DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERIA EN SISTEMAS E INFORMÁTICA

PROGRAMACIÓN MOVIL

TIENDA VIRTUAL

INTEGRANTES:

PASPUEL YÁNEZ MAYRA ALEXANDRA

QUISTANCHALA SUNTAXI KARLA DANIELA

VILLARRUEL ARCINIEGA MICHAEL ALEJANDRO

NRC:

6112

08 DE SEPTIEMBRE 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 comercio electrónico ha tenido que actualizarse para ofrecer lo mejor en aplicaciones móviles adaptadas a las necesidades de sus usuarios. La tienda virtual es un medio para que los clientes puedan acceder a los productos fabricados o distribuidos por las empresas sean del tamaño que sean. Las tiendas virtuales estás estructuradas para que en la sección de cada negocio, cada vendedor pueda subir sus propios artículos de forma muy rápida y sencilla sin costo alguno. La idea es que cualquier persona con informática básica pueda hacerlo.

2. Objetivo

- Construir una aplicación móvil que permita emular una tienda virtual para la compra y venta de productos, además de poder contactar directamente al proveedor a través de un chat interno.

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

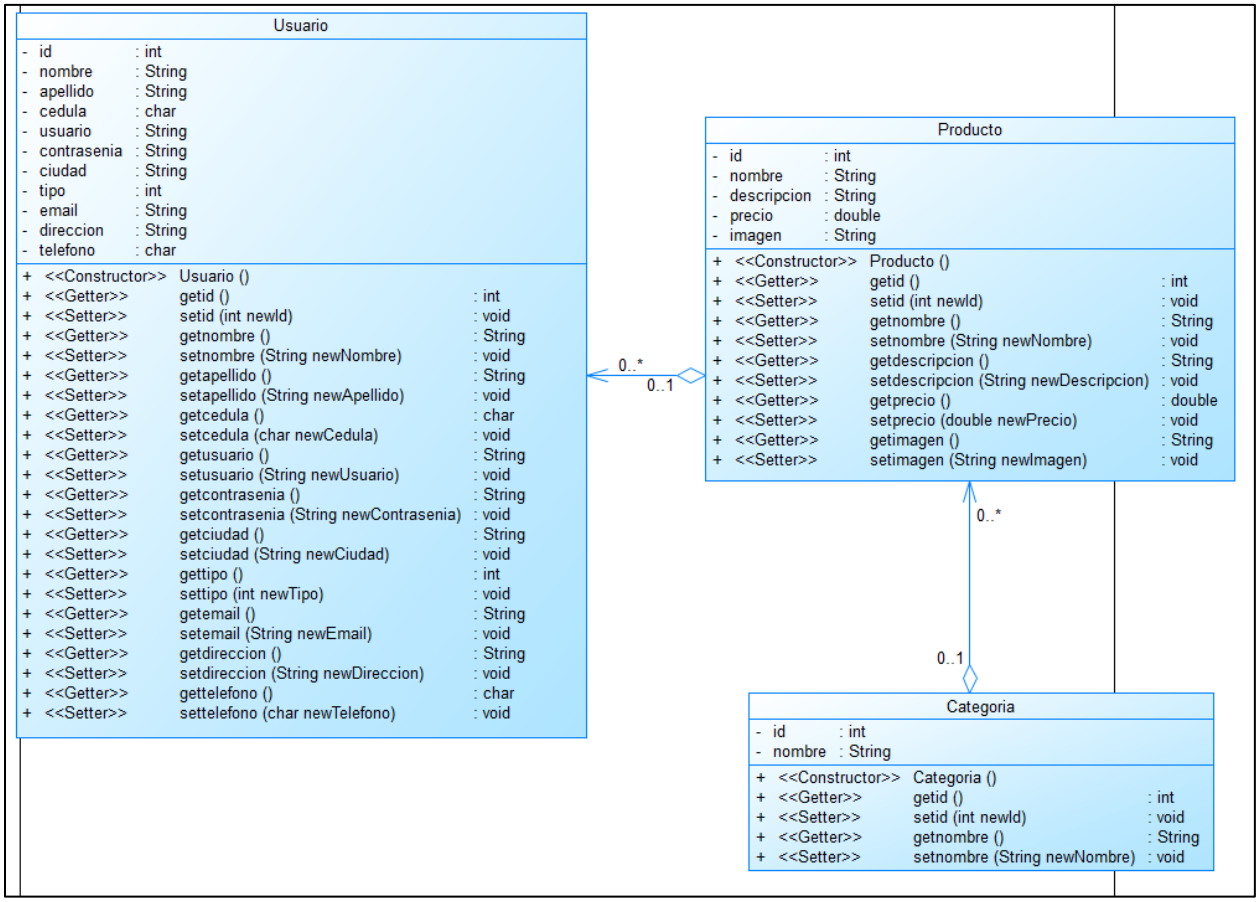
El SDK de Android incluye todas las bibliotecas y archivos necesarios para que los desarrolladores de Android puedan comenzar. Lo bueno de Android Studio es que el SDK está integrado y es de fácil acceso simplemente haciendo clic en un botón en la barra de herramientas superior. Los elementos junto al 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 tienda virtual, además de que es una aplicación totalmente intuitiva, permite registrar productos de todo tipo y categoría donde los usuarios pueden vender y comprar.
- El aplicativo permite establecer una comunicación entre clientes y proveedores en todo momento además de enviar su localización para coordinar el envio o entrega de los productos.

