1);
                        if (miMensaje != null &&
miMensaje.getReceptor().equals(conexion.getAutenticacion().getUid())) {
                            notificacion(context, miMensaje);
                        }
                    } catch (Exception ex) {
                    }
                }
                ban=1;
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }

    /**
     * Metodo notificacion que muestra una alerta cuando se recibe un mensaje
     * @param context
     * @param mensaje
     */
    public void notificacion(Context context, Mensaje mensaje) {
        NotificationCompat.Builder mBuilder;
        NotificationManager mNotifyMgr = (NotificationManager)
context.getSystemService(NOTIFICATION_SERVICE);
        int icono = R.mipmap.ic_launcher;

        Intent intent = new Intent(context, MensajeActivity.class);
        intent.putExtra("id", mensaje.getEmisor());

        PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);

        mBuilder = new NotificationCompat.Builder(context)
                .setContentIntent(pendingIntent)
                .setSmallIcon(icono)
                .setContentTitle("Nuevo Mensaje")
                .setContentText(mensaje.getContenido())
                .setVibrate(new long[]{100, 250, 100, 500})
                .setAutoCancel(true);
        mNotifyMgr.notify(1, mBuilder.build());
    }
}
```

- **Clase Usuario**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.modelo;
/**
 * Clase que contiene los datos del usuario
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Usuario {

    private String id;
    private String nombreUsuario;
    private String correo;
    private String contrasenia;
    private String foto;
    /**
     * Constructor vacio
     */
    public Usuario() {
    }
    /**
     * Constructor con parametros
     * @param foto
     * @param nombreUsuario
     * @param contrasenia
     * @param correo
     * @param id
     */
    public Usuario(String id, String nombreUsuario, String correo, String
contrasenia, String foto) {
        this.id = id;
        this.nombreUsuario = nombreUsuario;
        this.correo = correo;
        this.contrasenia = contrasenia;
        this.foto = foto;
    }
    /**
     * Metodo getId que obtiene el id del usuario
     * @return id
     */
```

```java
    public String getId() {
        return id;
    }
    /**
     * Metodo setId que setea el id del usuario
     * @param id
     */
    public void setId(String id) {
        this.id = id;
    }

    /**
     * Metodo getNombreUsuario que obtiene el nombre del usuario
     * @return nombreUsuario
     */
    public String getNombreUsuario() {
        return nombreUsuario;
    }
    /**
     * Metodo setNombreUsuario que setea el nombre del usuario
     * @param nombreUsuario
     */
    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;
    }
    /**
     * Metodo getCorreo que obtiene el correo del usuario
     * @return correo
     */
    public String getCorreo() {
        return correo;
    }
    /**
     * Metodo setCorreo que setea el correo del usuario
     * @param correo
     */
    public void setCorreo(String correo) {
        this.correo = correo;
    }
    /**
     * Metodo getContrasenia que obtiene la contrasenia del usuario
     * @return contrasenia
     */
    public String getContrasenia() {
        return contrasenia;
    }
    /**
     * Metodo setContrasenia que setea la contrasenia del usuario
     * @param contrasenia
     */
    public void setContrasenia(String contrasenia) {
        this.contrasenia = contrasenia;
    }
    /**
     * Metodo getFoto que obtiene la foto del usuario
```

```
     * @return foto
     */
    public String getFoto() {
        return foto;
    }
    /**
     * Metodo setFoto que setea la foto del usuario
     * @param foto
     */
    public void setFoto(String foto) {
        this.foto = foto;
    }
}
```

Presentador

- **Clase Presentador**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.presentador;

import android.content.Context;
import android.net.Uri;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.example.chat.modelo.Mensaje;
import com.example.chat.modelo.Modelo;
import com.example.chat.modelo.Usuario;
import com.example.chat.vista.adapters.UsuarioAdapter;

import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;

/**
 * Clase que contiene los metodos intermediarios entre el modelo y la vista
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
```

```java
public class Presentador {

    Modelo modelo = new Modelo();

