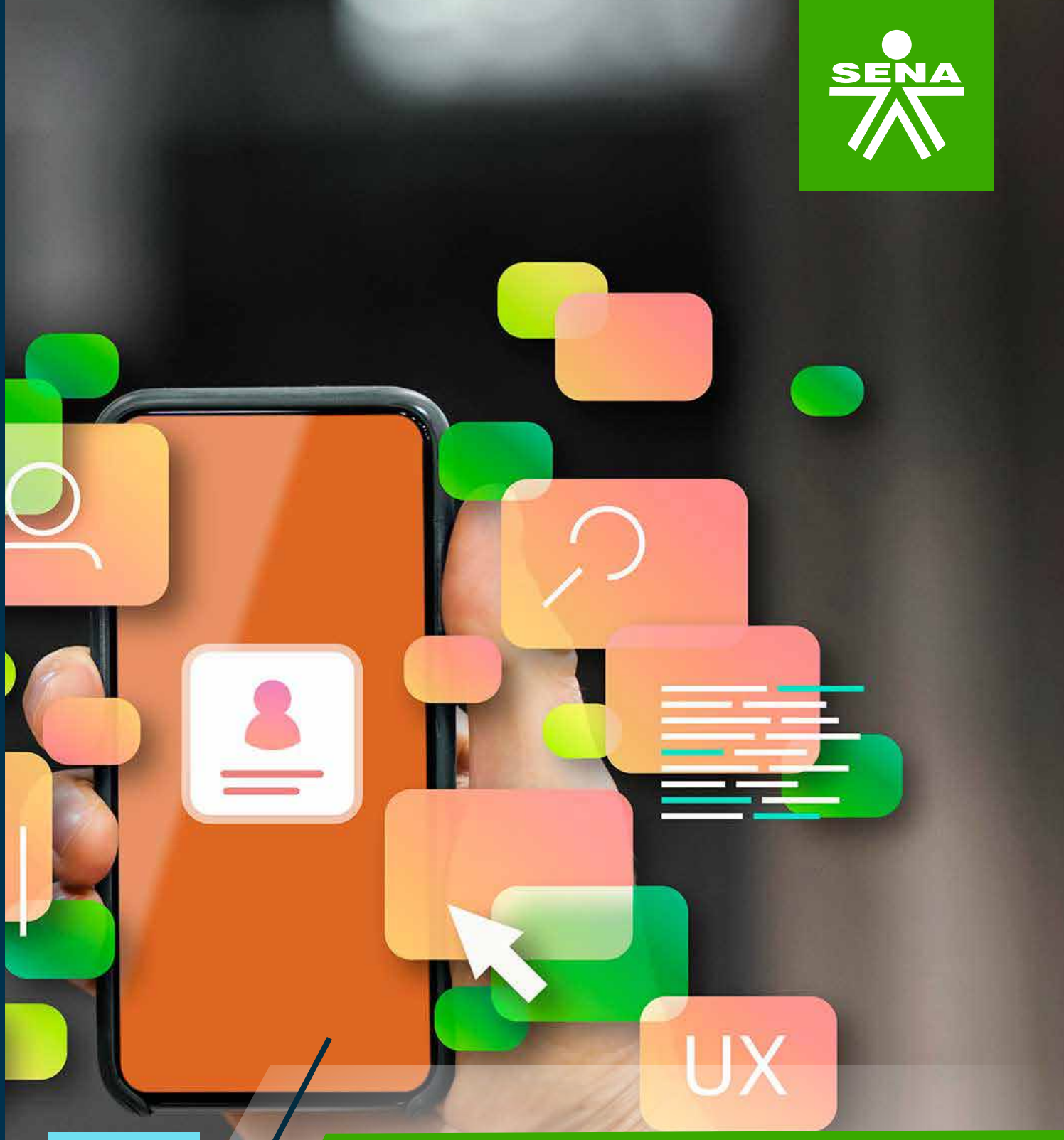


Uso de base de datos en Android

Desarrollo de aplicaciones móviles

Servicio Nacional de Aprendizaje - SENA

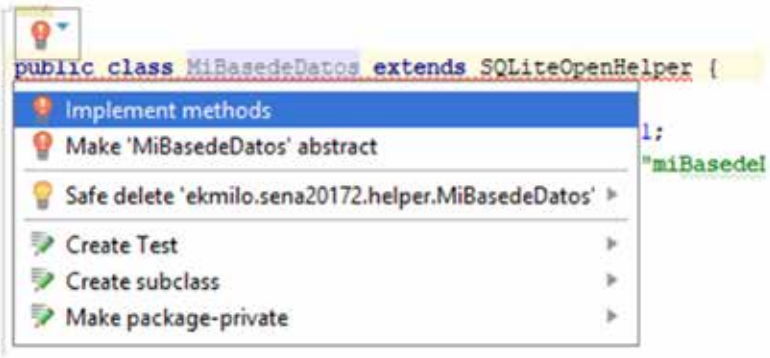
Uso de base de datos en Android

1. Una vez que estemos en nuestro proyecto crearemos una nueva clase llamada MiBasedeDatos la cual debe de extender o heredar de **SQLiteOpenHelper**. Esta clase se encuentra en el paquete android.database.sqlite.

Una vez que tenemos nuestra clase heredando de SQLiteOpenHelper necesitaremos agregar dos nuevos métodos:

- Un método onCreate el cual recibe como parámetro un objeto SQLIteDatabase.
- Un método onUpgrade que recibe como parámetros un objeto SQLiteDatabase y dos enteros (versión anterior y nueva versión).

Quedando de la siguiente manera:




```
public class MiBasedeDatos extends SQLiteOpenHelper {

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

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

    }
}
```

2. En esta misma clase creamos dos variables estáticas constantes para manejar la versión de nuestra base de datos y el nombre de la misma:

```
public class MiBasedeDatos extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "miBasedeDatos.db";
```

3. Definimos ahora el constructor de nuestra clase que recibirá como parámetro un objeto de tipo Context. Este constructor debe llamar al constructor de la clase de la cual heredamos (SQLiteOpenHelper), pasándole como parámetros el contexto, el nombre de nuestra base de datos, un cursor Factory (para este tutorial será null) y una versión, quedando de la siguiente manera:

```
public class MiBasedeDatos extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "miBasedeDatos.db";

    public MiBasedeDatos(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }
```

