CORPORACIÓN UNIVERSITARIA AUTÓNOMA DEL CAUCA

PROGRAMACION II

ENTREGABLE EJERCICIOS CLASE 11

APP GESTION NOTAS DE ESTUDIANTES

PRESENTADO POR:

SALOMÓN MONTILLA LUNA

PRESENTADO A:

CRISTHIAN ALEJANDRO CAÑAR MUÑOZ

POPAYÁN, CAUCA

2025

CAPTURAS DE PANTALLA

CONTROLLERS:

```java
public class EstudianteController {
    6 usages
    private DataBaseHelper dbHelper;

    4 usages
    public EstudianteController(Context context) {
        dbHelper = new DataBaseHelper(context);
    }

    1 usage
    public void agregarEstudiante(String nombre, String codigo) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("nombre", nombre);
        values.put("codigo", codigo);
        db.insert( table: "estudiantes",  nullColumnHack: null, values);
        db.close();
    }

    2 usages
    public List<Estudiante> obtenerEstudiantes() {
        List<Estudiante> lista = new ArrayList<>();
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        Cursor cursor = db.rawQuery( sql: "SELECT * FROM estudiantes",  selectionArgs: null);

        if (cursor.moveToFirst()) {
            do {
                lista.add(new Estudiante(cursor.getInt( i: 0), cursor.getString( i: 1), cursor.getString( i: 2)));
            } while (cursor.moveToNext());
        }
        cursor.close();
        return lista;
    }

    6 usages
    public Estudiante obtenerEstudiantePorCodigo(String codigo){
        Estudiante estudiante = null;
        SQLiteDatabase db = dbHelper.getReadableDatabase();

        Cursor cursor = db.rawQuery( sql: "SELECT * FROM estudiantes WHERE codigo = ?", new String[]{codigo});

            if (cursor.moveToFirst()) {
                estudiante = new Estudiante(
                        cursor.getInt( i: 0),    // id
                        cursor.getString( i: 1), // nombre
                        cursor.getString( i: 2)  // codigo
                );
            }
            cursor.close();
            db.close();
        return estudiante;
    }

    1 usage
    public void eliminarEstudiante(int id) {
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        db.delete( table: "estudiantes",  whereClause: "id = ?",new String[]{String.valueOf(id)});
        db.close();
    }

    1 usage
    public void editarEstudiante(String nuevoNombre, String nuevoCodigo, int id){
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues valores = new ContentValues();
        valores.put("nombre", nuevoNombre);
        valores.put("codigo", nuevoCodigo);
        db.update( table: "estudiantes", valores,  whereClause: "id = ?", new String[]{String.valueOf(id)});
        db.close();
    }
}
```

```java
public class NotaController {

    5 usages
    private DataBaseHelper dbHelper;

    5 usages
    public NotaController(Context context) {
        dbHelper = new DataBaseHelper(context);
    }

    2 usages
    public void agregarNota(int id, double valor) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("estudiante_id", id);
        values.put("nota", valor);
        db.insert( table: "notas",  nullColumnHack: null, values);
        db.close();
    }

    3 usages
    public List<Nota> obtenerNotasPorEstudiante(int id) {
        List<Nota> lista = new ArrayList<>();
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        Cursor cursor = db.rawQuery( sql: "SELECT * FROM notas WHERE estudiante_id = ?",
                new String[]{String.valueOf(id)});

        if (cursor.moveToFirst()) {
            do {
                lista.add(new Nota(cursor.getInt( i: 0), cursor.getInt( i: 1), cursor.getDouble( i: 2)));
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
        return lista;
    }

    3 usages
    public double calcularPromedio(List<Nota> notas) {
        if(notas.isEmpty()){
            return 0;
        }
        //itera sobre cada elemento de la lista para obtener su valor y sumarlo para obtener promedio
        return (notas.stream() Stream<Nota>
                .mapToDouble(Nota::getValor) DoubleStream
                .sum())/notas.size();
    }

    1 usage
    public void eliminarNota(int id) {
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        db.delete( table: "notas",  whereClause: "id = ?",new String[]{String.valueOf(id)});
        db.close();
    }

    1 usage
    public void editarNota(int notaId, double nuevoValor) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues valores = new ContentValues();
        valores.put("nota", nuevoValor);
        db.update( table: "notas", valores,  whereClause: "id = ?", new String[]{String.valueOf(notaId)});
        db.close();
    }
}
```

MODELS

```java
public class DataBaseHelper extends SQLiteOpenHelper {
    1 usage
    private static final String NOMBRE_DB = "universidad.db";
    1 usage
    private static final int VERSION_DB = 1;


    2 usages
    public DataBaseHelper(Context context) { super(context, NOMBRE_DB, factory: null, VERSION_DB); }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("CREATE TABLE estudiantes (id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "nombre TEXT, codigo TEXT)");
        sqLiteDatabase.execSQL("CREATE TABLE notas (id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "estudiante_id INTEGER, nota REAL," +
                " FOREIGN KEY (estudiante_id) REFERENCES estudiantes(id))");
    }

    no usages
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS estudiantes");
        db.execSQL("DROP TABLE IF EXISTS notas");
        onCreate(db);
    }
}
```

```java
public class Estudiante {
    3 usages
    private int id;
    3 usages
    private String nombre;
    3 usages
    private String codigo;

    2 usages
    public Estudiante(int id, String nombre, String codigo) {
        this.id = id;
        this.nombre = nombre;
        this.codigo = codigo;
    }

    no usages
    public Estudiante() {
    }

    public int getId() { return id; }

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

    public String getNombre() { return nombre; }

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

    public String getCodigo() { return codigo; }

    public void setCodigo(String codigo) { this.codigo = codigo; }
}
```

```java
public class Nota {
    3 usages
    private int id;
    3 usages
    private int estudianteId;
    3 usages
    private double valor;

    1 usage
    public Nota(int id, int estudianteId, double valor) {
        this.id = id;
        this.estudianteId = estudianteId;
        this.valor = valor;
    }

    public int getId() { return id; }

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

    no usages
    public int getEstudianteId() { return estudianteId; }

    no usages
    public void setEstudianteId(int estudianteId) { this.estudianteId = estudianteId; }

    3 usages
    public double getValor() {
        return valor;
    }

    no usages
    public void setValor(double valor) { this.valor = valor; }
}
```

