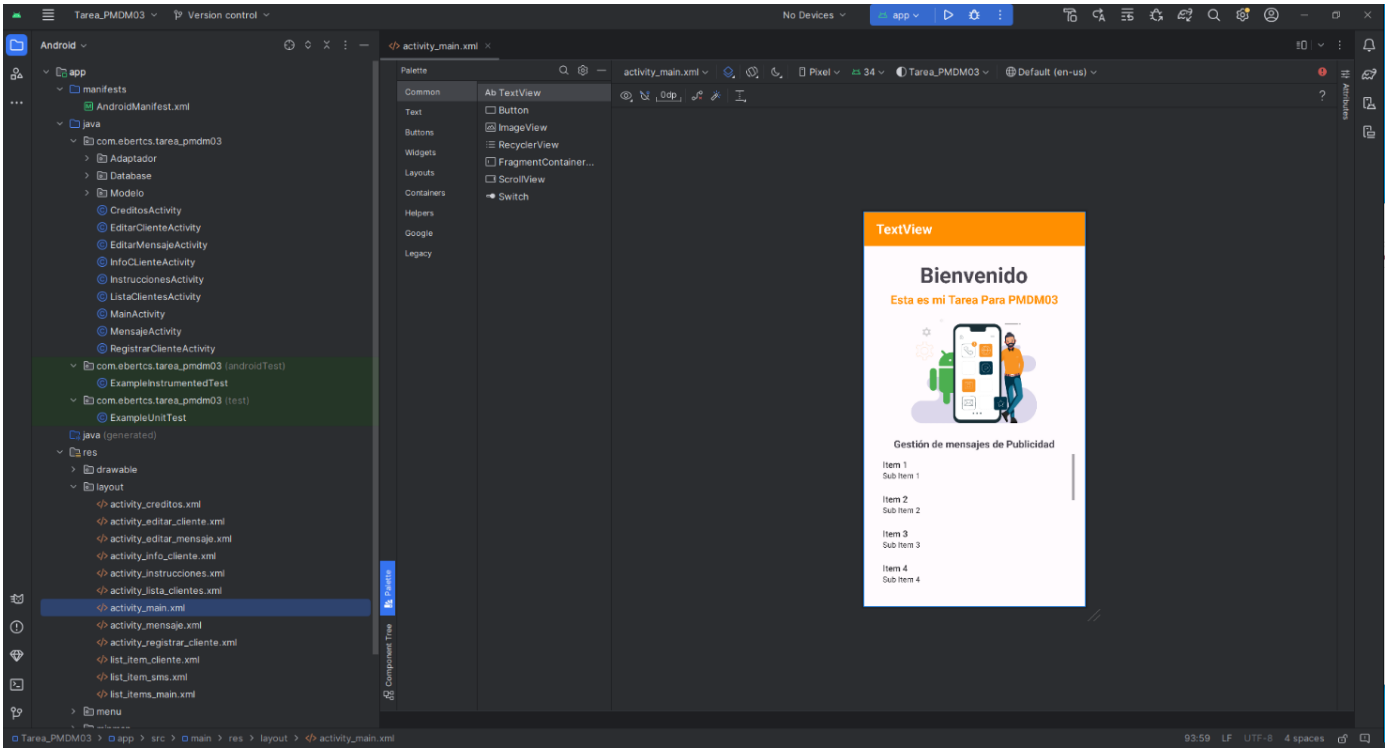
TAREA PARA PMDM03

Nombre del proyecto: Tarea_PMDM03

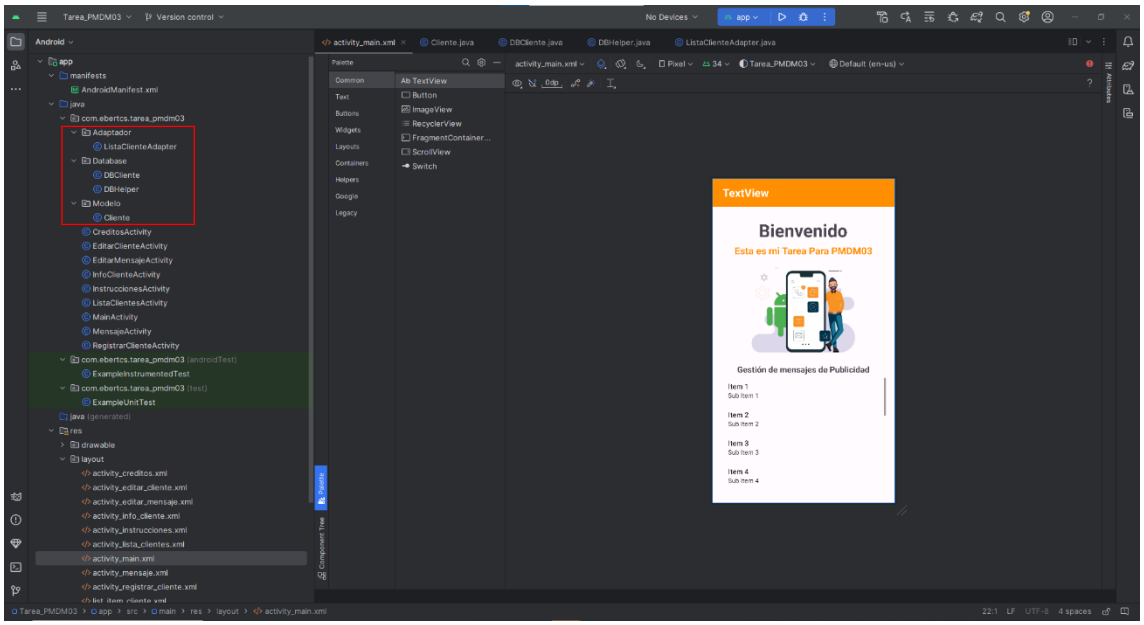
Lenguaje: Java

SDK: API 24(“Nougat”; Android 7.0)



A cada **Acitvity(.java)** le corresponde una **vista (.xml)**

- CreditosActivity **(activity_creditos)**
- EditarClienteActivity **(activity_editar_cliente)**
- EditarMensajeActivity**(activity_editar_mensaje)**
- InfoClienteActivity **(activity_info_cliente)**
- InstruccionesActivity **(activity_instrucciones)**
- ListaClientesActivity **(activity_lista_cliente)**
- MainActivity **(activity_main)**
- MensajeActivity **(activity_mensaje)**
- RegistrarClienteActivity **(activity_registrar_cliente)**



Tenemos la clase **ListaClientesAdapter** que es un adaptador que se encarga de definir cómo se mostrarán y organizarán los elementos individuales en la pantalla mediante **(RecyclerView)**, gestionando la creación de las vistas para cada elemento, la asignación de datos a esas vistas y facilita la presentación de información de manera ordenada.

DBCliente, contiene el código que permite hacer CRUD en nuestra tabla cliente.

DBHelper, contiene el código que facilita la creación inicial de la base de datos, define su estructura, y controla las actualizaciones de esquema. Ayuda a la gestión de la base de datos SQLite local en nuestra aplicación.

Cliente, la clase cliente representa a la entidad cliente y tiene atributos como nombre y teléfono. Permite realizar operaciones como la creación, lectura, actualización y eliminación (CRUD), y proporcionar una interfaz coherente para interactuar con los datos de clientes.

- **MainActivity,** esta actividad contiene un **ListView** al cual se le han pasado ítems mediante un ArrayList y un adaptador. Y mediante un método **setOnItemClickListener** se va a permitir realizar una acción cuando se presione sobre alguno de los ítems.

```
mensajero = new ArrayList<>();
mensajero.add("Lista de clientes");
mensajero.add("Modificar mensaje publicitario");
mensajero.add("Enviar mensaje publicitario");
mensajero.add("Salir");

//pasa los datos del array a la lista
ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.list_items_main,
mensajero);
lista.setAdapter(adapter);
lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {//al presionar sobre
un item ejecuta una accion
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        switch (position) {
            case 0:
                intent = new Intent(MainActivity.this, ListaClientesActivity.class);
                startActivity(intent);
                break;

        }
    }
});
```