4. Lo que corresponde ahora es crear la base de datos y para ellos debemos hacerlo en nuestro método onCreate, sin embargo primero debemos establecer los nombres de las columnas y tablas, para una administración más fácil, para este colocaremos estos nombres en variables de tipo String estáticas en la misma clase:

```
public class MiBasedeDatos extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "miBasedeDatos.db";

    public static final String TABLA_USUARIO = "usuario";
    public static final String COLUMNA_USUARIO_ID = "id";
    public static final String COLUMNA_USUARIO_NOMBRE = "nombre";
    public static final String COLUMNA_USUARIO_EDAD = "edad";
    public static final String COLUMNA_USUARIO_CEDULA = "cedula";
    public static final String COLUMNA_USUARIO_ENTIDAD = "entidad_id";

    public static final String TABLA_ENTIDAD = "entidad";
    public static final String COLUMNA_ENTIDAD_ID = "id";
    public static final String COLUMNA_ENTIDAD_NOMBRE = "nombre";
    public static final String COLUMNA_ENTIDAD_NIT = "nit";
```

5. Para la creación de las tablas debemos realizarlas mediante sentencias SQL, las cuales definimos en variables:

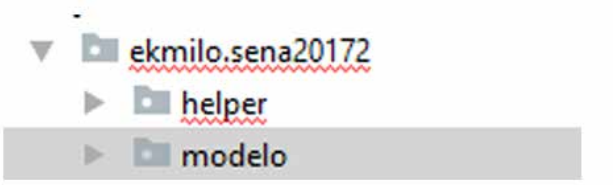
```
private static final String SQL_CREAR_ENTIDAD = "create table "
        + TABLA_ENTIDAD + "(" + COLUMNA_ENTIDAD_ID
        + " integer primary key autoincrement, " + COLUMNA_ENTIDAD_NOMBRE
        + " text not null, " + COLUMNA_ENTIDAD_NIT
        + " text not null);";

private static final String SQL_CREAR_USUARIO = "create table "
        + TABLA_USUARIO + "(" + COLUMNA_USUARIO_ID
        + " integer primary key autoincrement, " + COLUMNA_USUARIO_NOMBRE
        + " text not null, " + COLUMNA_USUARIO_EDAD
        + " integer, " + COLUMNA_USUARIO_CEDULA
        + " integer not null, " + COLUMNA_USUARIO_ENTIDAD
        + " integer not null);"
        + " FOREIGN KEY ("+COLUMNA_USUARIO_ENTIDAD+") REFERENCES "+TABLA_ENTIDAD+"("+COLUMNA_ENTIDAD_ID+"));";
```