    /**
     * Metodo login que realiza el inicio de sesion
     *
     * @param context
     * @param txtContrasenia
     * @param txtEmail
     */
    public boolean login(Context context, String txtEmail, String txtContrasenia) {
        try {
            modelo.login(context, txtEmail, txtContrasenia);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo salir que finaliza la sesion abierta
     */
    public boolean salir() {
        try {
            modelo.salir();
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo estaLogeado que verifica que se ha iniciado sesion
     */
    public boolean estaLogeado() {
        try {
            return modelo.estaLogeado();
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo enviarMensaje que se encarga de realizar el envio de mensajes
     *
     * @param mensaje
     */
    public boolean enviarMensaje(Mensaje mensaje) {
        try {
            modelo.enviarMensaje(mensaje);
            return true;
        } catch (Exception ex) {
            return false;
        }
```

```
    }

    /**
     * Metodo registrar que registra un nuevo usuario
     *
     * @param context
     * @param contrasenia
     * @param email
     * @param usuario
     */
    public boolean registrar(Context context, String usuario, String email, String
contrasenia) {
        try {
            modelo.registrar(context, usuario, email, contrasenia);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo leerMensaje en el cual muestra los mensajes que se le han enviado al
usuario
     *
     * @param context
     * @param recyclerView
     * @param usuarioId
     */
    public boolean leerMensajes(RecyclerView recyclerView, Context context, String
usuarioId) {
        try {
            modelo.leerMensajes(recyclerView, context, usuarioId);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo cargarImagenEmisor que muestra la imagen al emisor
     *
     * @param recyclerView
     * @param context
     * @param foto
     * @param nombreUsuario
     * @param userid
     */
    public boolean cargarImagenEmisor(Context context, String userid, RecyclerView
recyclerView, TextView nombreUsuario, CircleImageView foto) {
        try {
            modelo.cargarImagenEmisor(context, userid, recyclerView, nombreUsuario,
foto);
            return true;
        } catch (Exception ex) {
```

```
            return false;
        }
    }

    /**
     * Metodo cargarImagenUsuario que muestra la imagen al usuario
     *
     * @param nombreUsuario
     * @param foto
     * @param context
     */
    public boolean cargarImagenUsuario(Context context, TextView nombreUsuario,
CircleImageView foto) {

        try {
            modelo.cargarImagenUsuario(context, nombreUsuario, foto);
            return true;
        } catch (Exception ex) {
            return false;
        }

    }

    /**
     * Metodo leerUsuarios muestra la lista de usuarios registrados
     *
     * @param recyclerView
     * @param usuarioAdapter
     * @param usuarios
     */
    public boolean leerUsuarios(List<Usuario> usuarios, UsuarioAdapter
usuarioAdapter, RecyclerView recyclerView) {

        try {
            modelo.leerUsuarios(usuarios, usuarioAdapter, recyclerView);
            return true;
        } catch (Exception ex) {
            return false;
        }

    }

    /**
     * Metodo subirImagen que guarda en la base de datos la imagen enviada
     *
     * @param context
     * @param imagenUri
     */
    public boolean subirImagen(Uri imagenUri, Context context) {
        try {
            modelo.subirImagen(imagenUri, context);
            return true;
        } catch (Exception ex) {
            return false;
        }
```

```
    }

    /**
     * Metodo idUsuarioActual que ubica el usuario que esta ingresando
     */
    public String idUsuarioActual() {
        try {
            return modelo.idUsuarioActual();
        } catch (Exception ex) {
            return null;
        }
    }

    /**
     * Metodo enviarImagen que envia una imagen mediante el chat
     *
     * @param context
     * @param imagenUri
     * @param receptor
     */
    public boolean enviarImagen(Uri imagenUri, Context context, String receptor) {
        try {
            modelo.enviarImagen(imagenUri, context, receptor);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

    /**
     * Metodo leer que muestra el char completo entre dos personas
     *
     * @param context
     */
    public boolean leer(Context context) {
        try {
            modelo.leer(context);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