También se tiene un toolbar que contiene un menú contextual con dos opciones que nos llevaran a la vista de Créditos e Instrucciones.

```
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_contextual, menu);
    return super.onCreateOptionsMenu(menu);
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.creditos) {
        Intent intent = new Intent(MainActivity.this, CreditosActivity.class);
        startActivity(intent);
    }

    if (item.getItemId() == R.id.instrucciones) {
        Intent intent = new Intent(MainActivity.this, InstruccionesActivity.class);
        startActivity(intent);
        finish();
    }
    return  true;
}
```

Y para poder cerrar la aplicación se ha utilizado:

```
case 3:
    finishAffinity();
    System.exit(0);
```

- **ListaClientesActivity,** esta actividad contine un RecyclerView que es donde se van a mostrar todos los clientes que se han registrado en la base de datos, muestra tanto el nombre como el número de teléfono, con el método **cargarLista()** mostrado a continuación vemos como se obtiene datos de la base de datos para rellenarlos en la lista.

  Para ello utiliza la clase **ListaClienteAdapter** para pasar los datos obtenidos al **RecyclerView** y se utiliza la clase DBCliente que es al que contiene el **CRUD** y en este caso solicitamos el método **mostrarClientes.** En la clase **ListaClientesAdapter**, se ha programado para que cuando se carguen los datos de la base de datos en la lista y luego presionemos sobre alguno de los clientes por tiempo prolongado, este nos abre un cuadro de dialogo donde nos preguntara si queremos ver la información del cliente.

```
public void cargarLista() {

    listaClientes = findViewById(R.id.listaClientes);
    listaClientes.setLayoutManager(new LinearLayoutManager(this));

    DBCliente dbCliente= new DBCliente(ListaClientesActivity.this);

    listaArrayClientes = new ArrayList<>();

    ListaClienteAdapter adapter = new ListaClienteAdapter(dbCliente.mostrarClientes());
    listaClientes.setAdapter(adapter);

}
```

Esta actividad al igual que el MainActivity tiene un menú contextual con un ítem que va a permitir ir a la vista que permite registrar cliente y otro ítem que permite volver a la vista principal.

- **InfoClienteActivity,** aquí se van a mostrar los datos del cliente seleccionado en la lista de clientes, se utiliza la clase **DBCliente** y en este caso solicitamos el método **verClienter(id),** esta vista contiene dos TexteView que es donde se van cargar los datos obtenidos del cliente seleccionado y botones flotantes uno para editar la información del cliente, y otra para eliminar al cliente.

```
TextView txtNombre = findViewById(R.id.infoNombre);
TextView txtTelefono = findViewById(R.id.infoTelefono);

final DBCliente dbCliente = new DBCliente(InfoClienteActivity.this);
cliente = dbCliente.verCliente(id);

if(cliente != null){
    txtNombre.setText(cliente.getNombre());
    txtTelefono.setText(cliente.getTelefono());
}
```

Cuando presionamos el botón de editar nos lleva a la vista de editar cliente y se manda el id del cliente se que se va editar mediante **intent.putExtra("ID", id)**

```
fabEditar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(InfoClienteActivity.this,
EditarClienteActivity.class);
        intent.putExtra("ID", id);
        startActivity(intent);
    }
});
```

Cuando presionamos el botón eliminar se abre un cuadro de dialogo que nos pide confirmación para poder eliminar al cliente. En este caso se utiliza el método **eliminiarCliente(id).**

```
fabEliminar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(InfoClienteActivity.this);
        builder.setMessage("¿Desea eliminar este Cliente?")
                .setPositiveButton("SI", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        if(dbCliente.eliminarCliente(id)){
                            lista();
                        }
                    }
                })
                .setNegativeButton("NO", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {

                    }
                }).show();
    }
});
```