VIEWS

```java
public class AgregarEstudiantesActivity extends AppCompatActivity {

    15 usages
    private ActivityAgregarEstudiantesBinding binding;
    3 usages
    private EstudianteController estudianteController = new EstudianteController( context: this);
    1 usage
    private NotaController notaController = new NotaController( context: this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityAgregarEstudiantesBinding.inflate(getLayoutInflater());
        setContentView (binding.getRoot());

        //Accion agregar estudiante
        binding.agregarButton.setOnClickListener(view -> {
            String codigo = binding.codigoInput.getText().toString();
            String nombre = binding.nombreInput.getText().toString();
            String nota = binding.notaInput.getText().toString();

            if (codigo.isEmpty() || nombre.isEmpty() || nota.isEmpty()) {
                binding.codigoInput.setError("Este campo no puede estar vacio!");
                binding.nombreInput.setError("Este campo no puede estar vacio!");
                binding.notaInput.setError("Este campo no puede estar vacio!");
                //Toast.makeText(getApplicationContext(), "Por favor, complete todos los campos.", Toast.LENGTH_
            }else {
                double notaValid = Double.parseDouble(nota);
                if((notaValid <1 || notaValid > 5)){//valida entre 1 y 5
                    binding.notaInput.setError("La nota debe ser entre 1 y 5!");
                }else{
                    Estudiante estudiante = estudianteController.obtenerEstudiantePorCodigo(codigo);
                    if(estudiante != null){//valida que no haya codigos en uso
                        binding.codigoInput.setError("Este codigo ya esta en uso!");
                    }else{
                        estudianteController.agregarEstudiante(nombre, codigo);
                        Estudiante nuevoEstudiante = estudianteController.obtenerEstudiantePorCodigo(codigo);
                        notaController.agregarNota(nuevoEstudiante.getId(), Double.parseDouble(nota));
                        Toast.makeText(getApplicationContext(), text: "Estudiante agregado exitosamente.",
                                Toast.LENGTH_SHORT).show();
                        binding.codigoInput.setText("");
                        binding.nombreInput.setText("");
                        binding.notaInput.setText("");
                    }
                }
            }
        });

        binding.volverBtn.setOnClickListener(view ->{
            Intent intent = new Intent( packageContext: AgregarEstudiantesActivity.this, MainActivity.class);
            startActivity(intent);
        });


    }
}
```

```java
public class DetallesEstudianteActivity extends AppCompatActivity {
    21 usages
    ActivityDetallesEstudianteBinding binding;
    6 usages
    NotaController notaController = new NotaController( context: this);
    2 usages
    EstudianteController estudianteController = new EstudianteController( context: this);

    9 usages
    List<Nota> notas = new ArrayList<>();
    4 usages
    NotaListaAdapter notasAdapter = new NotaListaAdapter( context: this, notas);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityDetallesEstudianteBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        ListView listNotas = binding.listaDeNotas;

        listNotas.setAdapter(notasAdapter);



        //Accion ver promedio
        binding.verPromedioBtn.setOnClickListener(v -> {
        String codigoEstudiante = binding.codigoEstudiante.getText().toString().trim();

            if (codigoEstudiante.isEmpty()) {
                //Toast.makeText(this, "Por favor ingresa un código", Toast.LENGTH_SHORT).show();
                binding.codigoEstudiante.setError("Este campo no puede estar vacio!");
                return;
            }

            Estudiante estudiante = estudianteController.obtenerEstudiantePorCodigo(codigoEstudiante);

            if (estudiante == null) {
                binding.codigoEstudiante.setError("Estudiante no encontrado!");
                return;
            }

            // Actualiza las notas del estudiante
            notas.clear();
            notas.addAll(notaController.obtenerNotasPorEstudiante(estudiante.getId()));
            notasAdapter.notifyDataSetChanged();

            // Calcula el promedio con las notas actualizadas
            double promedio = notaController.calcularPromedio(notas);


            // Muestra los datos
            binding.tvNombre.setText(estudiante.getNombre());
            binding.tvPromedio.setText(String.format("%.1f", promedio));

        });

        //Accion Agregar notas
        binding.btnAgregarNotaDetalles.setOnClickListener(view->{
            String codigoEstudiante = binding.codigoEstudiante.getText().toString().trim();
            String nota = String.valueOf(binding.inputAgregarNotaDetalles.getText());
            Estudiante estudiante = estudianteController.obtenerEstudiantePorCodigo(codigoEstudiante);

            if (codigoEstudiante.isEmpty() || nota.isEmpty()) {
//              Toast.makeText(this, "Por favor llena todos los campos", Toast.LENGTH_SHORT).show();
                binding.codigoEstudiante.setError("Este campo no puede estar vacio!");
                binding.inputAgregarNotaDetalles.setError("Este campo no puede estar vacio!");
                return;
            }
```

```java
            //valida la nota agregada
            double notaAgregada = Double.parseDouble(nota);
            if(notaAgregada < 1 || notaAgregada > 5){
                binding.inputAgregarNotaDetalles.setError("La nota debe ser entre 1 y 5!");
                return;
            }

            //Agrega la nota y actualiza
            notaController.agregarNota(estudiante.getId(), notaAgregada);
            notas.clear();
            notas.addAll(notaController.obtenerNotasPorEstudiante(estudiante.getId()));
            notasAdapter.notifyDataSetChanged();

            // Calcula el promedio con las notas actualizadas
            double promedio = notaController.calcularPromedio(notas);

            Toast.makeText( context: this,  text: "Nota agregada!", Toast.LENGTH_SHORT).show();
            binding.tvPromedio.setText(String.format("%.1f",  promedio));
            binding.inputAgregarNotaDetalles.setText("");
        });

        //Accion para editar o eliminar notas