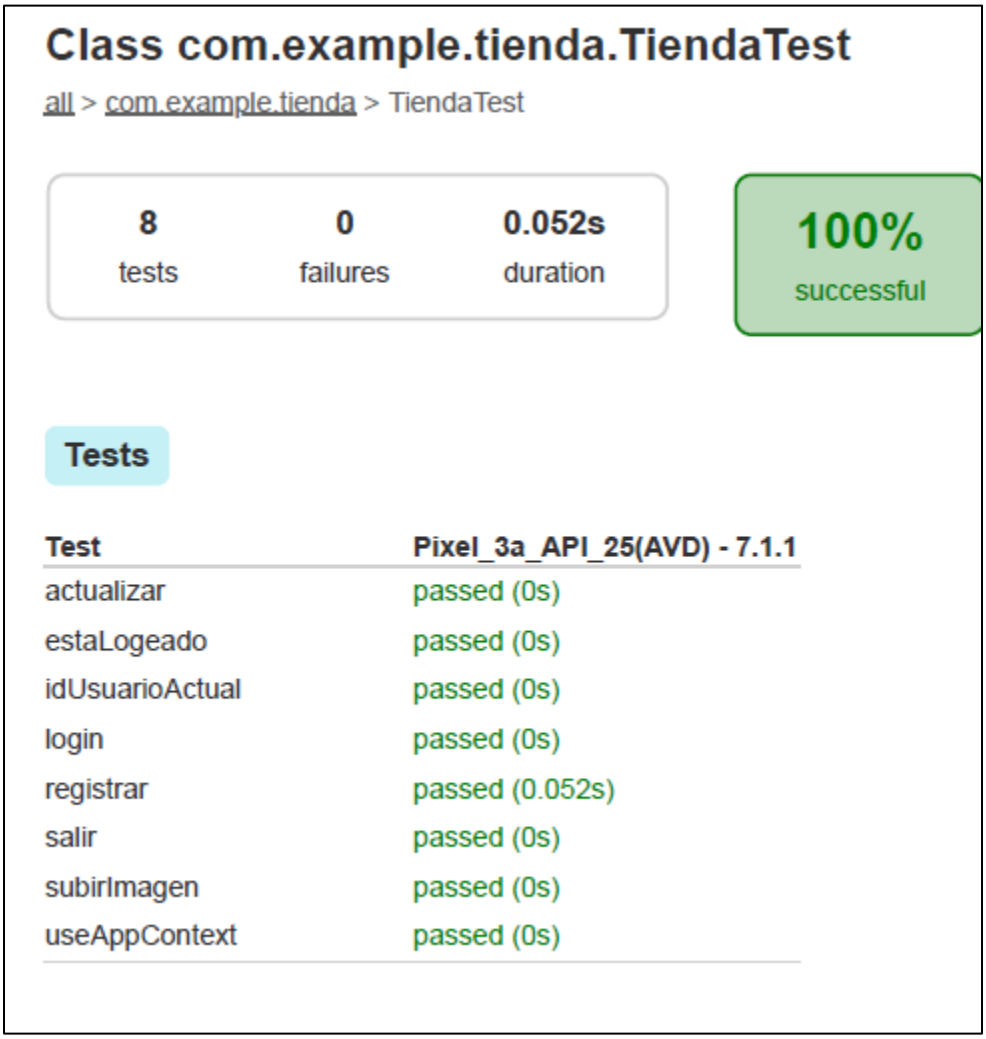
5. Anexos

5.1. Modelo



5.2. Test

5.2.1. Reporte



Class com.example.tienda.TiendaTest

all > com.example.tienda > TiendaTest

8	0	0.052s
tests	failures	duration

100% successful

Tests

Test	Pixel_3a_API_25(AVD) - 7.1.1
actualizar	passed (0s)
estaLogeado	passed (0s)
idUsuarioActual	passed (0s)
login	passed (0s)
registrar	passed (0.052s)
salir	passed (0s)
subirImagen	passed (0s)
useAppContext	passed (0s)

5.2.2. Código

- **Clase TiendaTest**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda;
```

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

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

import com.example.tienda.modelo.Modelo;
import com.example.tienda.modelo.Producto;
import com.example.tienda.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 TiendaTest {
    private Presentador presentador = new Presentador();
    private Modelo modelo = new Modelo();

    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext =
InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals("com.example.tienda", 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","12345678"));
    }

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

    @Test
    public void idUsuarioActual() {
        assertEquals(null,presentador.idUsuarioActual());
    }

    @Test
    public void actualizar() {
        String mensaje="";
        try {
            modelo.actualizar(new Producto("1", "TV", "45", "TV", "https://q-cf.bstatic.com/images/hotel/max500/161/161016875.jpg",
"KCiPL01cAUhkZTNaoO7RJjatCOz2", "Hogar"));
        }catch (Exception ex) {
            mensaje = ex.getMessage();
        }

        assertEquals("",mensaje);
    }

}
```

5.3. Código

Modelo

- **Clase Categoria**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.modelo;

/**
 * Clase que contiene los datos de las categorias del producto
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Categoria {

    private String nombre;

    /**
     * Constructor vacio
     */
    public Categoria() {
    }

    /**
     * Constructor con parametros
     * @param nombre
     */
    public Categoria(String nombre) {
        this.nombre = nombre;
    }

    /**
     * Metodo getNombre que obtiene el nombre de la categoría
     * @return nombre
     */
    public String getNombre() {
        return nombre;
    }
    /**
     * Metodo setNombre que setea el nombre de la categoría
     * @param nombre
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