}
```

Vista

- **Clase MensajeAdapter**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: Chat
```

```java
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
 * Los contenidos de este archivo son propiedad privada y estan protegidos por
 * la licencia BSD
 *
 * Se puede utilizar, reproducir o copiar el contenido de este archivo.
 */
package com.example.chat.vista.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.chat.R;
import com.example.chat.modelo.Mensaje;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.List;
/**
 * Clase que adapta el mensaje
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class MensajeAdapter extends RecyclerView.Adapter<MensajeAdapter.ViewHolder>
{

    public static final int MSG_IZ = 0;
    public static final int MSG_DER = 1;
    private Context contexto;
    private List<Mensaje> mensajes;
    FirebaseUser fuser;
    /**
     * Constructor con parametros
     * @param chats
     * @param contexto
     */
    public MensajeAdapter(Context contexto, List<Mensaje> chats){
        this.mensajes = chats;
        this.contexto = contexto;
    }
    /**
     * Metodo onCreateViewHolder que crea un marcador de vista para cada elemento
     * @param parent
     * @param viewType
```

```
     */
    @NonNull
    @Override
    public MensajeAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        if (viewType == MSG_DER) {
            View view =
LayoutInflater.from(contexto).inflate(R.layout.chat_item_der, parent, false);
            return new MensajeAdapter.ViewHolder(view);
        } else {
            View view =
LayoutInflater.from(contexto).inflate(R.layout.chat_item_izq, parent, false);
            return new MensajeAdapter.ViewHolder(view);
        }
    }
    /**
     * Metodo onBindViewHolder obtiene nuevos titulares de vista
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull MensajeAdapter.ViewHolder holder, int
position) {

        Mensaje chat = mensajes.get(position);
        if (chat.getTipo().equals("img")){
            holder.mostrarMensaje.setVisibility(View.INVISIBLE);

Glide.with(contexto).load(chat.getContenido()).into(holder.imagenMensaje);
            holder.imagenMensaje.setAdjustViewBounds(true);
        }else {
            holder.mostrarMensaje.setText(chat.getContenido());
        }
        holder.hora.setText(chat.getHora());
    }
    /**
     * Metodo getItemCount que devuelve el número de elementos en el adaptador
vinculado al RecyclerView padre.
     * @return numero de mensajes
     */
    @Override
    public int getItemCount() {
        return mensajes.size();
    }
    /**
     * Metodo ViewHolder muestra de manera interactiva la ventana de chat
     */
    public  class ViewHolder extends RecyclerView.ViewHolder{

        public TextView mostrarMensaje;
        public TextView hora;
        public ImageView imagenMensaje;

        public ViewHolder(View itemView) {
            super(itemView);
```

```
            mostrarMensaje = itemView.findViewById(R.id.txtMensaje);
            hora = itemView.findViewById(R.id.txtHora);
            imagenMensaje=itemView.findViewById(R.id.imagenMensaje);
        }
    }
    /**
     * Metodo getItemViewType obtiene la información para ser mostrada
     * @param position
     * @return mensajes
     */
    @Override
    public int getItemViewType(int position) {
        fuser = FirebaseAuth.getInstance().getCurrentUser();
        if (mensajes.get(position).getEmisor().equals(fuser.getUid())){
            return MSG_DER;
        } else {
            return MSG_IZ;
        }
    }
}
```