        binding.listaDeNotas.setOnItemClickListener((parent, view, position, id)->{
            Nota notaSeleccionada = notas.get(position);
            mostrarDialogoOpciones(notaSeleccionada);
        });

        //cambio de actividad
        binding.btnVolver.setOnClickListener(v -> {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        });

    }
    1 usage
    private void mostrarDialogoOpciones(Nota nota) {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setTitle("Selecciona una acción")
                .setItems(new CharSequence[]{"Editar", "Eliminar"}, (dialog, which) -> {
                    if (which == 0) {
                        // Editar
                        Intent intent = new Intent( packageContext: this, EditarNotaActivity.class);
                        intent.putExtra( name: "idNota", nota.getId()); // pasar el id de la nota
                        intent.putExtra( name: "notaActual", nota.getValor());
                        startActivity(intent);

                    } else if (which == 1) {
                        // Eliminar
                        mostrarDialogoConfirmacion(nota);
                    }
                })
                .show();
    }
    1 usage
    private void mostrarDialogoConfirmacion(Nota nota) {
        new AlertDialog.Builder( context: this)
                .setTitle("Confirmar eliminación")
                .setMessage("¿Estás seguro de eliminar esta nota?")
                .setPositiveButton( text: "Sí", (dialog, which) -> {
                    notaController.eliminarNota(nota.getId());
                    notas.remove(nota);
                    notasAdapter.notifyDataSetChanged(); // actualiza la lista
                    Toast.makeText( context: this,  text: "Nota eliminada", Toast.LENGTH_SHORT).show();
                    binding.tvNombre.setText("Ingresa el codigo");
                    binding.tvPromedio.setText("Ingresa el codigo");
                })
                .setNegativeButton( text: "Cancelar",  listener: null)
                .show();
    }
}
```

```java
public class EditarEstudianteActivity extends AppCompatActivity {
    11 usages
    ActivityEditarEstudianteBinding binding;
    3 usages
    EstudianteController estudianteController = new EstudianteController( context: this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        String codEstudiante = getIntent().getStringExtra( name: "codEstudiante");
        super.onCreate(savedInstanceState);
        binding = ActivityEditarEstudianteBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        Estudiante estudiante = estudianteController.obtenerEstudiantePorCodigo(codEstudiante);

        String nombreActual = estudiante.getNombre();
        TextView tvNombreActual = binding.tvNombreActualEst;
        tvNombreActual.setText(nombreActual);

        String codigoActual = estudiante.getCodigo();
        TextView tvCodigoActual = binding.tvCodigoActual;
        tvCodigoActual.setText(codigoActual);

        binding.btnEditarEst.setOnClickListener(v -> {
            String nuevoNombre = binding.inputNuevoNombre.getText().toString().trim();
            String nuevoCodigo = binding.inputNuevoCodigo.getText().toString().trim();

            // Verifica si no hay cambios
            if (nuevoNombre.isEmpty() && nuevoCodigo.isEmpty()) {
                Toast.makeText( context: this,  text: "No se detectaron cambios.", Toast.LENGTH_SHORT).show();
                return;
            }

            // Usa los valores actuales si los inputs están vacíos
            if (nuevoNombre.isEmpty()) nuevoNombre = estudiante.getNombre();
            if (nuevoCodigo.isEmpty()) nuevoCodigo = estudiante.getCodigo();

            // Verifica si el código ingresado ya está en uso por otro estudiante
            Estudiante estudianteExistente = estudianteController.obtenerEstudiantePorCodigo(nuevoCodigo);
            if (estudianteExistente != null && estudianteExistente.getId() != estudiante.getId()) {
                binding.inputNuevoCodigo.setError("Este código ya está en uso.");
                return;
            }

            // Llama al método para editar
            estudianteController.editarEstudiante(nuevoNombre, nuevoCodigo, estudiante.getId());
            Toast.makeText( context: this,  text: "Estudiante editado exitosamente.", Toast.LENGTH_SHORT).show();

            tvNombreActual.setText(nuevoNombre);
            tvCodigoActual.setText(nuevoCodigo);

            binding.inputNuevoNombre.setText("");
            binding.inputNuevoCodigo.setText("");
        });

        binding.btnVolverEditarEst.setOnClickListener(v->{
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        });
    }
}
```

```java
public class EditarNotaActivity extends AppCompatActivity {
    10 usages
    ActivityEditarNotaBinding binding;
    1 usage
    NotaController notaController = new NotaController( context: this);


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityEditarNotaBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        int notaId = getIntent().getIntExtra( name: "idNota",  defaultValue: -1);
        double notaActual = getIntent().getDoubleExtra( name: "notaActual",  defaultValue: -1);


        binding.tvNombreActualNota.setText(String.valueOf(notaActual));

        binding.btnEditarNota.setOnClickListener(v -> {
            String nuevaNotaStr = binding.inputNuevaNota.getText().toString().trim();

            if (nuevaNotaStr.isEmpty()) {
                binding.inputNuevaNota.setError("Ingresa una nota!");
                return;
            }

            double nuevaNota = Double.parseDouble(nuevaNotaStr);;

            if (nuevaNota < 1.0 || nuevaNota > 5.0) {
                binding.inputNuevaNota.setError("La nota debe estar entre 1.0 y 5.0.");
                return;
            }

            notaController.editarNota(notaId, nuevaNota);
            Toast.makeText(getApplicationContext(),  text: "Nota editada exitosamente.", Toast.LENGTH_SHORT)
                    .show();
            binding.tvNombreActualNota.setText(nuevaNotaStr);
            binding.inputNuevaNota.setText("");
        });



        binding.btnVolverEditarNota.setOnClickListener(v-> {
            Intent intent = new Intent( packageContext: this, DetallesEstudianteActivity.class);
            startActivity(intent);
        });
    }
}
```