}
```

- **Clase Conexión**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * Creado 23/07/2020
 * Modificado 02/08/2020
 *
```

```java
 * 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.tienda.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.
     */
    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;
    }
```

```java
    /**
     * 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: TiendaVirtual
 * 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.tienda.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
```

```java
     */
    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: TiendaVirtual
 * 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.tienda.modelo;

import android.app.Activity;
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.graphics.Color;
import android.net.Uri;
import android.text.Html;
import android.view.View;
import android.webkit.MimeTypeMap;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
```

```java
import com.example.tienda.R;
import com.example.tienda.vista.MainActivity;
import com.example.tienda.vista.MensajeActivity;
import com.example.tienda.vista.ProductoActivity;
import com.example.tienda.vista.ProductoUsuarioActivity;
import com.example.tienda.vista.StartActivity;
import com.example.tienda.vista.VenderActivity;
import com.example.tienda.vista.adapters.MensajeAdapter;
import com.example.tienda.vista.adapters.ProductoAdapter;
import com.example.tienda.vista.adapters.UsuarioAdapter;
import com.example.tienda.vista.fragments.PerfilFragment;
import com.example.tienda.vista.fragments.ProductosFragment;
import com.example.tienda.vista.fragments.UsuariosFragment;
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.DatabaseReference;
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.lang.reflect.Array;
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();
    ArrayList<String> idUsuarios = new ArrayList<String>();

    /**
     * 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;
        }

        mensaje.setHora(hora + ":" + minutos);
```

```
        conexion.getBaseDeDatos().child("Mensajes").push().setValue(mensaje);

    }

    /**
     * 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 != null) {
                        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 usuariosChat muestra la lista de usuarios que tienen un chat activo
     *
     * @param recyclerView
```

```
     * @param usuarioAdapter
     * @param usuarios
     */
    public void usuariosChat(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())
&& idUsuarios.contains(snapshot.getKey())) {
                        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 leerParaNotificar que muestra el chat completo entre dos personas
     *
     * @param context
     */
    public void leerParaNotificar(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);

        String contenido = mensaje.getContenido();
        if(mensaje.getTipo().equals("img")){
            contenido = "Imagen";
        }else if(mensaje.getTipo().equals("gps")){
            contenido = "ubicacion";
        }

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



    /*
     *
     *
     * TIENDA
     *
     *
     *
     * */


    /**
     * Metodo listarProductos muestra la lista de productos registrados en la base
     *
     * @param productos
     * @param productoAdapter
```

```
     * @param recyclerView
     */
    public void listarProductos(final List<Producto> productos, final
ProductoAdapter productoAdapter, final RecyclerView recyclerView) {

        conexion.getBaseDeDatos().child("Productos").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                productos.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Producto producto = snapshot.getValue(Producto.class);
                    producto.setId(snapshot.getKey());
                    if (!producto.getVendedor().equals(idUsuarioActual())) {
                        productos.add(producto);
                    }
                }

                productoAdapter.setProductos(productos);
                recyclerView.setAdapter(productoAdapter);
            }

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

            }
        });

    }

    /**
     * Metodo listarProductos muestra la lista de productos registrados en la base
     *
     * @param productos
     * @param productoAdapter
     * @param recyclerView
     * @param buscarProducto
     * @param categorias
     * @param minimo
     * @param maximo
     */
    public void listarProductos(final List<Producto> productos, final
ProductoAdapter productoAdapter, final RecyclerView recyclerView, final EditText
buscarProducto, final Spinner categorias, final EditText minimo, final EditText
maximo) {

        conexion.getBaseDeDatos().child("Productos").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                productos.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Producto producto = snapshot.getValue(Producto.class);
```

```
                    producto.setId(snapshot.getKey());
                    if (!producto.getVendedor().equals(idUsuarioActual())) {

                        if
(producto.getNombre().toLowerCase().contains(buscarProducto.getText().toString().to
LowerCase()) &&
producto.getCategoria().contains(categorias.getSelectedItem().toString())) {

                            if (filtrarPrecio(minimo, maximo, producto)) {
                                productos.add(producto);
                            }
                        }
                    }
                }

                productoAdapter.setProductos(productos);
                recyclerView.setAdapter(productoAdapter);
            }

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

            }
        });

    }

    /**
     * Metodo filtrarPrecio que permite filtrar un producto por su precio
     *
     * @param producto
     * @param minimo
     * @param maximo
     */
    public boolean filtrarPrecio(EditText minimo, EditText maximo, Producto
producto) {

        Double precio, min, max;
        try {
            precio = Double.parseDouble(producto.getPrecio());
        } catch (Exception ex) {
            precio = null;
        }

        try {
            min = Double.parseDouble(minimo.getText().toString());
        } catch (Exception ex) {
            min = null;
        }

        try {
            max = Double.parseDouble(maximo.getText().toString());
        } catch (Exception ex) {
            max = null;
        }
```

```
        if (min == null && max == null) {
            return true;
        } else if (min == null && max != null) {
            if (precio > max) {
                return false;
            } else {
                return true;
            }
        } else if (max == null && min != null) {
            if (precio < min) {
                return false;
            } else {
                return true;
            }
        } else if (precio >= min && precio <= max) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * Metodo buscarProducto que busca un produco en especifico
     *
     * @param context
     * @param idProducto
     * @param imagen
     * @param nombreProducto
     * @param descripcion
     * @param precio
     * @param vendedor
     */
    public void buscarProducto(final Context context, String idProducto, final
ImageView imagen, final TextView nombreProducto, final TextView descripcion, final
TextView precio, final TextView vendedor) {


conexion.getBaseDeDatos().child("Productos").child(idProducto).addListenerForSingle
ValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Producto producto = dataSnapshot.getValue(Producto.class);
                if (producto != null) {
                    nombreProducto.setText(producto.getNombre());
                    descripcion.setText(producto.getDescripcion());
                    precio.setText("$" + producto.getPrecio());
                    setNombreVendedor(vendedor, producto.getVendedor());
                    Glide.with(context).load(producto.getImagen()).into(imagen);
                }
            }

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

```
            }
        });

    }

    /**
     * Metodo setNombreVendedor que setea el nombre del vendedor a un producto
     *
     * @param vendedor
     * @param idUsuario
     */
    public void setNombreVendedor(final TextView vendedor, String idUsuario) {


conexion.getBaseDeDatos().child("Usuarios").child(idUsuario).addValueEventListener(
new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Usuario usuario = dataSnapshot.getValue(Usuario.class);
                vendedor.setText(usuario.getNombreUsuario());
            }

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

            }
        });