- **RegistrarClienteActiviy,** aquí mediante dos EditText uno llamado txtNombre y otro txtTelefono se van a pasar los datos a la tabla cuando se presione sobre el botón registrar, se han validado los campos para que los campos siempre contengan información y que en el teléfono se ingresen valores numéricos de 9 dígitos. Se ha utilizado el método insertarCliente(nombre,Telefono), cuando se registra un cliente aparece un mensaje en un alertDialog que se cierra automáticamente en 1 segundo.

```
txtNombre = findViewById(R.id.txtNombre);
txtTelefono = findViewById(R.id.txtTelefono);
String nombre = txtNombre.getText().toString();
String telefono =txtTelefono.getText().toString();

if (!nombre.toString().equals("") && !telefono.toString().equals("") &&
telefono.length() ==9) {

          DBCliente clientes = new DBCliente(RegistrarClienteActivity.this);
          long id = clientes.insetarCLiente(nombre,telefono);

          AlertDialog.Builder builder = new
AlertDialog.Builder(RegistrarClienteActivity.this);
          if(id > 0){

              builder.setMessage("EL CLIENTE " + txtNombre.getText().toString() + " SE
HA REGISTRADO CORRECTAMENTE")
                      .setTitle("REGISTRO")
                      .setCancelable(false);

              // Crear el cuadro de diálogo
              final AlertDialog alertDialog = builder.create();
              alertDialog.show();

              // Crear un Handler para gestionar el retraso y cerrar el cuadro de
diálogo después de cierto tiempo
              new Handler().postDelayed(new Runnable() {
                  @Override
                  public void run() {
                      alertDialog.dismiss(); // Cierra el cuadro de diálogo después del
retraso
                  }
              }, 1000); // 3000 milisegundos (3 segundos) - puedes ajustar el tiempo
según tus necesidades

              limpiarCampos();
              ocultarTeclado();

          }else{
              builder.setMessage("ERROR AL INTENTAR REGISTRAR AL CLIENTE")
                      .setCancelable(false)
                      .setPositiveButton("Aceptar", new
DialogInterface.OnClickListener() {
                          public void onClick(DialogInterface dialog, int id) {
                              dialog.dismiss(); // Cierra el cuadro de diálogo
                          }
                      });

              // Crear y mostrar el cuadro de diálogo
              AlertDialog alertDialog = builder.create();
              alertDialog.show();
          }
      }else{

          if(nombre.equals("")){
              txtNombre.setError("Ingresa un Nombre");
          }
          if(telefono.length()!=9){
              txtTelefono.setError("El numero debe tener 9 digitos");
          }
          if(telefono.equals("")){
              txtTelefono.setError("Ingresa un Numero de Telefono");
          }
      }

  }
```

- **EditarClienteActiviy,** aquí se utilizan dos métodos, uno es el de **verCliente(id),** porque esto va a permitir cargar los EditTex nombre y teléfono con los datos del cliente; el otro método es **editarCliente(id,nombre,telefono),** el id fue pasado desde la vista InfoClienteActivity mediante **intent.putExtra("ID", id)** y aquí lo recibe mediante **extra.getInt("ID"),** se realiza la misma validación que en **RegistrarClienteActivity**

```
if (savedInstanceState == null) {
    Bundle extras = getIntent().getExtras();
    if (extras == null) {
        id = Integer.parseInt(null);
    } else {
        id = extras.getInt("ID");
    }
} else {
    id = (int) savedInstanceState.getSerializable("ID");
}

final DBCliente dbCliente = new DBCliente(EditarClienteActivity.this);
cliente = dbCliente.verCliente(id);

if(cliente != null){
    txtNombre.setText(cliente.getNombre());
    txtTelefono.setText(cliente.getTelefono());
}

btnActualizar.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {

        String nombre = txtNombre.getText().toString();
        String telefono =txtTelefono.getText().toString();

        if (!nombre.toString().equals("") && !telefono.toString().equals("") &&
telefono.length() ==9) {
            correcto = dbCliente.editarCliente(id, txtNombre.getText().toString(),
txtTelefono.getText().toString());

            if(correcto){
                Toast.makeText(EditarClienteActivity.this, "Se ha Modificado los datos
del cliente", Toast.LENGTH_LONG).show();
                verRegistro();
            } else {
                Toast.makeText(EditarClienteActivity.this, "Error al modificar datos del
cliente", Toast.LENGTH_LONG).show();
            }
        } else {
            if(nombre.equals("")){
                txtNombre.setError("Ingresa un Nombre");
            }
            if(telefono.length()!=9){
                txtTelefono.setError("El número debe tener 9 digitos");
            }
            if(telefono.equals("")){
                txtTelefono.setError("Ingresa un Numero de Telefono");
            }
        }
    }
});
```