- **Clase UsuarioAdapter**

```
package com.example.chat.vista.adapters;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.example.chat.vista.MensajeActivity;
import com.example.chat.R;
import com.example.chat.modelo.Usuario;

import java.util.List;

/**
 * Clase que se encarga de manejar los adaptadores del usuario
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class UsuarioAdapter extends RecyclerView.Adapter<UsuarioAdapter.ViewHolder>
{
```

```
    private Context context;
    private List<Usuario> usuarios;


    /**
     * Constructor con parametros
     * @param contexto
     * @param usuarios
     */
    public UsuarioAdapter(Context contexto, List<Usuario> usuarios){
        this.usuarios = usuarios;
        this.context = contexto;
    }

    /**
     * Metodo getContext que devuelve el contexto del adapter vinculado
     * @return contexto tipo Context
     */
    public Context getContext() {
        return context;
    }

    /**
     * Metodo setContext que que crea las vistas y adaptadores
     * @param context
     */
    public void setContext(Context context) {
        this.context = context;
    }

    /**
     * Metodo gestUsuarios que recupera los usuarios registrados
     * @return lista de usuarios
     */
    public List<Usuario> getUsuarios() {
        return usuarios;
    }

    /**
     * Metodo setUsuarios que
     * @param usuarios publica los usuarios registrados
     */
    public void setUsuarios(List<Usuario> usuarios) {
        this.usuarios = usuarios;
    }


    /**
     * Metodo onCreateViewHolder que crea un marcador de vista para cada usuario
     * @param parent
     * @param viewType
     */
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
```

```
        View view = LayoutInflater.from(context).inflate(R.layout.usuario_item,
parent, false);
        return new UsuarioAdapter.ViewHolder(view);
    }

    /**
     * Metodo onBindViewHolder obtiene usuarios con la actividad registrada
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        final Usuario usuario = usuarios.get(position);
        holder.nombreUsuario.setText(usuario.getNombreUsuario());

        if(usuario.getFoto().equals("default")){
            holder.foto.setImageResource(R.mipmap.ic_launcher);
        }else{
            Glide.with(context).load(usuario.getFoto()).into(holder.foto);
        }

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(context, MensajeActivity.class);
                intent.putExtra("id",usuario.getId());
                context.startActivity(intent);
            }
        });

    }

    /**
     * Metodo getItemCount que devuelve el número de usuarios en el adaptador
vinculado al RecyclerView padre.
     * @return numero de usuarios
     */
    @Override
    public int getItemCount() {
        return usuarios.size();
    }


    /**
     * Metodo ViewHolder muestra de manera interactiva los datos del usuario
     */
    public  class ViewHolder extends RecyclerView.ViewHolder{

        public TextView nombreUsuario;
        public ImageView foto;

        public ViewHolder(View itemView) {
            super(itemView);

            nombreUsuario = itemView.findViewById(R.id.txtNombreUsuario);
```

```
            foto = itemView.findViewById(R.id.imgFoto);
        }
    }

}
```

- **Clase PerfilFragment**

```
package com.example.chat.vista.fragments;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import com.example.chat.R;
import com.example.chat.presentador.Presentador;

import de.hdodenhof.circleimageview.CircleImageView;

import static android.app.Activity.RESULT_OK;

/**
 * Clase que maneja los fragmentos de los perfiles
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class PerfilFragment extends Fragment {

    CircleImageView foto;
    TextView nombreUsuario;
    Presentador presentador = new Presentador();

    private static final int IMAGE_REQUEST = 1;
    private Uri imagenUri;

    /**
     * Metodo onCreateView que crea y devuelve la jerarquía de vistas asociadas con
los elementos del perfil de usuario
     * @param inflater
     * @param container
     * @param savedInstanceState
     */
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_perfil, container, false);
```

```
        foto = view.findViewById(R.id.imgFoto);
        nombreUsuario = view.findViewById(R.id.txtNombreUsuario);
        presentador.cargarImagenUsuario(getContext(),nombreUsuario,foto);
        foto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                abrirImagenes();
            }
        });

        return view;
    }

    /**
     * Metodo abrirImagenes que permite abrir imagenes para el envio de contenido
     */
    private void abrirImagenes() {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, IMAGE_REQUEST);
    }

    /**
     * Metodo onActivityResult que proporciona componentes para registrar, iniciar
y controlar el resultado
     * @param requestCode
     * @param resultCode
     * @param data
     */
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == IMAGE_REQUEST && resultCode == RESULT_OK && data != null
&& data.getData() != null) {
            imagenUri = data.getData();
            presentador.subirImagen(imagenUri, getContext());
        }
    }
}
```