    }

    /**
     * Metodo listarCategorias muestra la lista de categorias registradas en la
base
     *
     * @param context
     * @param categorias
     */
    public void listarCategorias(final Context context, final Spinner categorias) {

        conexion.getBaseDeDatos().child("Categoria").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                ArrayList<String> misCategorias = new ArrayList<String>();
                misCategorias.add("");
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Categoria categoria = snapshot.getValue(Categoria.class);
                    if(categorias!=null) {
                        misCategorias.add(categoria.getNombre());
                    }
                }

                ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<String>(context, android.R.layout.simple_spinner_item, misCategorias);
```

```
                categorias.setAdapter(arrayAdapter);
            }

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

            }
        });

    }

    /**
     * Metodo publicarProducto que registra un nuevo producto
     *
     * @param context
     * @param producto
     * @param bandera
     */
    public void publicarProducto(final Context context, final Producto producto,
final Boolean bandera) {
        StorageTask storageTask;
        final StorageReference fileReference;
        final ProgressDialog progressDialog = new ProgressDialog(context);

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

        if (producto.getImagen() != null &&
producto.getImagen().contains("content://")) {

            fileReference =
conexion.getAlmacenamiento().child("Archivos").child(System.currentTimeMillis() +
"." + getExtensionArchivo(Uri.parse(producto.getImagen()), context));
            storageTask = fileReference.putFile(Uri.parse(producto.getImagen()));

            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();
                        producto.setImagen(uriBD);

                        if (bandera) {
```

```
conexion.getBaseDeDatos().child("Productos").push().setValue(producto);
                            Intent intent = new Intent(context,
ProductoUsuarioActivity.class);
                            context.startActivity(intent);
                            ((Activity) context).finish();
                        } else {
                            actualizar(producto);
                            ((Activity) context).onBackPressed();
                            ((Activity) context).finish();
                        }


                    } 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 if (producto.getImagen()!=null) {
            actualizar(producto);
            ((Activity) context).onBackPressed();
            ((Activity) context).finish();
            progressDialog.dismiss();
        } else {
            Toast.makeText(context, "No se ha seleccionado ninguna imagen",
Toast.LENGTH_SHORT).show();
            progressDialog.dismiss();
        }
    }

    public void botones(final Button comprar, final Button eliminar, final Button
actualizar, String productoId) {


conexion.getBaseDeDatos().child("Productos").child(productoId).addListenerForSingle
ValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Producto producto = dataSnapshot.getValue(Producto.class);
                if (producto.getVendedor().equals(idUsuarioActual())) {
                    comprar.setVisibility(View.GONE);
                    comprar.setEnabled(false);
                } else {
                    eliminar.setVisibility(View.GONE);
                    eliminar.setEnabled(false);
                    actualizar.setVisibility(View.GONE);
```

```
                    actualizar.setEnabled(false);
                }
            }

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

            }
        });

    }

    public void chatComprar(final Context context, final String productoId) {


conexion.getBaseDeDatos().child("Productos").child(productoId).addListenerForSingle
ValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Producto producto = dataSnapshot.getValue(Producto.class);
                Intent intent = new Intent(context, MensajeActivity.class);
                intent.putExtra("id", producto.getVendedor());

                Mensaje mensaje = new Mensaje();

                mensaje.setContenido("¿Sigue disponible el producto?\n" +
producto.getNombre());
                mensaje.setTipo("txt");
                mensaje.setEmisor(idUsuarioActual());
                mensaje.setReceptor(producto.getVendedor());
                enviarMensaje(mensaje);

                mensaje.setTipo("img");
                mensaje.setContenido(producto.getImagen());
                enviarMensaje(mensaje);

                context.startActivity(intent);
            }

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

            }
        });
    }

    public void listarMisProductos(final List<Producto> productos, final
ProductoAdapter productoAdapter, final RecyclerView recyclerView) {

        conexion.getBaseDeDatos().child("Productos").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                productos.clear();
```

```
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Producto producto = snapshot.getValue(Producto.class);
                    producto.setId(snapshot.getKey());
                    if (producto.getVendedor().equals(idUsuarioActual())) {
                        productos.add(producto);
                    }
                }

                productoAdapter.setProductos(productos);
                recyclerView.setAdapter(productoAdapter);
            }

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

            }
        });

    }

    public void eliminar(Context context, String productoId) {

conexion.getBaseDeDatos().child("Productos").child(productoId).removeValue();
        ((Activity) context).onBackPressed();
        ((Activity) context).finish();
    }

    public void leerUsuarios(final List<Usuario> usuarios, final UsuarioAdapter
usuarioAdapter, final RecyclerView recyclerView) {

        conexion.getBaseDeDatos().child("Mensajes").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                idUsuarios.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Mensaje mensaje = snapshot.getValue(Mensaje.class);
                    if (mensaje.getEmisor().equals(idUsuarioActual())) {
                        idUsuarios.add(mensaje.getReceptor());
                    } else if (mensaje.getReceptor().equals(idUsuarioActual())) {
                        idUsuarios.add(mensaje.getEmisor());
                    }
                }
                usuariosChat(usuarios, usuarioAdapter, recyclerView);
            }

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

            }
        });

    }

    public void leerUltimoMensaje(final TextView textView, 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();
                Mensaje ultimoMensaje = null;
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    Mensaje mensaje = snapshot.getValue(Mensaje.class);
                    if (mensaje != null) {
                        if (mensaje.getReceptor().equals(miId) &&
mensaje.getEmisor().equals(usuarioId) || mensaje.getReceptor().equals(usuarioId) &&
mensaje.getEmisor().equals(miId)) {
                            ultimoMensaje = mensaje;
                        }
                    }
                }

                if (ultimoMensaje != null) {
                    if(ultimoMensaje.getTipo().equals("gps")){
                        textView.setText("Ubicación");
                    }else if(ultimoMensaje.getTipo().equals("img")){
                        textView.setText("Imagen");
                    }else if (ultimoMensaje.getContenido().length() > 32) {
                        textView.setText(ultimoMensaje.getContenido().substring(0,
32) + "...");
                    } else {
                        textView.setText(ultimoMensaje.getContenido());
                    }
                }