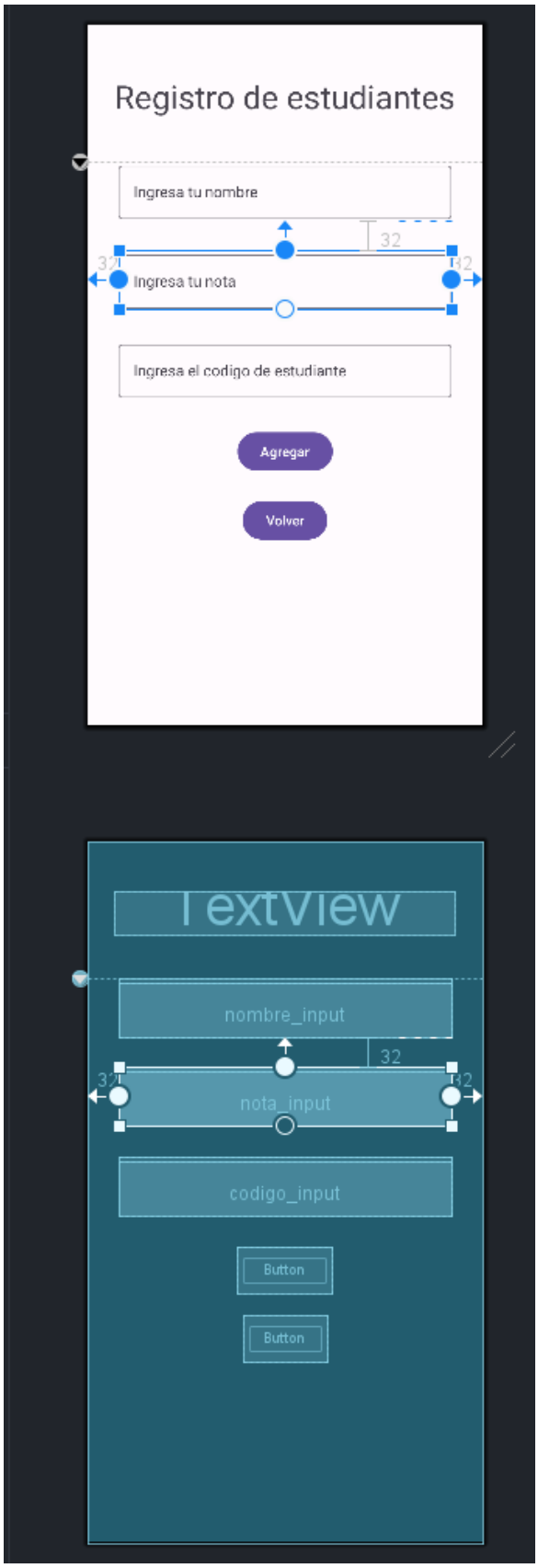
```java
public class EstudianteListaAdapter extends BaseAdapter {
    4 usages
    private List<Estudiante> estudiantes;
    3 usages
    private Context context;

    1 usage
    public EstudianteListaAdapter(List<Estudiante> estudiantes, Context context) {
        this.estudiantes = estudiantes;
        this.context = context;
    }

    @Override
    public int getCount() {
        return estudiantes.size();
    }

    @Override
    public Object getItem(int i) {
        return estudiantes.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }


    @Override
    public View getView(int i, View convertView, ViewGroup viewGroup) {
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(R.layout.item_estudiante,
                    viewGroup, attachToRoot: false);
        }

        // Obtener el estudiante de la lista
        Estudiante estudiante = estudiantes.get(i);
        NotaController notaController = new NotaController(context);

        TextView nombre = convertView.findViewById(R.id.notaNum);
        TextView codigo = convertView.findViewById(R.id.codigo);
        TextView promedio = convertView.findViewById(R.id.estudianteNota);

        // Asignar datos
        nombre.setText(estudiante.getNombre());
        codigo.setText(estudiante.getCodigo());

        //obtiene las notas por codigo del estudiante, luego de esa lista obtiene el promedio de notas

        double promedioFormat = notaController
                .calcularPromedio(notaController.obtenerNotasPorEstudiante(estudiante.getId()));
        promedio.setText(String.format("%.1f", promedioFormat));

        return convertView;
    }
}
```