- **EditarMensajeActivy,** aquí se va editar el mensaje publicitario para ello se utiliza la clase **SharedPreferences** que se utiliza para almacenar y recuperar pequeñas cantidades de datos de manera persistente. Cuando presionemos el botón de guardar el texto se va a guardar, el texto puede ser editado las veces que sean necesarias.

```
preferences = getSharedPreferences("miSharedPreferences",MODE_PRIVATE);
editMensaje = (EditText) findViewById(R.id.textArea_information);

guardarMensaje = preferences.getString("textKey", "");
editMensaje.setText(guardarMensaje);


Button btnGuardar = (Button) findViewById(R.id.btnGuardarMensaje);
btnGuardar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String textToSave = editMensaje.getText().toString();
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("textKey", textToSave);
        editor.apply();
        Toast.makeText(EditarMensajeActivity.this,"Texto guardado
correctamente",Toast.LENGTH_SHORT).show();
    }
});
```

en la primera línea de código en el **getSharedPreferences** observamos entre comillas **"miSharedPreferences" y "textKey"** son importantes porque van a permitir que se obtenga la información de nuestro mensaje desde otra actividad.

- **MensajeActivy,** primero vamos recuperar el mensaje que queremos enviar a los clientes y que hemos editado en la actividad **EditarMensajeActivity**, para ellos recuperamos el mensaje con **preferences.getString** cómo se observa en el siguiente código:

```
preferences = getSharedPreferences("miSharedPreferences",MODE_PRIVATE);
mensajeRecuperado = preferences.getString("textKey", "valor_por_defecto");
```

Luego se ha creado un **ArrayList mensajeConcatenado** que va a mostrar una lista para cada mensaje publicitario donde cada ítem va estar dirigido para cada cliente de la base de datos. Para ello **ArrayList<Cliente> listaArrayClientes** se le ha asignado los datos obtenidos con el método **mostarCliente ().** y se ha recorrido mediante el bucle for.

```
DBCliente dbCliente= new DBCliente(MensajeActivity.this);
listaArrayClientes = dbCliente.mostrarClientes();

mensajesConcatenados = new ArrayList<>();

for (Cliente cliente : listaArrayClientes) {

    String nombre = cliente.getNombre().toString();
    String m = this.mensajeRecuperado.replace("*", nombre);//el * es remplazado por el
nombre

    mensajesConcatenados.add(m);
}
ArrayAdapter adapter = new ArrayAdapter<String>(this,
R.layout.list_item_sms,mensajesConcatenados);
lista.setAdapter(adapter);
```

Luego tenemos el método **enviarMensajeSMS ()** que comprueba que se tenga permiso para enviar el mensaje y si se tiene permiso se ejecuta el método **sendSms().** En esta actividad es importante importar **android.Manifest** y también hacer modificaciones **AndroidManifest.xml** como muestro a continuación:

```
<uses-permission android:name="android.permission.SEND_SMS" />

<uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
```

Código para verificar si se tiene permiso para enviar SMS.

```
private void enviarMensajeSMS() {
    // Verificar si ya se tiene el permiso
    if (ContextCompat.checkSelfPermission(this,
Manifest.permission.SEND_SMS)
            == PackageManager.PERMISSION_GRANTED) {
        // Si se tiene el permiso, enviar el SMS
        sendSms();
    } else {
        // Si no se tiene el permiso, solicitarlo
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.SEND_SMS},
                PERMISSION_REQUEST_SMS);
    }

}
```

Se utiliza la clase SmsManager que permite enviar mensajes SMS directamente desde una aplicación, sin necesidad de interactuar con la aplicación de mensajería predeterminada del dispositivo. Luego se hace el mismo procedimiento que hicimos para llenar la lista con el mensaje para cada cliente y metodo que va permitir enviar el mensaje es, **sendTextMesage():**