            }

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

            }
        });
    }


    public void llenarVista(final Context context, String idProducto, final
ImageButton imageButton, final EditText nombreProducto, final EditText precio,
final EditText descripcion, final Spinner categorias) {


conexion.getBaseDeDatos().child("Productos").child(idProducto).addListenerForSingle
ValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Producto producto = dataSnapshot.getValue(Producto.class);
                if (producto != null) {
                    nombreProducto.setText(producto.getNombre());
```

```
                    descripcion.setText(producto.getDescripcion());
                    precio.setText(producto.getPrecio());

                    for (int i = 0; i < categorias.getAdapter().getCount(); i++) {
                        if
(categorias.getItemAtPosition(i).equals(producto.getCategoria())) {
                            categorias.setSelection(i);
                            break;
                        }
                    }

Glide.with(context).load(producto.getImagen()).into(imageButton);
                    imageButton.setAdjustViewBounds(true);

                    Intent intent = new Intent();
                    intent.putExtra("uri", producto.getImagen());
                    ((Activity) context).setIntent(intent);
                }
            }

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

            }
        });

    }

    public void actualizar(Producto producto) {
        HashMap<String, Object> miProducto = new HashMap<String, Object>();
        miProducto.put("nombre", producto.getNombre());
        miProducto.put("precio", producto.getPrecio());
        miProducto.put("categoria", producto.getCategoria());
        miProducto.put("descripcion", producto.getDescripcion());
        miProducto.put("vendedor", producto.getVendedor());
        miProducto.put("imagen", producto.getImagen());

conexion.getBaseDeDatos().child("Productos").child(producto.getId()).updateChildren
(miProducto);
    }

}
```

- **Clase Producto**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.
```

```java
 */
package com.example.tienda.modelo;

/**
 * Clase que contiene los datos del producto
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class Producto {

    private String id;
    private String nombre;
    private String precio;
    private String descripcion;
    private String imagen;
    private String vendedor;
    private String categoria;

    /**
     * Constructor vacio
     */
    public Producto() {
    }

    /**
     * Constructor con parametros
     * @param id
     * @param nombre
     * @param precio
     * @param descripcion
     * @param imagen
     * @param vendedor
     * @param categoria
     */
    public Producto(String id, String nombre, String precio, String descripcion,
String imagen, String vendedor, String categoria) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.descripcion = descripcion;
        this.imagen = imagen;
        this.vendedor = vendedor;
        this.categoria = categoria;
    }

    /**
     * Metodo getId que obtiene el id del producto
     * @return id
     */
    public String getId() {
        return id;
    }
```

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

    /**
     * Metodo getNombre que obtiene el nombre del producto
     * @return nombre
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Metodo setNombre que setea el nombre del producto
     * @param nombre
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
     * Metodo getPrecio que obtiene el precio del producto
     * @return precio
     */
    public String getPrecio() {
        return precio;
    }

    /**
     * Metodo setPrecio que setea el precio del producto
     * @param precio
     */
    public void setPrecio(String precio) {
        this.precio = precio;
    }

    /**
     * Metodo getDescripcion que obtiene la descripcion del producto
     * @return descripcion
     */
    public String getDescripcion() {
        return descripcion;
    }

    /**
     * Metodo setDescripcion que setea la descripcion del producto
     * @param descripcion
     */
    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }
```

```
    /**
     * Metodo getImagen que obtiene la imagen del producto
     * @return imagen
     */
    public String getImagen() {
        return imagen;
    }

    /**
     * Metodo setImagen que setea la imagen del producto
     * @param imagen
     */
    public void setImagen(String imagen) {
        this.imagen = imagen;
    }

    /**
     * Metodo getVendedor que obtiene el vendedor del producto
     * @return vendedor
     */
    public String getVendedor() {
        return vendedor;
    }

    /**
     * Metodo setVendedor que setea el vendedor del producto
     * @param vendedor
     */
    public void setVendedor(String vendedor) {
        this.vendedor = vendedor;
    }

    /**
     * Metodo getCategoria que obtiene la categoria del producto
     * @return categoria
     */
    public String getCategoria() {
        return categoria;
    }

    /**
     * Metodo setCategoria que setea la categoria del producto
     * @param categoria
     */
    public void setCategoria(String categoria) {
        this.categoria = categoria;
    }

}
```

- **Clase Usuario**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
```

```java
 * Sistema: TiendaVirtual
 * 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.tienda.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
     */
    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: TiendaVirtual
 * 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.tienda.presentador;

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

import androidx.recyclerview.widget.RecyclerView;

import com.example.tienda.modelo.Mensaje;
import com.example.tienda.modelo.Modelo;
import com.example.tienda.modelo.Usuario;
import com.example.tienda.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
 */
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 leerParaNotificar(Context context) {
        try {
            modelo.leerParaNotificar(context);
            return true;
        } catch (Exception ex) {
            return false;
        }
    }

}
```

Vista

- **Clase MensajeAdapter**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.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 androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

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

import java.util.ArrayList;
import java.util.List;
```

```java
/**
 * 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) {

        final Mensaje chat = mensajes.get(position);
        if (chat.getTipo().equals("gps")){

            holder.mostrarMensaje.setVisibility(View.GONE);

            holder.ubicacion.setOnClickListener(new View.OnClickListener()
{
                @Override
                public void onClick(View v) {
                    Intent intent = new Intent(contexto,
MapsActivity.class);
                    String[] coordenadas = chat.getContenido().split("/");
                    intent.putExtra("lat",coordenadas[0]);
                    intent.putExtra("lon",coordenadas[1]);
                    contexto.startActivity(intent);
                }
            });

            holder.hora.setPadding(8,120,0,0);

        }else if (chat.getTipo().equals("img")){
            holder.mostrarMensaje.setVisibility(View.GONE);
            holder.ubicacion.setVisibility(View.GONE);