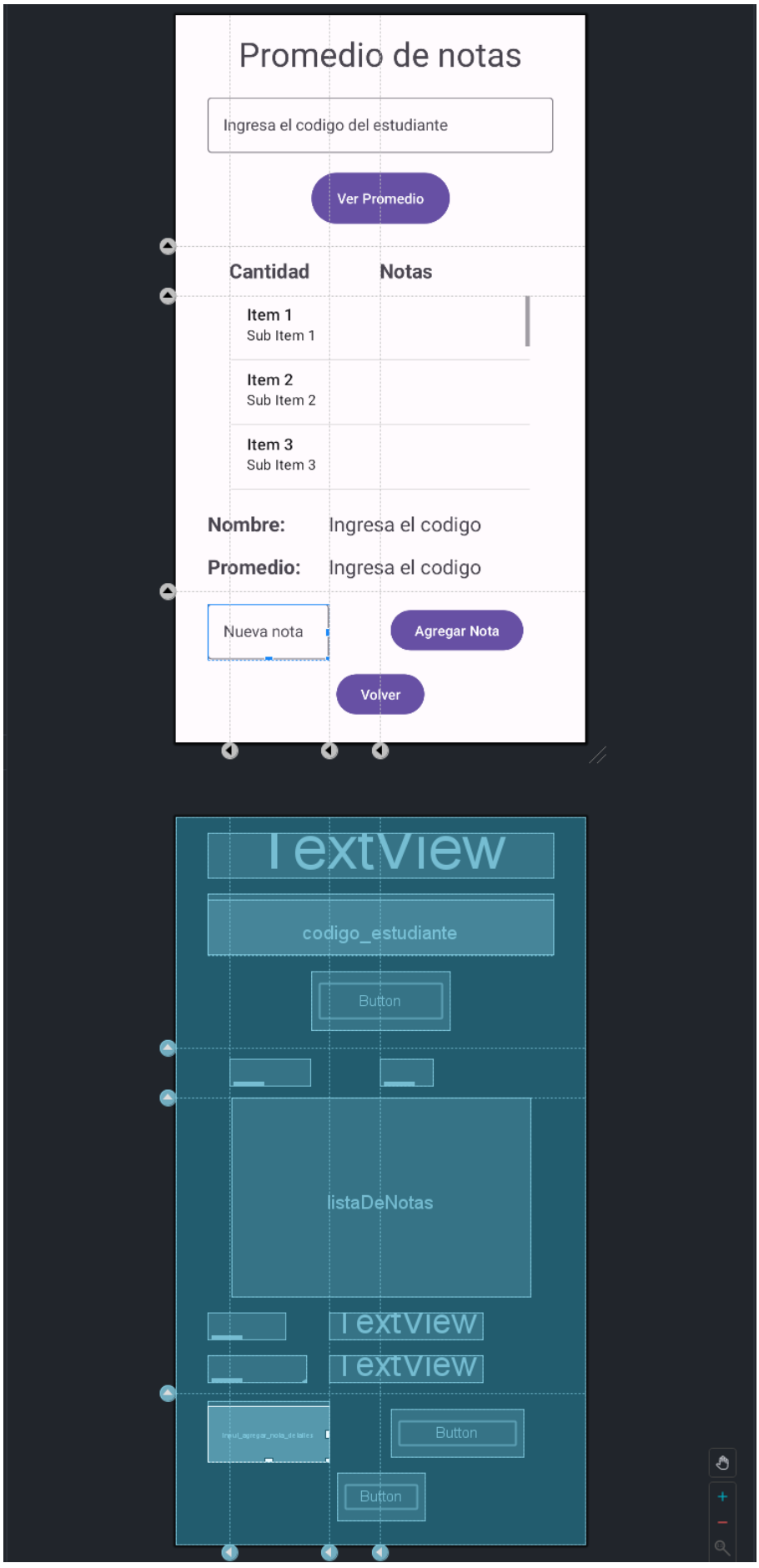
```java
public class MainActivity extends AppCompatActivity {
    5 usages
    private ActivityMainBinding binding;
    3 usages
    EstudianteController estudianteController = new EstudianteController( context: this);
    no usages
    NotaController notaController = new NotaController( context: this);

    6 usages
    List<Estudiante> estudiantes = new ArrayList<>();
    3 usages
    EstudianteListaAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView (binding.getRoot());

        estudiantes = estudianteController.obtenerEstudiantes();

        binding.agregarBtn.setOnClickListener(v -> {
            Intent intent = new Intent( packageContext: MainActivity.this, AgregarEstudiantesActivity.class);
            startActivity(intent);
        });
        //Asigna el adapter a la list view
        ListView lvListaEstudiantes = binding.listaEstudiantes;
        adapter = new EstudianteListaAdapter(estudiantes,  context: this);
        lvListaEstudiantes.setAdapter(adapter);

        binding.btnNotas.setOnClickListener(v->{
            Intent intent = new Intent ( packageContext: this, DetallesEstudianteActivity.class);
            startActivity(intent);
        });

        lvListaEstudiantes.setOnItemClickListener((parent, view, position, id)->{
            Estudiante estudianteSeleccionado = estudiantes.get(position);
            mostrarDialogoOpciones(estudianteSeleccionado);
        });

    }
    //Muestra dialogos de opciones
    1 usage
    private void mostrarDialogoOpciones(Estudiante estudiante) {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setTitle("Selecciona una acción")
                .setItems(new CharSequence[]{"Editar datos", "Eliminar"}, (dialog, which) -> {
                    if (which == 0) {
                        // Editar
                        Intent intent = new Intent( packageContext: this, EditarEstudianteActivity.class);
                        intent.putExtra( name: "codEstudiante", estudiante.getCodigo()); // pasar el codigo
                        System.out.println("Codigo estudiante: " + estudiante.getCodigo());
                        startActivity(intent);

                    } else if (which == 1) {
                        // Eliminar
                        mostrarDialogoConfirmacion(estudiante);
                    }
                })
                .show();
    }
```