- **Clase UsuariosFragment**

```
package com.example.chat.vista.fragments;

import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
```

```java
import android.view.ViewGroup;

import com.example.chat.presentador.Presentador;
import com.example.chat.vista.adapters.UsuarioAdapter;
import com.example.chat.R;
import com.example.chat.modelo.Usuario;

import java.util.ArrayList;
import java.util.List;
/**
 * Clase que maneja los fragmentos de los usuarios
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class UsuariosFragment extends Fragment {

    private RecyclerView recyclerView;
    private UsuarioAdapter usuarioAdapter;
    private List<Usuario> usuarios;
    private Presentador presentador = new Presentador();

    /**
     * Metodo onCreateView que crea y devuelve la jerarquía de vistas asociadas con
los elementos de usuario
     * @param inflater
     * @param container
     * @param savedInstanceState
     */
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_usuarios,container,false);

        recyclerView = view.findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        usuarios = new ArrayList<>();
        usuarioAdapter = new UsuarioAdapter(getContext(),usuarios);

        presentador.leerUsuarios(usuarios, usuarioAdapter, recyclerView);
        return view;

    }
}
```

- **Clase LoginActivity**

```java
package com.example.chat.vista;

import androidx.appcompat.app.AppCompatActivity;
```

```java
import androidx.appcompat.widget.Toolbar;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.example.chat.R;
import com.example.chat.presentador.Presentador;
import com.google.firebase.auth.FirebaseAuth;
import com.rengwuxian.materialedittext.MaterialEditText;
/**
 * Clase que contiene las propiedades de la vista de Login
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class LoginActivity extends AppCompatActivity {

    MaterialEditText correo, contrasenia;
    Button btnIngresar;

    FirebaseAuth auth;
    Presentador presentador = new Presentador();

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Ingreso");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        auth = FirebaseAuth.getInstance();

        correo = findViewById(R.id.txtCorreo);
        contrasenia = findViewById(R.id.txtContrasenia);
        btnIngresar = findViewById(R.id.btnIngresar);

        btnIngresar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String txtEmail = correo.getText().toString();
                String txtContrasenia = contrasenia.getText().toString();
                if (txtEmail.isEmpty() || txtContrasenia.isEmpty()){
                    Toast.makeText(LoginActivity.this, "Todos los campos deben ser
llenados correctamente", Toast.LENGTH_SHORT).show();
                } else {
```

```
                    presentador.login(LoginActivity.this,txtEmail,txtContrasenia);
                }
            }
        });
    }
}
```