Glide.with(contexto).load(chat.getContenido()).into(holder.imagenMensaje);
            holder.imagenMensaje.setAdjustViewBounds(true);
        }else {
            holder.ubicacion.setVisibility(View.GONE);
            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 CardView ubicacion;

        public ViewHolder(View itemView) {
            super(itemView);
            mostrarMensaje = itemView.findViewById(R.id.txtMensaje);
            hora = itemView.findViewById(R.id.txtHora);
            imagenMensaje=itemView.findViewById(R.id.imagenMensaje);
            ubicacion = itemView.findViewById(R.id.btnUbicacion);
        }
    }
    /**
     * 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 ProductoAdapter**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.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;
```

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

import com.bumptech.glide.Glide;
import com.example.tienda.R;
import com.example.tienda.modelo.Producto;
import com.example.tienda.vista.ProductoActivity;

import java.util.List;

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

    private Context context;
    private List<Producto> productos;
    /**
     * Constructor vacio
     */
    public ProductoAdapter() {
    }

    /**
     * Constructor
     * @param context
     * @param productos
     */
    public ProductoAdapter(Context context, List<Producto> productos) {
        this.context = context;
        this.productos = productos;
    }
    /**
     * Método getContexto que devuelve el contexto del producto
     * @return context
     */
    public Context getContext() {
        return context;
    }

    /**
     * Método setContexto que setea el contexto del producto
     * @param context
     */
    public void setContext(Context context) {
        this.context = context;
    }

    /**
```

```
     * Método getProductos que devuelve la lista de productos
     * @return productos
     */
    public List<Producto> getProductos() {
        return productos;
    }

    /**
     * Método setProductos que setea la lista de productos
     * @param productos
     */
    public void setProductos(List<Producto> productos) {
        this.productos = productos;
    }
    /**
     * Metodo onCreateViewHolder que crea un marcador de vista para cada elemento
     * @param parent
     * @param viewType
     */
    @NonNull
    @Override
    public ProductoAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.producto_item,
parent, false);
        return new ProductoAdapter.ViewHolder(view);
    }

    /**
     * Metodo onBindViewHolder obtiene nuevos titulares de vista
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull ProductoAdapter.ViewHolder holder, int
position) {
        final Producto producto = productos.get(position);
        holder.nombre.setText(producto.getNombre());
        holder.precio.setText("$"+producto.getPrecio());
        Glide.with(context).load(producto.getImagen()).into(holder.imagen);

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(context, ProductoActivity.class);
                intent.putExtra("id",producto.getId());
                context.startActivity(intent);
            }
        });
    }
    /**
     * Metodo getItemCount que devuelve el número de elementos en el adaptador
vinculado al RecyclerView padre.
     * @return numero de mensajes
     */
```

```
    @Override
    public int getItemCount() {
        return productos.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {

        public ImageView imagen;
        public TextView nombre, precio;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            imagen = itemView.findViewById(R.id.imgImagen);
            nombre = itemView.findViewById(R.id.txtNombreProducto);
            precio = itemView.findViewById(R.id.txtPrecio);
        }

    }
}
```

- **Clase UsuarioAdapter**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.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 androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

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

import java.util.List;

/**
```

```java
 * 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;
    Modelo modelo = new Modelo();


    /**
     * 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);
            }
        });

        modelo.leerUltimoMensaje(holder.ultimoMensaje, usuario.getId());
    }

    /**
     * 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,ultimoMensaje;
        public ImageView foto;

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

            nombreUsuario = itemView.findViewById(R.id.txtNombreUsuario);
            foto = itemView.findViewById(R.id.imgFoto);
            ultimoMensaje = itemView.findViewById(R.id.txtUltimoMensaje);
        }
    }

}
```

- **Clase PerfilFragment**

```
package com.example.tienda.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.Button;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import com.example.tienda.R;
import com.example.tienda.presentador.Presentador;
import com.example.tienda.vista.ProductoUsuarioActivity;
import com.example.tienda.vista.VenderActivity;

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();
    Button vender, misProductos;
```

```
    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);
        vender = view.findViewById(R.id.btnVender);
        misProductos = view.findViewById(R.id.btnMisProductos);


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


        vender.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getContext(), VenderActivity.class);
                startActivity(intent);
            }
        });

        misProductos.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getContext(),
ProductoUsuarioActivity.class);
                startActivity(intent);
            }
        });

        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 ProductosFragment**

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

import android.opengl.Visibility;
import android.os.Bundle;

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

import android.text.Editable;
import android.text.Layout;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.tienda.R;
import com.example.tienda.modelo.Modelo;
import com.example.tienda.modelo.Producto;
import com.example.tienda.presentador.Presentador;
import com.example.tienda.vista.adapters.ProductoAdapter;
```

```java
import java.util.ArrayList;
import java.util.List;

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

    private RecyclerView recyclerView;
    private ProductoAdapter productoAdapter;
    private List<Producto> productos;
    private Presentador presentador = new Presentador();
    private EditText buscarProducto, minimo, maximo;
    private Spinner categorias;
    private CardView cardView;
    private ImageButton filtrar;
    private boolean bandera=true;
    private Modelo modelo = new Modelo();

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

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

        productos = new ArrayList<>();
        productoAdapter = new ProductoAdapter(getContext(),productos);

        categorias = view.findViewById(R.id.spnCategoría);
        buscarProducto = view.findViewById(R.id.txtBuscarProducto);

        filtrar = view.findViewById(R.id.btnFiltrar);
        cardView = view.findViewById(R.id.miCardView);


        cardView.setVisibility(View.GONE);

        filtrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(bandera) {
```

```
                    cardView.setVisibility(View.VISIBLE);
                    bandera=false;
                }else{
                    cardView.setVisibility(View.GONE);
                    categorias.setSelection(0);
                    minimo.setText("");
                    maximo.setText("");
                    bandera=true;
                }
            }
        });


        minimo = view.findViewById(R.id.txtMin);
        maximo = view.findViewById(R.id.txtMax);

        minimo.addTextChangedListener(textWatcher);
        maximo.addTextChangedListener(textWatcher);
        buscarProducto.addTextChangedListener(textWatcher);



        categorias.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
                //if(!buscarProducto.getText().toString().equals("")) {
                    modelo.listarProductos(productos, productoAdapter,
recyclerView, buscarProducto, categorias, minimo, maximo);