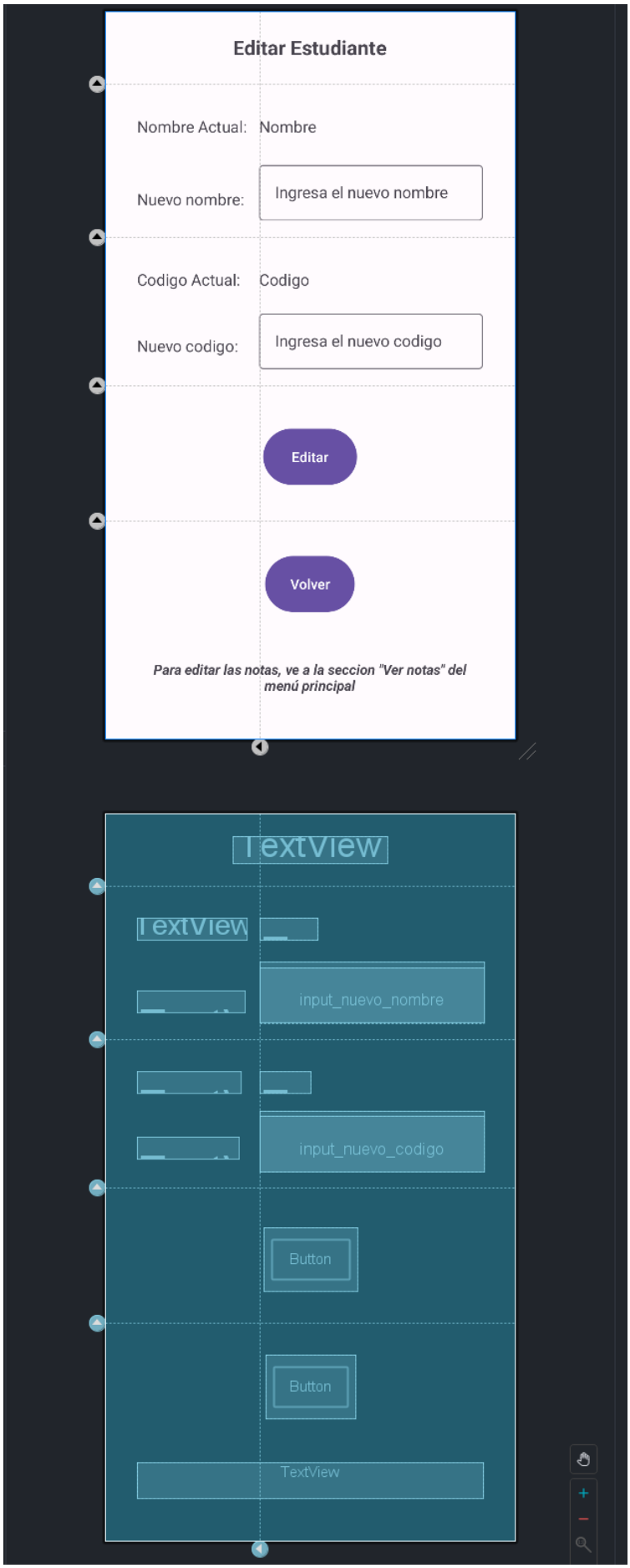
```
    private void mostrarDialogoConfirmacion(Estudiante estudiante) {
        new AlertDialog.Builder( context: this)
                .setTitle("Confirmar eliminación")
                .setMessage("¿Estás seguro de eliminar esta nota?")
                .setPositiveButton( text: "Sí", (dialog, which) -> {
                    estudianteController.eliminarEstudiante(estudiante.getId());
                    estudiantes.remove(estudiante);

                    estudiantes.clear();
                    estudiantes.addAll(estudianteController.obtenerEstudiantes());
                    adapter.notifyDataSetChanged(); // actualiza la lista

                    Toast.makeText( context: this,  text: "Estudiante eliminado!", Toast.LENGTH_SHORT).show();
                })
                .setNegativeButton( text: "Cancelar",  listener: null)
                .show();
    }
}
```