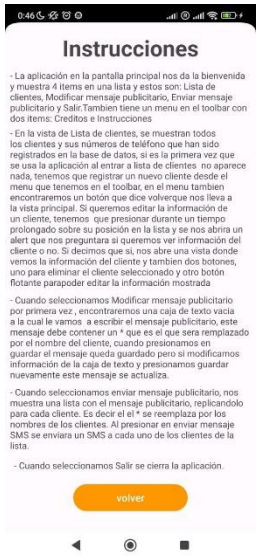
```
private void sendSms(){

    SmsManager smsManager = SmsManager.getDefault();

    DBCliente dbCliente= new DBCliente(MensajeActivity.this);
    listaArrayClientes = dbCliente.mostrarClientes();

    for (Cliente cliente : listaArrayClientes) {

        String nombre = cliente.getNombre().toString();
        String telefono = cliente.getTelefono().toString();

        String mensajeSMS = this.mensajeRecuperado.replace("*", nombre);
        smsManager.sendTextMessage("+34"+telefono,null, mensajeSMS, null, null);
        Toast.makeText(this,"SE HA ENVIADO EL MENSAJE A : " + listaArrayClientes.size()+
" CLIENTES",Toast.LENGTH_SHORT).show();
    }

}
```

Se agrego el prefijo +34 para poder enviar los mensajes ya que los datos de teléfono ingresados en la tabla de clientes eran solo 9 dígitos.

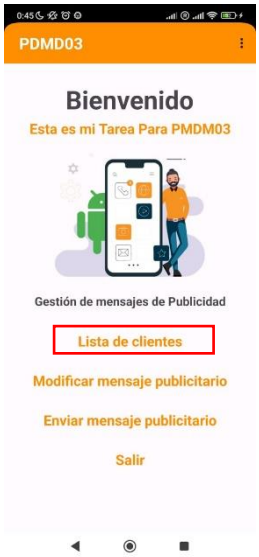
Funcionamiento de la app desde mi dispositivo móvil.

Pantalla de principal con menú contextual y una lista donde se va poder gestionar clientes, escribir y modificar mensaje publicitario que se va enviar a cada cliente de la base de datos, enviar mensaje publicitario que contiene los mensajes y clientes a los que se le va mandar los mensajes al presionar el botón enviar. Y salir que va hacer que la aplicación se cierre

El menú contextual tiene el botón Créditos que nos lleva a la vista de créditos y la vista crédito cuenta con un botón para volver a la pantalla principal

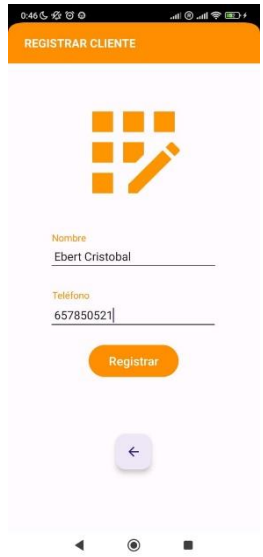


El menú contextual tiene el botón instrucciones que nos lleva a la vista de Instrucciones y la vista Instrucciones cuenta con un botón para volver a la pantalla principal



Cuando presionamos en Lista de cliente se abre esta otra vista, esta vacia porque hemos instalado la app por primera vez y la base de datos no tiene datos

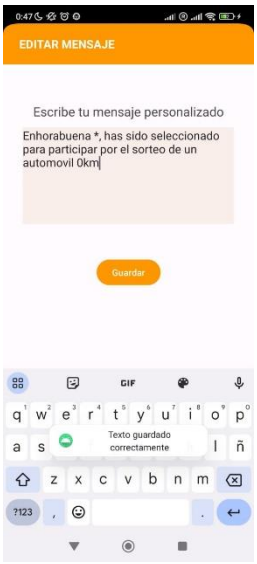
Lista de clientes tambien tambien tienemenu contextual uno para registarr cliente y otro para volver a la vista principal.



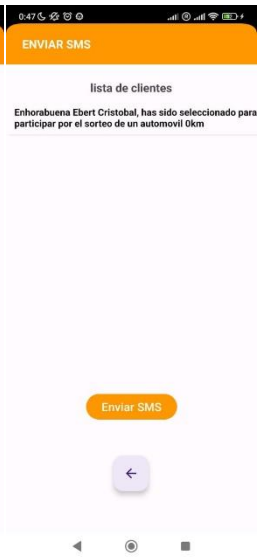
Hemos entrado a la vista registrar cliente y hemos ingresado mis datos para realizar la prueba de se crea un suario correctamente

Al dar en registrar se limpia el EditText y podemos ingresar nuevos datos para ingresar mas clientes. En este caso presionamos el botón flotante para volver a lista de clientes. Y como se puede observar ya se ha registrado un cliente. Ahora presionamos donde los tres puntitos y se abre el menu contextual y selecionamos donde dice volver.