//Toast.makeText(getContext(),"HOla",Toast.LENGTH_SHORT).show();
                //}
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // sometimes you need nothing here
            }
        });

        modelo.listarCategorias(getContext(),categorias);
        modelo.listarProductos(productos, productoAdapter, recyclerView);
        return view;
    }

    /**
     * La interfaz TextWatcher contiene una serie de métodos que permite ejecutar
texto antes, durante o después de realizar un cambio a través de EditText
     */
    TextWatcher textWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
```

```
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count)
{
            modelo.listarProductos(productos, productoAdapter, recyclerView,
buscarProducto, categorias, minimo, maximo);
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    };

}
```

- **Clase UsuariosFragment**

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

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

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

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

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**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

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

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

import com.example.tienda.R;
import com.example.tienda.presentador.Presentador;
import com.google.firebase.auth.FirebaseAuth;
```

```java
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**

```java
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

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

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 com.example.tienda.R;
import com.example.tienda.presentador.Presentador;
import com.example.tienda.vista.fragments.PerfilFragment;
import com.example.tienda.vista.fragments.ProductosFragment;
import com.example.tienda.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 {

    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.leerParaNotificar(MainActivity.this);

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

        tabLayout = findViewById(R.id.tabLayout);
        viewPager = findViewById(R.id.viewPager);

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


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


        tabLayout.getTabAt(0).setIcon(R.drawable.ic_action_productos);
        tabLayout.getTabAt(1).setIcon(R.drawable.ic_action_perfil);
        tabLayout.getTabAt(2).setIcon(R.drawable.ic_action_chat);
    }

    /**
     * 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 MapsActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;

import com.example.tienda.R;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

/**
 * Clase que contiene las propiedades de la sección de mapas
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
```

```java
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    LatLng ubicacionActual = null;

    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la sección de mapas de ubicación
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);


        ubicacionActual = new LatLng(Double.parseDouble(getIntent().getStringExtra("lat")), Double.parseDouble(getIntent().getStringExtra("lon")));


    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 0);
            onBackPressed();
            finish();
        } else {
            mMap.setMyLocationEnabled(true);
            mMap.addMarker(new MarkerOptions().position(ubicacionActual).title("Ubicación"));
            mMap.moveCamera(CameraUpdateFactory.newLatLng(ubicacionActual));
```

```
            mMap.animateCamera(CameraUpdateFactory.zoomTo(16));
        }
    }
}
```

- **Clase MensajeActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
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 androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.tienda.R;
import com.example.tienda.modelo.Mensaje;
import com.example.tienda.presentador.Presentador;
import com.google.android.gms.maps.model.LatLng;

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, btnEnviarUbicacion;
    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);
        btnEnviarUbicacion = findViewById(R.id.btnEnviarUbicacion);
        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();
            }
        });

        btnEnviarUbicacion.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Mensaje mensaje = new Mensaje();
                mensaje.setEmisor(presentador.idUsuarioActual());
                mensaje.setReceptor(userid);

                LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
                if (ActivityCompat.checkSelfPermission(MensajeActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(MensajeActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 0);
                } else {

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, new
LocationListener() {
                        @Override
                        public void onLocationChanged(Location location) {

                        }

                        @Override
                        public void onStatusChanged(String provider, int status,
Bundle extras) {

                        }
```

```
                        @Override
                        public void onProviderEnabled(String provider) {

                        }

                        @Override
                        public void onProviderDisabled(String provider) {

                        }
                    });

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, new
LocationListener() {
                        @Override
                        public void onLocationChanged(Location location) {

                        }

                        @Override
                        public void onStatusChanged(String provider, int status,
Bundle extras) {

                        }

                        @Override
                        public void onProviderEnabled(String provider) {

                        }

                        @Override
                        public void onProviderDisabled(String provider) {

                        }
                    });

                    Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

                    if (location == null) {
                        location =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    }

                    if (location != null) {
                        mensaje.setContenido(location.getLatitude() + "/" +
location.getLongitude());
                        mensaje.setTipo("gps");

                        if (!mensaje.getContenido().equals("")) {
                            presentador.enviarMensaje(mensaje);
                        } else {
                            Toast.makeText(MensajeActivity.this, "No se puede
enviar la ubicación", Toast.LENGTH_SHORT).show();
                        }
```

```
                    } else {
                        Toast.makeText(MensajeActivity.this, "Es necesario activar
la ubicación del dispositivo", Toast.LENGTH_SHORT).show();
                    }
                }

            }
        });

    }

    /**
     * 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);
        }
    }
}
```

- **ProductoActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;
```

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

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.tienda.R;
import com.example.tienda.modelo.Modelo;

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

    ImageView imagen;
    TextView nombreProducto, descripcion, precio, vendedor;
    Button comprar, eliminar, actualizar;
    Modelo modelo = new Modelo();
    String productoId;

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

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

        final Intent intent = getIntent();
        productoId  = intent.getStringExtra("id");

        imagen = findViewById(R.id.imgImagen);
        nombreProducto = findViewById(R.id.txtNombreProducto);
```

```
        descripcion = findViewById(R.id.txtDescripcion);
        precio = findViewById(R.id.txtPrecio);
        vendedor = findViewById(R.id.txtVendedor);

        comprar = findViewById(R.id.btnComprar);
        eliminar = findViewById(R.id.btnEliminar);
        actualizar = findViewById(R.id.btnActualizar);

        modelo.botones(comprar,eliminar, actualizar, productoId);

        comprar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                modelo.chatComprar(ProductoActivity.this, productoId);
            }
        });

        eliminar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                modelo.eliminar(ProductoActivity.this,productoId);
            }
        });

        actualizar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ProductoActivity.this,
VenderActivity.class);
                intent.putExtra("id",productoId);
                startActivity(intent);
            }
        });

        modelo.buscarProducto(ProductoActivity.this, productoId, imagen,
nombreProducto, descripcion,precio,vendedor);

    }

    /**
     * Metodo onResume que permite reanudar las actividades y componentes en donde
fueron detenidos o pausados
     */
    @Override
    public void onResume() {
        super.onResume();
        modelo.buscarProducto(ProductoActivity.this, productoId, imagen,
nombreProducto, descripcion, precio, vendedor);
    }
}
```