- **Clase MainActivity**

```
package com.example.chat.vista;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.viewpager.widget.ViewPager;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

import com.example.chat.R;
import com.example.chat.presentador.Presentador;
import com.example.chat.vista.fragments.PerfilFragment;
import com.example.chat.vista.fragments.UsuariosFragment;
import com.google.android.material.tabs.TabLayout;

import java.util.ArrayList;

import de.hdodenhof.circleimageview.CircleImageView;
/**
 * Clase que contiene las propiedades de la vista principal
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class MainActivity extends AppCompatActivity {

    CircleImageView foto;
    TextView nombreUsuario;
    ViewPager viewPager;
    TabLayout tabLayout;
    Presentador presentador = new Presentador();

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
principal
     * @param savedInstanceState
```

```
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        presentador.leer(MainActivity.this);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("");

        foto = findViewById(R.id.imgFoto);
        nombreUsuario = findViewById(R.id.txtNombreUsuario);
        tabLayout = findViewById(R.id.tabLayout);
        viewPager = findViewById(R.id.viewPager);

        presentador.cargarImagenUsuario(MainActivity.this, nombreUsuario, foto);

        ViewPagerAdapter viewPagerAdapter = new
ViewPagerAdapter(getSupportFragmentManager());
        viewPagerAdapter.addFragment(new UsuariosFragment(), "Usuarios");
        viewPagerAdapter.addFragment(new PerfilFragment(), "Perfil");

        viewPager.setAdapter(viewPagerAdapter);
        tabLayout.setupWithViewPager(viewPager);

    }

    /**
     * Metodo onCreateOptionsMenu que permite mostrar el menu de opciones de las
actividades
     * @param menu
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

    /**
     * Metodo onOptionsItemSelected que permite identificar al elemento
seleccionado dentro del menú
     * @param item
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.btnSalir:
                presentador.salir();
                startActivity(new Intent(MainActivity.this,
StartActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
                return true;
        }
```

```
        return false;
    }

    /**
     * Clase que contiene los adaptadores de la vista
     */

    class ViewPagerAdapter extends FragmentPagerAdapter {

        private ArrayList<Fragment> fragmentos;
        private ArrayList<String> titulos;

        /**
         * Constructor con parametros
         * @param fm
         */
        ViewPagerAdapter(FragmentManager fm) {
            super(fm);
            this.fragmentos = new ArrayList<>();
            this.titulos = new ArrayList<>();
        }

        /**
         * Metodo getItem que permite recuperar la posición un elemento
seleccionado
         * @param position
         */
        @NonNull
        @Override
        public Fragment getItem(int position) {
            return fragmentos.get(position);
        }

        /**
         * Metodo getCount que obtiene el tamaño del fragmento
         * @return el tamaño del fragmento
         */
        @Override
        public int getCount() {
            return fragmentos.size();
        }

        /**
         * Metodo addFragmen que permite agregar un nuevo fragmento
         * @param fragmento
         * @param titulo
         */
        public void addFragment(Fragment fragmento, String titulo) {
            fragmentos.add(fragmento);
            titulos.add(titulo);
        }

        /**
         * Metodo addFragmen que devuelve un caracter en una position especifa
         * @param position
```

```
         */
        //Ctrl + O
        @Nullable
        @Override
        public CharSequence getPageTitle(int position) {
            return titulos.get(position);
        }
    }

}
```

- **Clase MensajeActivity**

```
package com.example.chat.vista;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import com.example.chat.R;
import com.example.chat.modelo.Mensaje;
import com.example.chat.presentador.Presentador;

import de.hdodenhof.circleimageview.CircleImageView;

/**
 * Clase que contiene las propiedades de la vista del mensaje
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class MensajeActivity extends AppCompatActivity {

    CircleImageView foto;
    TextView nombreUsuario;

    private static final int IMAGE_REQUEST = 1;
    private Uri imagenUri;

    ImageButton btnEnviar, btnEnviarImagen;
    EditText txtEnviar;

    RecyclerView recyclerView;
```

```
    Presentador presentador = new Presentador();
    String userid;

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
del mensaje
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mensaje);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(getApplicationContext());
        linearLayoutManager.setStackFromEnd(true);
        recyclerView.setLayoutManager(linearLayoutManager);

        foto = findViewById(R.id.imgFoto);
        nombreUsuario = findViewById(R.id.txtNombreUsuario);
        btnEnviar = findViewById(R.id.btnEnviar);
        btnEnviarImagen = findViewById(R.id.btnEnviarImagen);
        txtEnviar = findViewById(R.id.txtEnviar);

        Intent intent = getIntent();
        userid = intent.getStringExtra("id");

        presentador.cargarImagenEmisor(MensajeActivity.this, userid, recyclerView,
nombreUsuario, foto);

        btnEnviar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Mensaje mensaje = new Mensaje();
                mensaje.setEmisor(presentador.idUsuarioActual());
                mensaje.setReceptor(userid);
                mensaje.setContenido(txtEnviar.getText().toString());
                mensaje.setTipo("txt");
                if (!txtEnviar.getText().toString().equals("")) {
                    presentador.enviarMensaje(mensaje);
                } else {
                    Toast.makeText(MensajeActivity.this, "No se puede enviar un
```

```
mensaje vacío", Toast.LENGTH_SHORT).show();
                }
                txtEnviar.setText("");
            }
        });

        btnEnviarImagen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                abrirImagenes();
            }
        });

    }

    /**
     * Metodo abrirImagenes que permite abrir imagenes en la recepción de un
mensaje
     */
    private void abrirImagenes() {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, IMAGE_REQUEST);
    }

    /**
     * Metodo onActivityResult que permite volver a una actividad del chat luego de
abrir una imagen desde la galeria o la cámara
     * @param requestCode
     * @param  resultCode
     * @param data
     */
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == IMAGE_REQUEST && resultCode == RESULT_OK && data != null
&& data.getData() != null) {
            imagenUri = data.getData();
            presentador.enviarImagen(imagenUri, MensajeActivity.this, userid);
        }
    }
}
```