6. Para correr las sentencias SQL, usamos el método execSQL del objeto de la clase SQLiteDataBase en el método onCreate en la clase:

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL(SQL_CREAR_ENTIDAD);
    sqLiteDatabase.execSQL(SQL_CREAR_USUARIO);
}
```

7. Crearemos un paquete modelo donde crearemos las clases Usuario y Entidad con los campos diseñados en nuestro diagrama entidad relación, esto lo hacemos con el fin de facilitar las operaciones sobre la base de datos:



Clase Entidad:

```
public class Entidad {
    String nombre,nit;
    Integer id;

    public Entidad(String nombre, String nit) {
        this.nombre = nombre;
        this.nit = nit;
    }
    public Entidad(String nombre, String nit,Integer id) {
        this.nombre = nombre;
        this.nit = nit;
        this.id=id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getNit() {
        return nit;
    }

    public void setNit(String nit) {
        this.nit = nit;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
}
```

Clase Usuario:

```
public class Usuario {
    String nombre;
    Integer edad, entidad,cedula,id;
    public Usuario(String nombre, Integer cedula, Integer edad, Integer entidad) {
        this.nombre = nombre;
        this.cedula = cedula;
        this.edad = edad;
        this.entidad = entidad;
    }
    public Usuario(String nombre, Integer cedula, Integer edad, Integer entidad, Integer id) {
        this.nombre = nombre;
        this.cedula = cedula;
        this.edad = edad;
        this.entidad = entidad;
        this.id=id;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public Integer getCedula() {
        return cedula;
    }
    public void setCedula(Integer cedula) {
        this.cedula = cedula;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public Integer getEdad() {
        return edad;
    }
    public void setEdad(Integer edad) {
        this.edad = edad;
    }
    public Integer getEntidad() {
        return entidad;
    }
    public void setEntidad(Integer entidad) {
```

8. En la case MiBasedeDatos, implementamos la función para Crear de nuestro CRUD de las tablas (Create, Read, Update, Delete), lo primero que hacemos es indicarle a la clase que vamos a hacer uso de la base de datos, lo segundo es crear un objeto ContentValues el cual nos ayudara a almacenar de manera temporal nuestros datos relacionándolos a la columna a la cual pertenecen, por ultimo le indicamos a la base de datos que deseamos insertar un nuevo elemento para ello utilizamos su método insert:

```
public void crearEntidad(Entidad entidad){
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();

    values.put(COLUMNA_ENTIDAD_NOMBRE, entidad.getNombre());
    values.put(COLUMNA_ENTIDAD_NIT, entidad.getNit());

    db.insert(TABLA_ENTIDAD,  nullColumnHack: null,values);
    db.close();

}
```

9. Ahora, implementamos la función leer (Read), lo primero que debemos de hacer es indicarle a la clase que vamos a utiliza la base datos (como en el método crear), para el query de select debemos de tener una proyección (nombre con que lo maneja la documentación) que no es más que un arreglo de todas nuestras columnas con las cuales vamos a trabajar. El método query recibe como parámetro principalmente el nombre de la tabla a utilizar, la proyección, nuestra condicional para el select, los siguiente cuatro parámetros son para realizar un query más complejo. Para almacenar el resultado de nuestra búsqueda debemos hacer mediante un objeto Cursor e iterar sobre él para obtener cada fila de los registros obtenidos:

```
public List<Entidad> leerEntidades() {
    SQLiteDatabase db = this.getReadableDatabase();
    String[] projection = {COLUMNA_ENTIDAD_ID, COLUMNA_ENTIDAD_NOMBRE, COLUMNA_ENTIDAD_NIT};
    List<Entidad> list = new ArrayList<Entidad>();