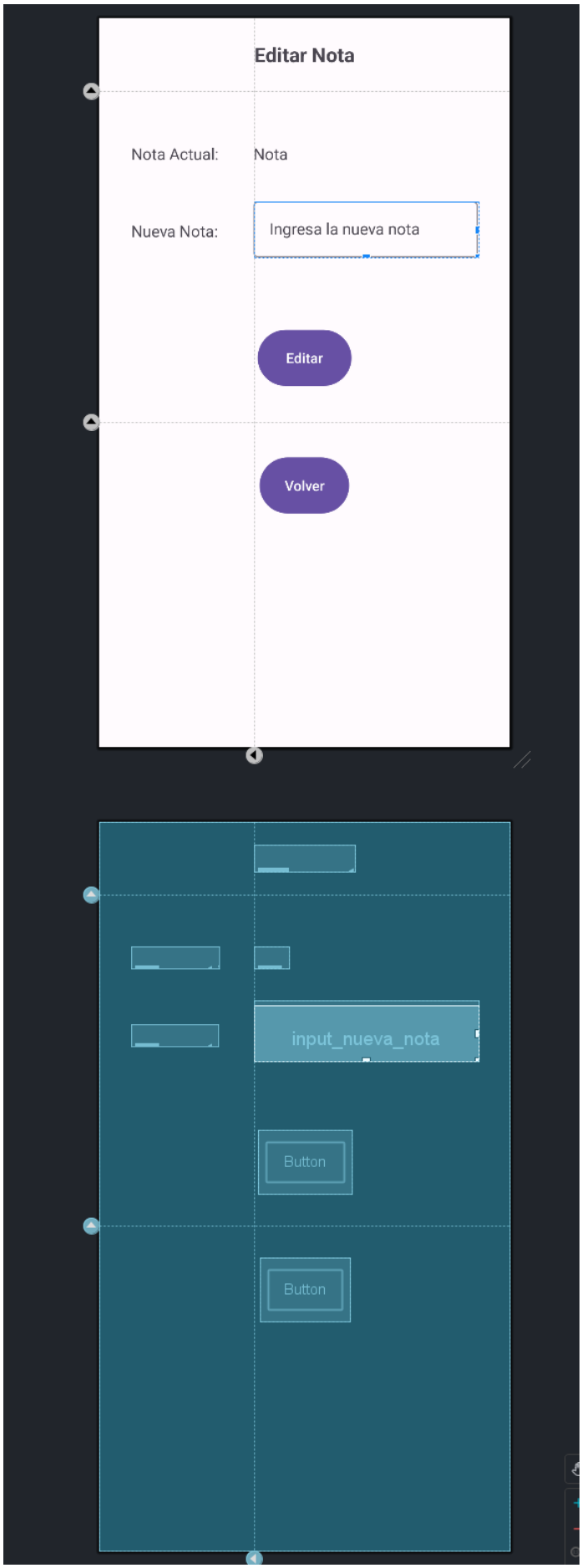
```
public class NotaListaAdapter extends BaseAdapter {
    2 usages
    private Context context;
    4 usages
    private List<Nota> notas;

    1 usage
    public NotaListaAdapter(Context context, List<Nota> notas) {
        this.context = context;
        this.notas = notas;
    }

    @Override
    public int getCount() {
        return notas.size();
    }

    @Override
    public Object getItem(int i) {
        return notas.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View convertView, ViewGroup viewGroup) {

        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(R.layout.item_notas, viewGroup,  attachToRoot: false);
        }

        Nota nota = notas.get(i);
        TextView numeroNota = convertView.findViewById(R.id.notaNum);
        TextView valorNota = convertView.findViewById(R.id.estudianteNota);

        numeroNota.setText(String.valueOf(getItemId( i: i+1)));
        valorNota.setText(String.valueOf(nota.getValor()));
        return convertView;
    }
}
```