- **Clase RegistroActivity**

```
package com.example.chat.vista;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```java
import android.widget.Toast;

import com.example.chat.R;
import com.example.chat.presentador.Presentador;
import com.rengwuxian.materialedittext.MaterialEditText;
/**
 * Clase que contiene las propiedades de la vista de registro de actividad
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class RegistroActivity extends AppCompatActivity {

    MaterialEditText usuario, correo, contrasenia;
    Button registrarse;
    Presentador presentador = new Presentador();

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
de registro de actividad
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Registro");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        usuario = findViewById(R.id.txtNombreUsuario);
        correo = findViewById(R.id.txtCorreo);
        contrasenia = findViewById(R.id.txtContrasenia);
        registrarse = findViewById(R.id.btnRegistrarse);

        registrarse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String txtUsuario = usuario.getText().toString();
                String txtEmail = correo.getText().toString();
                String txtContrasenia = contrasenia.getText().toString();


if(txtUsuario.isEmpty()||txtEmail.isEmpty()||txtContrasenia.isEmpty()) {
                    Toast.makeText(RegistroActivity.this, "Todos los campos deben
ser llenados correctamente", Toast.LENGTH_SHORT).show();
                }else if(txtContrasenia.length() < 8){
                    Toast.makeText(RegistroActivity.this, "La contraseña debe tener
almenos 8 caracteres", Toast.LENGTH_SHORT).show();
                }else {

presentador.registrar(RegistroActivity.this,txtUsuario,txtEmail,txtContrasenia);
```

```
                }
            }
        });
    }

}
```

- **Clase StartActivity**

```
package com.example.chat.vista;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.example.chat.R;
import com.example.chat.presentador.Presentador;
import com.rengwuxian.materialedittext.MaterialEditText;
/**
 * Clase que contiene las propiedades de la vista de registro de actividad
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class RegistroActivity extends AppCompatActivity {

    MaterialEditText usuario, correo, contrasenia;
    Button registrarse;
    Presentador presentador = new Presentador();

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
de registro de actividad
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Registro");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        usuario = findViewById(R.id.txtNombreUsuario);
        correo = findViewById(R.id.txtCorreo);
        contrasenia = findViewById(R.id.txtContrasenia);
        registrarse = findViewById(R.id.btnRegistrarse);
```

```
        registrarse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String txtUsuario = usuario.getText().toString();
                String txtEmail = correo.getText().toString();
                String txtContrasenia = contrasenia.getText().toString();


if(txtUsuario.isEmpty()||txtEmail.isEmpty()||txtContrasenia.isEmpty()) {
                    Toast.makeText(RegistroActivity.this, "Todos los campos deben
ser llenados correctamente", Toast.LENGTH_SHORT).show();
                }else if(txtContrasenia.length() < 8){
                    Toast.makeText(RegistroActivity.this, "La contraseña debe tener
almenos 8 caracteres", Toast.LENGTH_SHORT).show();
                }else {

presentador.registrar(RegistroActivity.this,txtUsuario,txtEmail,txtContrasenia);
                }
            }
        });
    }

}
```

6. Referencias Bibliográficas

Rouse, M. (2019). Native Code. Retrieved from SearchAppArchitecture: https://searchapparchitecture.techtarget.com/definition/native-code#:~:text=Native%20code%20is%20computer%20programming,computer%20emulates%20the%20original%20processor.

Samusko, B. (n.d.). “Model-View-Presenter: Our Choice of Architecture for Your Android App”. Retrieved from SteelWiki: https://steelkiwi.com/blog/model-view-presenter-our-choice-of-android-app/