- **ProductoUsuarioActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
```

```java
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

import androidx.activity.OnBackPressedCallback;
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.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Spinner;

import com.example.tienda.R;
import com.example.tienda.modelo.Modelo;
import com.example.tienda.modelo.Producto;
import com.example.tienda.presentador.Presentador;
import com.example.tienda.vista.adapters.ProductoAdapter;

import java.util.ArrayList;
import java.util.List;

/**
 * Clase que contiene las propiedades de la vista de productos por usuario
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class ProductoUsuarioActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private List<Producto> productos;
    private ProductoAdapter productoAdapter;
    Modelo modelo = new Modelo();

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

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

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new
LinearLayoutManager(ProductoUsuarioActivity.this));

        productos = new ArrayList<>();
        productoAdapter = new
ProductoAdapter(ProductoUsuarioActivity.this,productos);

        modelo.listarMisProductos(productos, productoAdapter, recyclerView);
    }

}
```

- **Clase RegistroActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

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

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

import com.example.tienda.R;
import com.example.tienda.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**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

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

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

    Button ingresar, registrarse;
    Presentador presentador = new Presentador();

    /**
     * Metodo onStart que permite que el inicio de la aplicación sea visible para
el usuario
     */
        @Override
        protected void onStart() {
            super.onStart();
            if (presentador.estaLogeado()){
                Intent intent = new Intent(StartActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }


    /**
     * Metodo onCreate que realiza una llamada a la creación inicial de la interfaz
de inicio
```

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

            ingresar = findViewById(R.id.btnIngresar);
            registrarse = findViewById(R.id.btnRegistrarse);

            ingresar.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    startActivity(new Intent(StartActivity.this,
LoginActivity.class));
                }
            });

            registrarse.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    startActivity(new Intent(StartActivity.this,
RegistroActivity.class));
                }
            });
    }
}
```

- **Clase VenderActivity**

```
/*
 * ESPE - DCC - PROGRAMACIÓN MÓVIL
 * Sistema: TiendaVirtual
 * 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.tienda.vista;

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

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

```java
import com.bumptech.glide.Glide;
import com.example.tienda.R;
import com.example.tienda.modelo.Modelo;
import com.example.tienda.modelo.Producto;

/**
 * Clase que contiene las propiedades de la vista de los vendedores
 *
 * @author Paspuel Mayra
 * @author Quistanchala Karla
 * @author Villarruel Michael
 */
public class VenderActivity extends AppCompatActivity {

    Spinner categorias;
    ImageButton imageButton;
    Button publicar;
    Modelo modelo = new Modelo();
    EditText nombreProducto, precio, descripcion;
    private static final int IMAGE_REQUEST = 1;
    private Uri imagenUri;
    final Producto miProducto = new Producto();

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

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


        imageButton = findViewById(R.id.btnSubirImagenProducto);
        publicar = findViewById(R.id.btnPublicar);

        nombreProducto = findViewById(R.id.txtNombreProducto);
        precio = findViewById(R.id.txtPrecio);
        descripcion = findViewById(R.id.txtDescripcion);

        categorias = findViewById(R.id.spnCategoría);
        modelo.listarCategorias(VenderActivity.this, categorias);
```

```
        final String idProducto = getIntent().getStringExtra("id");

        if (idProducto != null) {
            publicar.setText("Publicar Cambios");
            modelo.llenarVista(VenderActivity.this, idProducto, imageButton,
nombreProducto, precio, descripcion, categorias);
        }

        publicar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                miProducto.setVendedor(modelo.idUsuarioActual());


                if (imagenUri != null) {
                    miProducto.setImagen(imagenUri.toString());
                } else if (idProducto != null) {
                    miProducto.setImagen(getIntent().getStringExtra("uri"));
                } else {
                    miProducto.setImagen("");
                }

                miProducto.setNombre(nombreProducto.getText().toString());
                miProducto.setPrecio(precio.getText().toString());
                miProducto.setDescripcion(descripcion.getText().toString());
                miProducto.setCategoria(categorias.getSelectedItem().toString());

                if (!miProducto.getImagen().equals("")
                        && !miProducto.getNombre().equals("")
                        && !miProducto.getPrecio().equals("")
                        && !miProducto.getDescripcion().equals("")
                        && !miProducto.getCategoria().equals("") &&
!miProducto.getVendedor().equals("")) {
                    if (idProducto != null) {
                        miProducto.setId(idProducto);
                        modelo.publicarProducto(VenderActivity.this, miProducto,
false);
                    } else {
                        modelo.publicarProducto(VenderActivity.this, miProducto,
true);
                    }
                } else {

                    Toast.makeText(VenderActivity.this, "Todos los campos deben ser
llenados correctamente", Toast.LENGTH_LONG).show();
                }


            }
        });

        imageButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                abrirImagenes();

//Toast.makeText(VenderActivity.this,producto.getImagen(),Toast.LENGTH_LONG).show()
;
            }
        });

    }

    /**
     * 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();

Glide.with(this.getApplicationContext()).load(imagenUri).into(imageButton);
            imageButton.setAdjustViewBounds(true);
        }
    }
}
```

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/