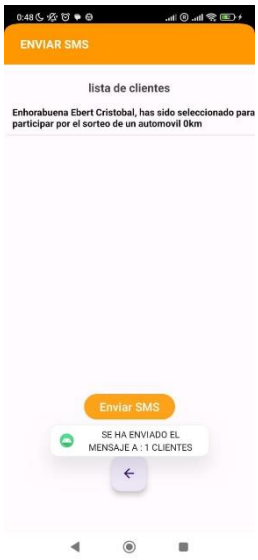
Presionamos donde dice modificar mensaje publicitario y se nos abre la vista editar mensaje.



Escribimos el mensaje que queremos enviar y donde va el * se reemplazara por el nombre del cliente, presionamos en guardar y el mensaje queda guardado, si modificamos algo del mensaje para guardar los cambios debemos presionar nuevamente en el botón. Cuando presionamos guardar podemos ver un mensaje que dice que el texto se guardo correctamente. Tambien cuenta con un botón flotante para volver hacia la pantalla principal.

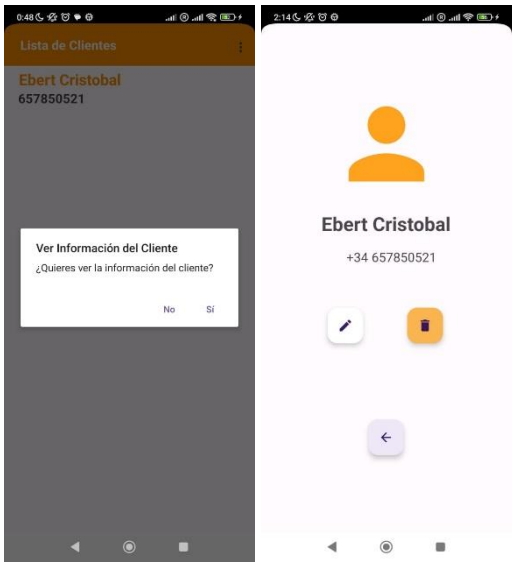


Cuando presionamos Enviar mensaje publicitario se nos abre la vista enviar mensaje y como podemos ver que aparece una lista con el mensaje y con el nombre del cliente, en este caso solo se ha ingresado un cliente para hacer la prueba de enviar el sms

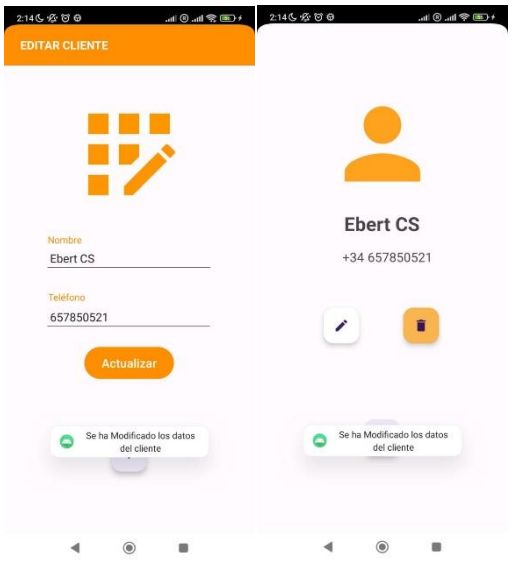


Cuando presionamos en el botón enviar SMS nos pide permiso para que la aplicación pueda leer y enviar mensajes. Despúes de dar a permitir volvemos a presionar el botón enviar SMS y podemos ver el mensaje que dice que se enviado el mensaje a un solo cliente.

Aquí podemos ver que se me ha enviado el sms , como le estoy probando desde mi movil, me autoenvio un mensaje. Esta aplicación solo puede probarse si se tiene una tarjeta sim en el dispositivo que tiene la aplicación.