    Cursor cursor =
            db.query(TABLA_ENTIDAD,
                    projection,
                    selection: null,
                    selectionArgs: null,
                    groupBy: null,
                    having: null,
                    orderBy: null,
                    limit: null);


    if (cursor != null) {
        if (cursor.moveToFirst()) {
            while (!cursor.isAfterLast()) {
                String nombre = cursor.getString(cursor.getColumnIndex(COLUMNA_ENTIDAD_NOMBRE));
                String nit = cursor.getString(cursor.getColumnIndex(COLUMNA_ENTIDAD_NIT));
                Integer id = cursor.getInt(cursor.getColumnIndex(COLUMNA_ENTIDAD_ID));
                Entidad entidad = new Entidad(nombre, nit, id);
                list.add(entidad);
                cursor.moveToNext();
            }
        }
        db.close();
    }
    return list;
}
```

10. Ahora implementamos el método actualizar (Update), al igual que con el método agregar debemos de almacenar nuestros valores en sus respectivas columnas mediante un objeto ContentValues y con el método update de nuestra base de datos establecemos que tabla queremos utilizar, los valores a cambiar (con el objeto ContentValues) y la condicional para realizar el cambio.

" id = ?" nos indica que utilizaremos el campo id como referencia para saber que filas serán actualizadas y con new String[] { String.valueOf(id) }); indicamos el valor que debe tener la columna.

```
public void actualizarEntidad (Entidad entidad){

    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(COLUMNA_ENTIDAD_NOMBRE, entidad.getNombre());
    values.put(COLUMNA_ENTIDAD_NIT, entidad.getNit());

    int i = db.update(TABLA_ENTIDAD,
            values,
             whereClause: " id = ?",
            new String[] { String.valueOf( entidad.getId() ) });
    db.close();
}
```

11. Finalmente creamos el método borrar (Delete),

```
public boolean eliminarEntidad(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    try{
        db.delete(TABLA_ENTIDAD,
                 whereClause: " id = ?",
                new String[] { String.valueOf (id ) });
        db.close();
        return true;

    }catch(Exception ex){
        return false;
    }
}
```

12. Basado en los métodos anteriores del CRUD de Entidad, haremos los de Usuario:

Crear:

```
public void crearUsuario(Usuario usuario){
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();

    values.put(COLUMNA_USUARIO_NOMBRE, usuario.getNombre());
    values.put(COLUMNA_USUARIO_EDAD, usuario.getEdad());
    values.put(COLUMNA_USUARIO_CEDULA, usuario.getCedula());
    values.put(COLUMNA_USUARIO_ENTIDAD, usuario.getEntidad());

    db.insert(TABLA_USUARIO,  nullColumnHack: null,values);
    db.close();

}
```

Leer:

```
public List<Usuario> leerUsuarios() {
    SQLiteDatabase db = this.getReadableDatabase();
    String[] projection = {COLUMNA_USUARIO_ID, COLUMNA_USUARIO_NOMBRE, COLUMNA_USUARIO_EDAD,COLUMNA_USUARIO_CEDULA,COLUMNA_USUARIO_ENTIDAD};
    List<Usuario> list = new ArrayList<Usuario>();

    Cursor cursor =
            db.query(TABLA_USUARIO,
                    projection,
                    selection: null,
                    selectionArgs: null,
                    groupBy: null,
                    having: null,
                    orderBy: null,
                    limit: null);


    if (cursor != null) {
        if (cursor.moveToFirst()) {
            while (!cursor.isAfterLast()) {
                String nombre = cursor.getString(cursor.getColumnIndex(COLUMNA_USUARIO_NOMBRE));
                Integer edad = cursor.getInt(cursor.getColumnIndex(COLUMNA_USUARIO_EDAD));
                Integer cedula = cursor.getInt(cursor.getColumnIndex(COLUMNA_USUARIO_CEDULA));
                Integer entidad = cursor.getInt(cursor.getColumnIndex(COLUMNA_USUARIO_ENTIDAD));
                Integer id = cursor.getInt(cursor.getColumnIndex(COLUMNA_USUARIO_ID));
                Usuario usuario  = new Usuario(nombre, cedula,edad, entidad,id);
                list.add(usuario);
                cursor.moveToNext();
            }
        }
        db.close();
    }
    return list;
}
```

Actualizar :