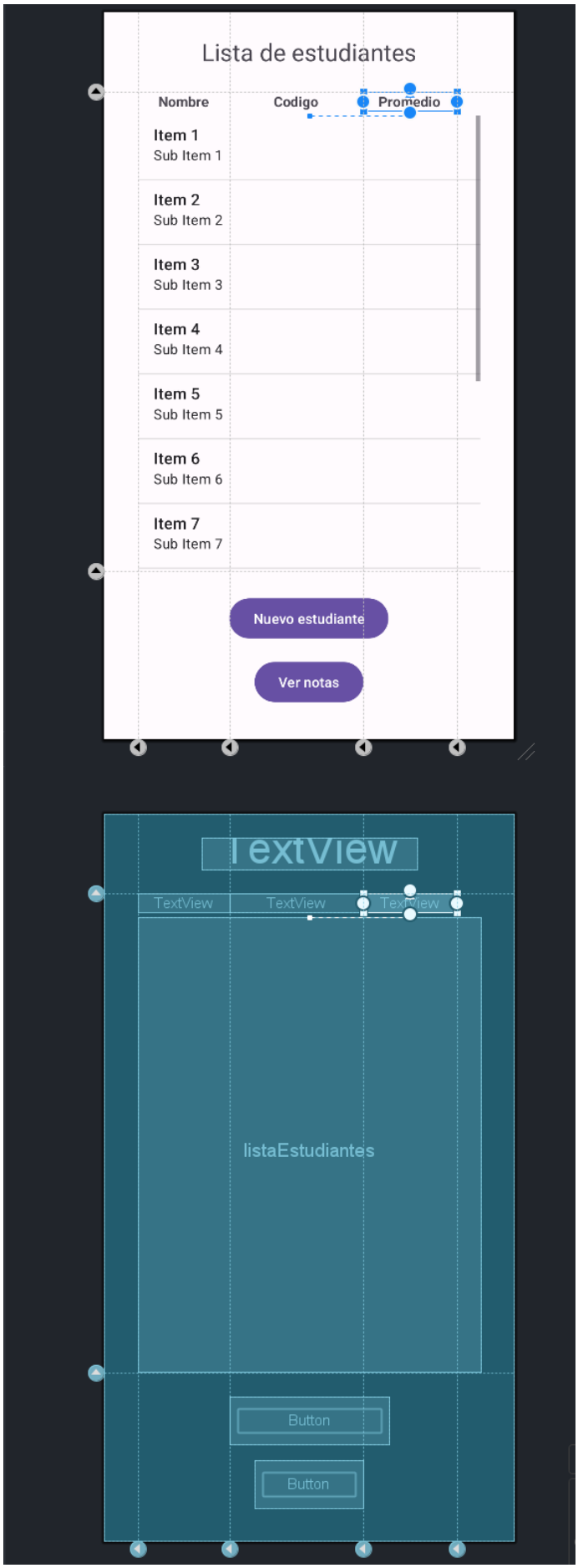
FRONTEND


Registro de estudiantes
Ingresa tu nombre
32
32
Ingresa tu nota
32
Ingresa el codigo de estudiante
Agregar
Volver
TextView
nombre_input
32
32
nota_input
32
codigo_input
Button
Button

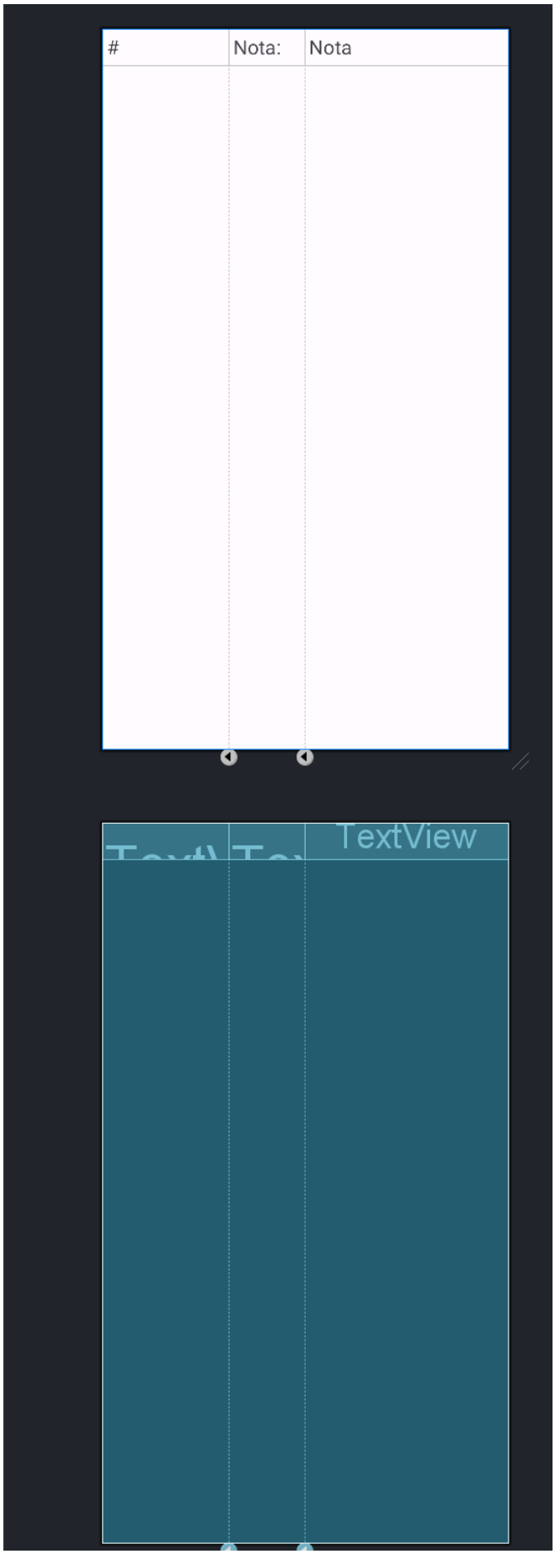

Promedio de notas
Ingresa el codigo del estudiante
Ver Promedio
Cantidad
Notas
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Nombre:
Ingresa el codigo
Promedio:
Ingresa el codigo
Nueva nota
Agregar Nota
Volver
TextView
codigo_estudiante
Button
listaDeNotas
TextView
TextView
Button
Button


Editar Estudiante
Nombre Actual: Nombre
Nuevo nombre: Ingresa el nuevo nombre
Codigo Actual: Codigo
Nuevo codigo: Ingresa el nuevo codigo
Editar
Volver
Para editar las notas, ve a la seccion "Ver notas" del menú principal
TextView
TextView
input_nuevo_nombre
input_nuevo_codigo
Button
Button
TextView


Editar Nota
Nota Actual:
Nota
Nueva Nota:
Ingresa la nueva nota
Editar
Volver
input_nueva_nota
Button
Button


Lista de estudiantes
Nombre
Codigo
Promedio
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5
Item 6
Sub Item 6
Item 7
Sub Item 7
Nuevo estudiante
Ver notas
TextView
TextView
TextView
TextView
listaEstudiantes
Button
Button


Nombre
Codigo
Nota
TextView


#
Nota:
Nota
TextView

REFERENCIAS

- https://developer.android.com/reference/android/widget/ListView
- https://www.youtube.com/watch?v=CCBno9jX3jI&ab_channel=ProgramadorNovato
- https://poo-y-sqlite-en-java-and-jlalf2r.gamma.site/