Si presionamos un sobre el usario unos segundos nos abre un alertDialog donde nos pregunta si queremos ver informacion del cliente, presionamos SI y se nos abre otra vista donde vemos la información del cliente, y se puede también editar o eliminar desde aquí.Vamos a presionar el botón editat.



Cambiamos el nombre y presionamos en acutalizar y automaticamente nos envia a la vista donde se muestra la informacion del cliente, aquí podemos ver que se ha cambiado el nombre

Por último si presionamos sobre el boton eliminar se nos abre un alertDialog que nos pide confirmación para eliminar el contacto

Codigo del DBCliente.

```
public class DBCliente extends DBHelper {

    Context context;

    public DBCliente(@Nullable Context context) {
        super(context);
        this.context = context;
    }

    //registra cliente
    public long insetarCLiente(String nombre, String telefono){

        long id=0;
        try{
            DBHelper dbHelper = new DBHelper(context);
            SQLiteDatabase db = dbHelper.getWritableDatabase();

            ContentValues values = new ContentValues();
            values.put("nombre" , nombre);
            values.put("telefono" , telefono);

            id = db.insert(TABLE_NAME,null,values);

        }catch (Exception e){
            e.toString();
        }

        return id;

    }

    //muetsra todos los clientes
    public ArrayList<Cliente> mostrarClientes(){

        DBHelper dbHelper = new DBHelper(context);
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        ArrayList<Cliente> listaClientes = new ArrayList<>();
        Cliente cliente = null;
        Cursor cursorClientes=null;

        cursorClientes = db.rawQuery("SELECT * FROM "+TABLE_NAME + " ORDER BY id DESC" ,
null);

        if(cursorClientes.moveToFirst()){

            do {
                cliente = new Cliente();
                cliente.setId(cursorClientes.getInt(0));
                cliente.setNombre(cursorClientes.getString(1));
                cliente.setTelefono(cursorClientes.getString(2));
                listaClientes.add(cliente);
            }while (cursorClientes.moveToNext());
        }

        cursorClientes.close();

        return listaClientes;
    }

    //nformacion de un solo cliente
    public  Cliente verCliente(int id){

        DBHelper dbHelper = new DBHelper(context);
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        Cliente cliente = null;
        Cursor cursorClientes;

        cursorClientes = db.rawQuery("SELECT * FROM "+TABLE_NAME + " WHERE id = " + id +
" LIMIT 1 ", null);

        if(cursorClientes.moveToFirst()){
```

```
            cliente = new Cliente();
            cliente.setId(cursorClientes.getInt(0));
            cliente.setNombre(cursorClientes.getString(1));
            cliente.setTelefono(cursorClientes.getString(2));

        }

        cursorClientes.close();
        return cliente;
    }

    //permite editar cliente
    public boolean editarCliente(int id, String nombre, String telefono) {

        boolean correcto = false;

        DBHelper dbHelper = new DBHelper(context);
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        try {
            db.execSQL("UPDATE " + TABLE_NAME + " SET nombre = '" + nombre + "',
telefono = '" + telefono + "' WHERE id='" + id + "' ");
            correcto = true;
        } catch (Exception ex) {
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }

    //permite eliminar un cliente
    public boolean eliminarCliente(int id) {

        boolean correcto = false;

        DBHelper dbHelper = new DBHelper(context);
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        try {
            db.execSQL("DELETE FROM " + TABLE_NAME + " WHERE id = '" + id + "'");
            correcto = true;
        } catch (Exception ex) {
            ex.toString();
            correcto = false;
        } finally {
            db.close();
        }

        return correcto;
    }
```

Codigo del DBHelper.

```
public class DBHelper extends SQLiteOpenHelper {

    private Context context;
    private static final String DATABASE_NAME = "publicidad.db";
    private static final int DATABASE_VERSION = 1;

    public static final String TABLE_NAME = "clientes";


    public DBHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null,DATABASE_VERSION);

    }
    //crea la tabla en la base de datos
    @Override
    public void onCreate(SQLiteDatabase db) {
     db.execSQL("CREATE TABLE " + TABLE_NAME + "( " +
             "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
             "nombre TEXT NOT NULL, " +
             "telefono TEXT NOT NULL) ");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```