```
public void actualizarUsuario (Usuario usuario){

    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(COLUMNA_USUARIO_NOMBRE, usuario.getNombre());
    values.put(COLUMNA_USUARIO_EDAD, usuario.getEdad());
    values.put(COLUMNA_USUARIO_CEDULA, usuario.getCedula());
    values.put(COLUMNA_USUARIO_ENTIDAD, usuario.getEntidad());

    int i = db.update(TABLA_USUARIO,
            values,
            whereClause: " id = ?",
            new String[] { String.valueOf( usuario.getId() ) });
    db.close();
}
```

Borrar :

```
public boolean eliminarUsuario(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    try{
        db.delete(TABLA_USUARIO,
                whereClause: " id = ?",
                new String[] { String.valueOf (id ) });
        db.close();
        return true;

    }catch(Exception ex){
        return false;
    }
}
```

13. Con el fin de dar acceso a una única instancia de la base de datos y gestionarla en el ciclo de vida de la aplicación, lo más sencillo es hacer que se cree un único objeto de la clase a través del patrón singleton que implementaremos a continuación, debemos dejar el constructor en privado y crear una variable privada para la clase así como un método público getInstance para obtener la misma instancia siempre:

```
private static MiBasedeDatos miBasedeDatos;
public static MiBasedeDatos getInstance(Context context){
    if(miBasedeDatos == null){
        miBasedeDatos = new MiBasedeDatos(context);
    }
    return miBasedeDatos;
}

private MiBasedeDatos(Context context) {
    super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
}
```

14. Ahora resta hacer las pruebas de nuestra base de datos, en nuestra actividad crearemos 4 botones para Crear, Actualizar, Listar y Borrar, por ahora usaremos datos estáticos para hacer las pruebas. También creamos un EditText para mostrar los elementos en el método Listar:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/crear"
        android:text="Crear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/leer"
        android:text="Leer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/actualizar"
        android:text="Actualizar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/borrar"
        android:text="Borrar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/edit_text"
        android:inputType="textMultiLine"
        android:gravity="left|top"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

15. En nuestra actividad:

```
MiBasedeDatos miBasedeDatos;
Button crear,leer,actualizar,borrar;
EditText editText;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    crear = (Button)findViewById(R.id.crear);
    leer = (Button)findViewById(R.id.leer);
    actualizar = (Button)findViewById(R.id.actualizar);
    borrar = (Button)findViewById(R.id.borrar);
    editText = (EditText)findViewById(R.id.edit_text);

    miBasedeDatos=MiBasedeDatos.getInstance(getApplicationContext());
```

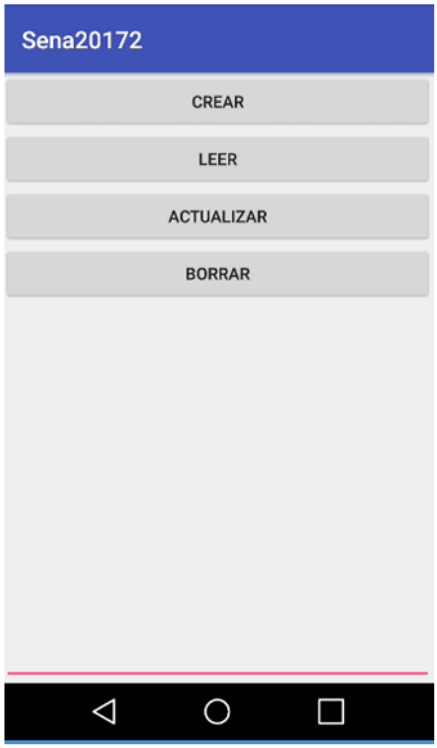
```
crear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Entidad entidad = new Entidad( nombre: "Entidad " + (int)(Math.random()*10),  nit: "nit "+(int)(Math.random()*10));
        miBasedeDatos.crearEntidad(entidad);
        Usuario usuario = new Usuario( nombre: "Usuario "+ (int)(Math.random()*10) ,(int)(Math.random()*10) ,(int)(Math.random()*10) , miBasedeDatos.leerEntidades().get(0).getId());
        miBasedeDatos.crearUsuario(usuario);
    }
});
leer.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
       editText.setText(mostrarDatos());
    }
});
borrar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        miBasedeDatos.eliminarEntidad(miBasedeDatos.leerEntidades().get(0).getId());
        miBasedeDatos.eliminarUsuario(miBasedeDatos.leerUsuarios().get(0).getId());
    }
});
actualizar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Entidad entidad = new Entidad( nombre: "Entidad " + (int)(Math.random()*10),  nit: "nit "+(int)(Math.random()*10),miBasedeDatos.leerEntidades().get(0).getId() );
        miBasedeDatos.actualizarEntidad(entidad);
        Usuario usuario = new Usuario( nombre: "Usuario "+ (int)(Math.random()*10) ,(int)(Math.random()*10) ,(int)(Math.random()*10) , miBasedeDatos.leerEntidades().get(0).getId(),
                miBasedeDatos.leerUsuarios().get(0).getId());
        miBasedeDatos.actualizarUsuario(usuario);
    }
});
```

Adicionamos un método para leer los nombres de entidades y usuarios:

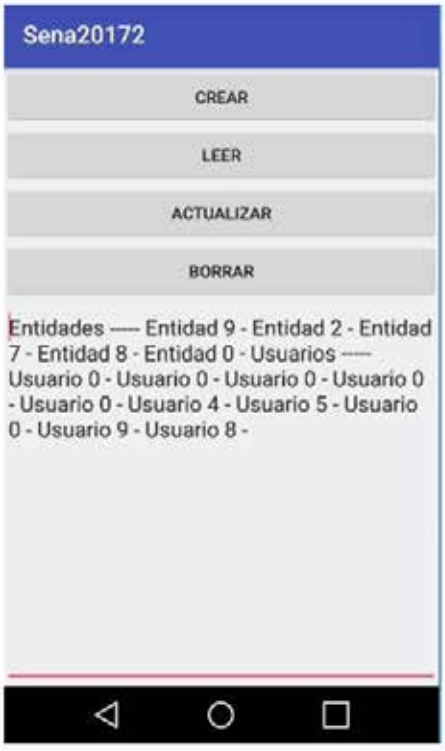
```
private String mostrarDatos(){
    StringBuilder sb=new StringBuilder();
    List<Entidad> listEntidades=miBasedeDatos.leerEntidades();
    List<Usuario> listUsuarios = miBasedeDatos.leerUsuarios();
    sb.append("Entidades ----- ");
    for (Entidad entidad:listEntidades){
        sb.append(entidad.getNombre()).append(" - ");
    }
    sb.append("Usuarios ----- ");
    for(Usuario usuario:listUsuarios){
        sb.append(usuario.getNombre()).append(" - ");
    }
    return sb.toString();
}
```

Generando:

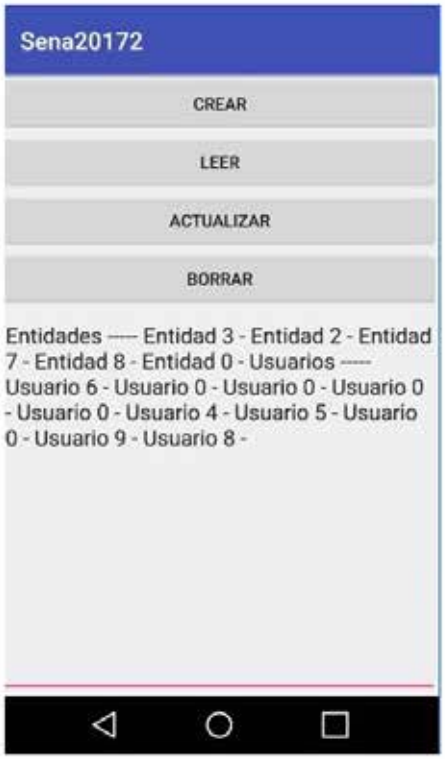
Crear:



Leer:



Actualizar:



Borrar: