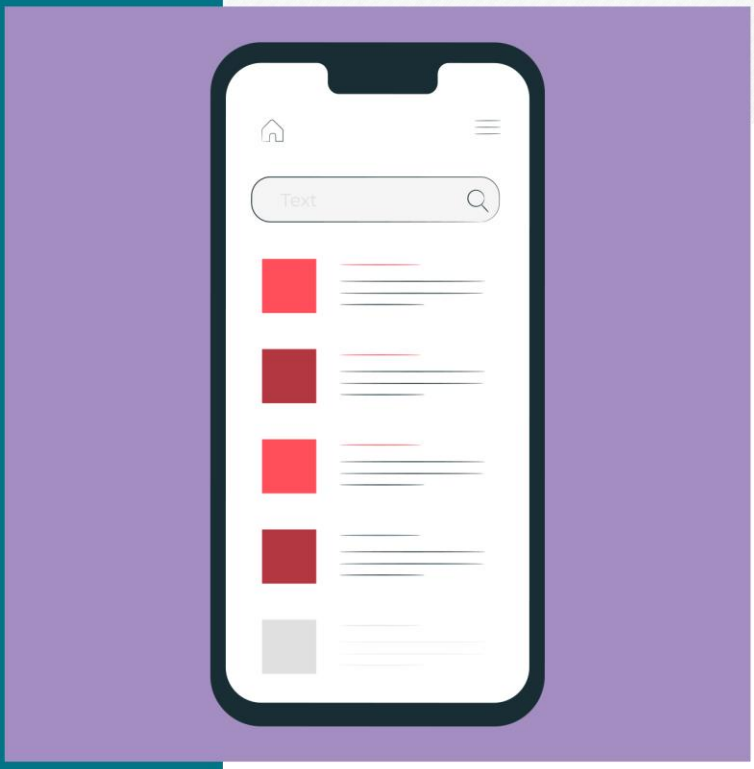
AGENDA 3

MANIPULAÇÃO DE BANCO DE DADOS LOCAL NO DISPOSITIVO



GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO MOBILE I

Expediente

Autor:

GUILHERME HENRIQUE GIROLLI

Revisão Técnica:

Eliana Cristina Nogueira Barion

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

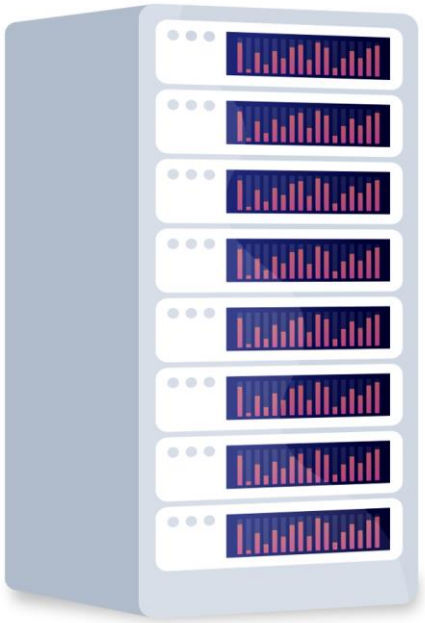
Editoração e Diagramação:

Flávio Biazim

São Paulo – SP, 2021

MERGULHANDO NO TEMA...

Banco de Dados (BD)



O Banco de dados é uma peça fundamental para a maioria dos sistemas computacionais que envolve no seu funcionamento a escrita, leitura, alteração e exclusão de dados. Segundo Korth, o banco de dados é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico.

Desta forma podemos comparar um BD com as antigas agendas de papel, que eram organizadas por ordem alfabética para facilitar a inclusão e recuperação das informações, e cada registro que era escrito nas folhas dessa agenda era composto de dados sobre uma determinada pessoa ou empresa.

Para o armazenamento digital de dados, é necessário a construção de uma estrutura composta por um arquivo denominado de Banco de Dados, que abriga em seu interior as tabelas desenvolvidas com colunas para receber os dados.

Quando falamos que um programa armazena dados, na verdade quem é responsável por essa etapa é um software que trabalha nos bastidores, e que abriga o BD previamente desenvolvido para oferecer de maneira organizada essa função.

O Sistema Gerenciador de Banco de Dados (SGBD), é um software que é executado em um ambiente computacional, é nesse sistema que encontramos e armazenamos o Banco de Dados.

Esse SGBD tem a função de controlar o acesso ao Banco de Dados, e garantir o perfeito funcionamentos no processo manipulação dos dados e da estrutura do BD.

SQLite

o sistema operacional Android encontramos a presença do SQLite, ele possui várias características de um gerenciador de banco, porém não podemos dizer que ele é um SGBD. Em seu site oficial, disponível através do link: https://www.sqlite.org/index.html, eles denominam SQLite como um mecanismo de BD implementado por classes que foram desenvolvidas na linguagem de programação C.

A diferença do SQLite para um tradicional SGBD é que ele não requer instalação e complexas configurações, dessa forma sua utilização se torna simples pelos desenvolvedores. Atualmente o SQLite é encontrado nos dispositivos com sistema operacional Android, em alguns computadores e em outros dispositivos.

Figura 4 – Logo SQLite. Fonte: https://www.sqlite.org/index.html

Structured Query Language (SQL)

O SQLite utiliza a Structured Query Language (SQL) ou Linguagem de Consulta Estruturada. Essa poderosa linguagem é basicamente dividida em dois grupos. O grupo de comandos de definição ou construção do BD e das Tabelas. E o grupo responsável por manipular os dados nesse BD, seja essa manipulação uma inclusão de registros ou leitura, como também uma alteração de dados ou exclusão.

Os principais comandos da **Data Definition Language (DDL)** ou Linguagem de Definição de Dados utilizados nesse projeto "AppCofre" são:

CREATE DATABASE: Responsável por criar o Banco de Dados.

CREATE TABLE: Responsável por criar uma tabela dentro do BD para abrigar os dados de maneira organizada.

Os principais comandos da **Data Manipulation Language (DML)** ou Linguagem de Manipulação de Dados utilizados nesse projeto "AppCofre" são:

INSERT: Responsável por inserir dados de maneira organizada em uma determinada tabela dentro do banco de dados.

UPDATE: Responsável por alterar dados que já foram inseridos em uma determinada tabela.

DELETE: Responsável por deletar dados que estão inseridos em uma determinada tabela.

SELECT: Responsável por ler dados que estão inseridos em uma determinada tabela.

Ao utilizar o SQLite, o desenvolvedor vai encontrar uma experiência diferente de utilização do SQL com a linguagem Java, a classe que efetua toda a parte de criação e comunicação com o BD possui métodos já prontos que facilita toda essa fase de integração do aplicativo com o BD.

SQLiteOpenHelper

No Android Studio vamos trabalhar com uma classe chamada SQLiteOpenHelper, ela vai nos auxiliar em todo o trabalho complexo de criação de um BD utilizando o SQLite. Conforme o material disponível no site oficial do Android Studio disponível no link: https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper, o SQLiteOpenHelper é uma classe auxiliar para gerenciar a criação de banco de dados e o gerenciamento de versão.

Você cria uma classe implementando os métodos **onCreate** que é responsável por criar a estrutura do banco quando essa ainda não existir, e o **onUpgrade** que atualiza a estrutura do banco quando isso for necessário. Essas implementações são usadas para garantir que o banco de dados funcione sempre em um estado sensato e integro, de acordo com a necessidade do aplicativo.

Essa classe facilita a implementação e a atualização do banco de dados até a primeira execução, a fim de evitar o bloqueio da inicialização do aplicativo.

O SQLiteOpenHelper é uma ótima opção quando o assunto é manipulação de banco de dados local no dispositivo. Para facilitar o uso do SQLite no projeto, o "AppCofre" foi desenvolvido utilizando alguns conceitos do padrão MVC de programação, e agora vamos finalizar o projeto inserindo os códigos responsáveis pela integração do aplicativo com o SQLite.

VOCÊ NO COMANDO

Inserindo o BD no AppCofre

Anteriormente desenvolvemos as camadas **"Model"**, **"View"** e **"Controller"** do nosso projeto e agora é necessário desenvolver mais um modelo de negócio para trabalhar com BD, ou seja, desenvolver o "modelo" de BD que será utilizado pelo nosso aplicativo.

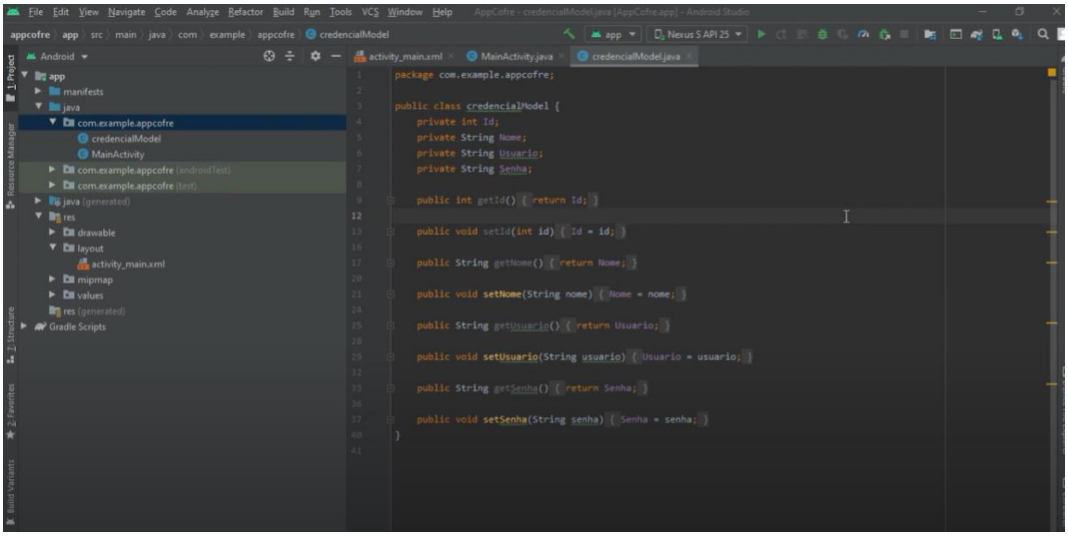
Vamos desenvolver mais uma classe em nosso projeto, ela será uma subclasse da classe "**SQLiteOpenHelper",** e vamos colocar o nome de **"bdModel",** nela vamos desenvolver todos os atributos e métodos responsáveis por criar e manipular nosso BD.

Como a classe **“bdModel”** é uma subclasse da classe “**SQLiteOpenHelper”,** vamos criar um construtor para a **“bdModel”** utilizando um “super” para construir a superclasse “**SQLiteOpenHelper”.**

Durante essa construção vamos instanciar um objeto do tipo **“SQLiteDatabase”,** ele vai ser responsável por manipular os dados do BD desenvolvido pelo “**SQLiteOpenHelper”.**

É importante informar que o objeto do tipo **“SQLiteDatabase”,** recebe o **“getWritableDatabase()”** que “Cria” e/ou “Abre” um banco de dados que será usado para leitura e/ou escrita de dados.

Assista o vídeo a seguir para auxiliar no desenvolvimento da classe **“bdModel”.** Verifique após o vídeo como ficou os códigos da classe **“bdModel”.**



Vídeo 1 – Desenvolvimento da classe “bdModel”. Link: https://youtu.be/HDdrJdzhYaA

```
package com.example.appcofre;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class bdModel extends SQLiteOpenHelper {

    SQLiteDatabase dataBase;

    public bdModel(Context context) {
        super(context, getDbNome(), null, getDbVersao());
        dataBase = getWritableDatabase();
    }

    private static String dbNome = "dbCredencial";
    private static int dbVersao = 1;
```

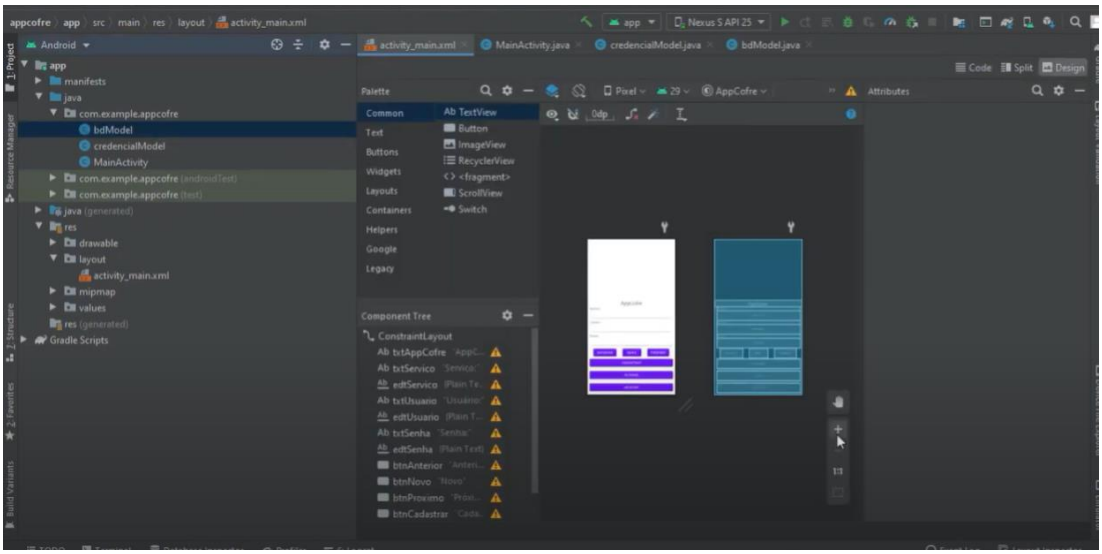
```
    private static String Tabela = "tblCredencial";
    private static String Id = "idCredencial";
    private static String Nome = "nomeCredencial";
    private static String Usuario = "usuarioCredencial";
    private static String Senha = "senhaCredencial";
    private String CmdSQL = "";


    public static String getDbNome() {
        return dbNome;
    }

    public static int getDbVersao() {
        return dbVersao;
    }

    public static String getTabela() {
        return Tabela;
    }

    public static String getId() {
        return Id;
    }

    public static String getNome() {
        return Nome;
    }

    public static String getUsuario() {
        return Usuario;
    }

    public static String getSenha() {
        return Senha;
    }

    public String getCmdSQL() {
        return CmdSQL;
    }

    public void setCmdSQL(String cmdSQL) {
        CmdSQL = cmdSQL;
    }

    public String criarTabela(){
        setCmdSQL("CREATE TABLE " + getTabela() + " (" +
                getId() + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                getNome() + " TEXT, " +
                getUsuario() + " TEXT, " +
                getSenha() + " TEXT" +
                ")"
        );
        return getCmdSQL();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(criarTabela());
```

```
    }

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

    }
}
```

Após o desenvolvimento do modelo do nosso BD, vamos continuar a codificação do nosso projeto com a etapa de inserção de dados na tabela. Vamos desenvolver no **"bdModel"** uma estrutura para receber um modelo de credencial e através de um comando do **"SQLiteDatabase"** armazenar os dados na tabela **"tblCredencial".**

Nesta etapa vamos contar com a ajuda do "ContentValues" que é uma classe usada para armazenar um conjunto de valores.

Assista o vídeo a seguir para continuar o desenvolvimento da classe **"bdModel".**



Vídeo 2 – Desenvolvimento da classe "bdModel". Link: https://youtu.be/SbWvadVGKEc

Verifique após o vídeo como ficou os códigos da classe **"bdModel".**

```
package com.example.appcofre;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class bdModel extends SQLiteOpenHelper {
```

```
    SQLiteDatabase dataBase;

    public bdModel(Context context) {
        super(context, getDbNome(), null, getDbVersao());
        dataBase = getWritableDatabase();
    }

    private static String dbNome = "dbCredencial";
    private static int dbVersao = 1;
    private static String Tabela = "tblCredencial";
    private static String Id = "idCredencial";
    private static String Nome = "nomeCredencial";
    private static String Usuario = "usuarioCredencial";
    private static String Senha = "senhaCredencial";
    private String CmdSQL = "";


    public static String getDbNome() {
        return dbNome;
    }

    public static int getDbVersao() {
        return dbVersao;
    }

    public static String getTabela() {
        return Tabela;
    }

    public static String getId() {
        return Id;
    }

    public static String getNome() {
        return Nome;
    }

    public static String getUsuario() {
        return Usuario;
    }

    public static String getSenha() {
        return Senha;
    }

    public String getCmdSQL() {
        return CmdSQL;
    }

    public void setCmdSQL(String cmdSQL) {
        CmdSQL = cmdSQL;
    }

    public String criarTabela(){
        setCmdSQL("CREATE TABLE " + getTabela() + " (" +
                getId() + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                getNome() + " TEXT, " +
                getUsuario() + " TEXT, " +
```

```
                getSenha() + " TEXT" +
                ")"
        );
        return getCmdSQL();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(criarTabela());
    }

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

    }

    public void insert(String tabela, credencialModel credencial)
    {
        ContentValues dados = new ContentValues();
        dados.put(getNome(), credencial.getNome());
        dados.put(getUsuario(), credencial.getUsuario());
        dados.put(getSenha(), credencial.getSenha());
        dataBase.insert(tabela,null, dados);
    }
}
```

Verifique após o vídeo como ficou os códigos da classe **“MainActivity.java”.**

```
package com.example.appcofre;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView txtServicoProg;
    EditText edtNomeProg;
    EditText edtUsuarioProg;
    EditText edtSenhaProg;

    credencialModel credencial = new credencialModel();

    bdModel bd;

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

        txtServicoProg = (TextView) findViewById(R.id.txtServico);
        edtNomeProg = (EditText) findViewById(R.id.edtServico);
        edtUsuarioProg = (EditText) findViewById(R.id.edtUsuario);
        edtSenhaProg = (EditText) findViewById(R.id.edtSenha);
```

```
    }

    public void clickBtnDeletar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public void clickBtnAlterar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public  void clickBtnCadastrar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
        bd = new bdModel(getApplicationContext());
        bd.insert(bdModel.getTabela(), credencial);
    }

    public  void clickBtnNovo(View v)
    {
        limpar();
    }

    public void limpar()
    {
        edtNomeProg.setText("");
        edtUsuarioProg.setText("");
        edtSenhaProg.setText("");
        txtServicoProg.setText("Serviço:");
        edtNomeProg.requestFocus();
    }
}
```

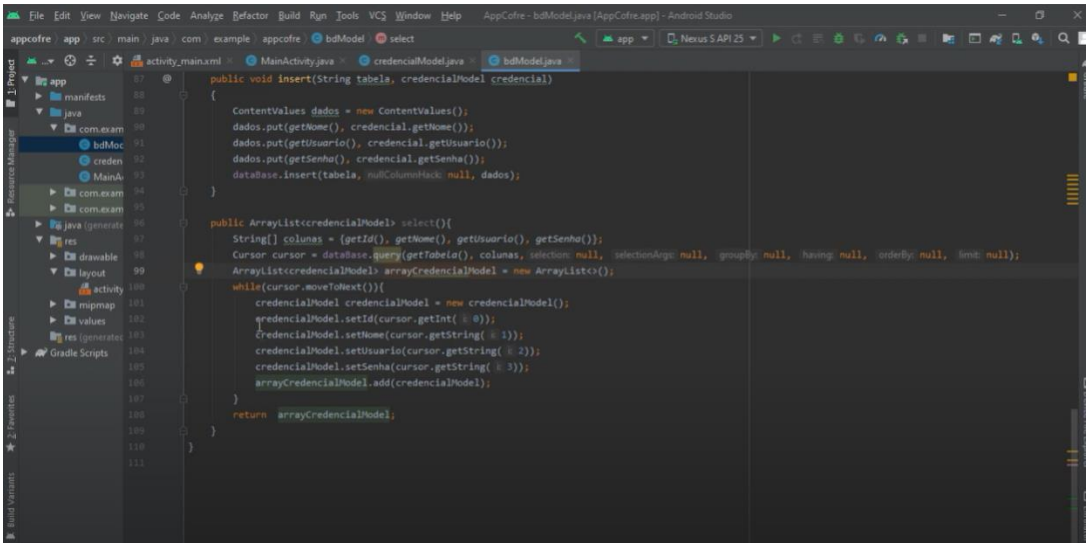
Depois de efetuar a etapa de inserção de dados através do **“bdModel”** no BD, vamos desenvolver a etapa inversa, ou seja, vamos efetuar o processo de ler os valores do BD e exibir em nossa camada **“View”.**

Utilizando um método da classe **“SQLiteDatabase”** para ler os valores que estão na tabela **“tblCredencial”** e com rotinas desenvolvidas da pseudocamada **“Controller”** vamos processar esses dados para construir uma informação na tela do usuário.

Vamos trabalhar com o auxílio do **“ArrayList”** que é uma estrutura que forma uma lista com conjuntos de dados.

É importante registrar a função do **“Cursor”**, que é uma interface que fornece acesso aleatório de leitura e gravação ao conjunto de resultados retornado por uma consulta ao banco de dados. Vamos utilizar essa interface para inserir no **“ArrayList”** os componentes com dados de cada credencial.

Assista o vídeo a seguir a explicação do processo de leitura de dados de um BD**.**



Vídeo 3 – Processo de leitura de dados de um BD utilizando as classes “bdModel” e “MainActivity”. Link: c

Verifique após o vídeo como ficou os códigos da classe **“bdModel”.**

```
package com.example.appcofre;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

public class bdModel extends SQLiteOpenHelper {

    SQLiteDatabase dataBase;

    public bdModel(Context context) {
        super(context, getDbNome(), null, getDbVersao());
        dataBase = getWritableDatabase();
    }

    private static String dbNome = "dbCredencial";
    private static int dbVersao = 1;
    private static String Tabela = "tblCredencial";
    private static String Id = "idCredencial";
    private static String Nome = "nomeCredencial";
```

```
    private static String Usuario = "usuarioCredencial";
    private static String Senha = "senhaCredencial";
    private String CmdSQL = "";


    public static String getDbNome() {
        return dbNome;
    }

    public static int getDbVersao() {
        return dbVersao;
    }

    public static String getTabela() {
        return Tabela;
    }

    public static String getId() {
        return Id;
    }

    public static String getNome() {
        return Nome;
    }

    public static String getUsuario() {
        return Usuario;
    }

    public static String getSenha() {
        return Senha;
    }

    public String getCmdSQL() {
        return CmdSQL;
    }

    public void setCmdSQL(String cmdSQL) {
        CmdSQL = cmdSQL;
    }

    public String criarTabela(){
        setCmdSQL("CREATE TABLE " + getTabela() + " (" +
                getId() + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                getNome() + " TEXT, " +
                getUsuario() + " TEXT, " +
                getSenha() + " TEXT" +
                ")"
        );
        return getCmdSQL();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(criarTabela());
    }

    @Override
```

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

    }

    public void insert(String tabela, credencialModel credencial)
    {
        ContentValues dados = new ContentValues();
        dados.put(getNome(), credencial.getNome());
        dados.put(getUsuario(), credencial.getUsuario());
        dados.put(getSenha(), credencial.getSenha());
        dataBase.insert(tabela,null, dados);
    }

    public ArrayList<credencialModel> select(){
        String[] colunas = {getId(), getNome(), getUsuario(), getSenha()};
        Cursor cursor = dataBase.query(getTabela(), colunas,null, null, null, null, null,
null);
        ArrayList<credencialModel> arrayCredencialModel = new ArrayList<>();
        while(cursor.moveToNext()){
            credencialModel credencialModel = new credencialModel();
            credencialModel.setId(cursor.getInt(0));
            credencialModel.setNome(cursor.getString(1));
            credencialModel.setUsuario(cursor.getString(2));
            credencialModel.setSenha(cursor.getString(3));
            arrayCredencialModel.add(credencialModel);
        }
        return  arrayCredencialModel;
    }
}
```

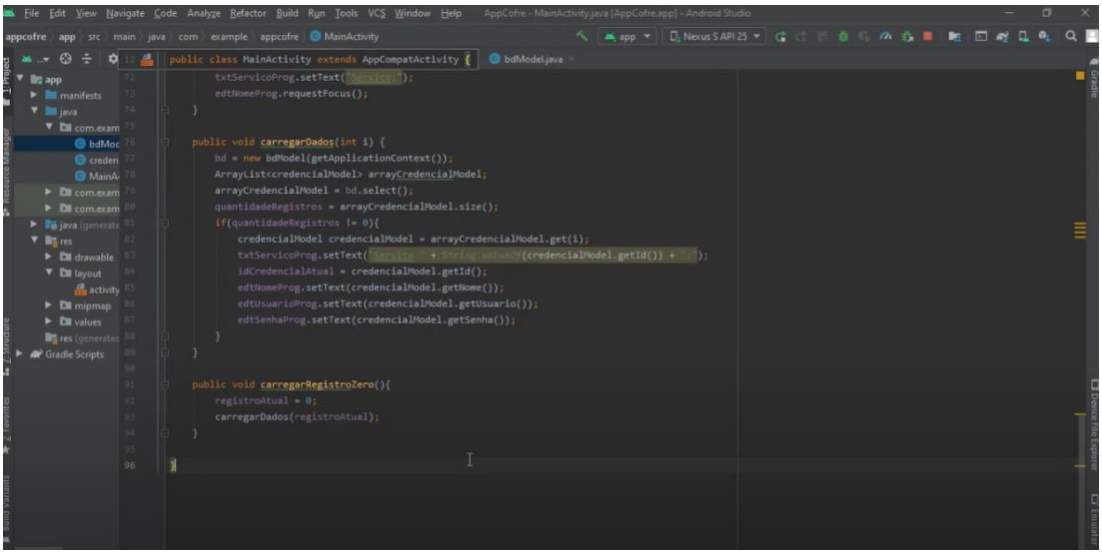
Verifique após o vídeo como ficou os códigos da classe **“MainActivity.java”.**

```
package com.example.appcofre;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    TextView txtServicoProg;
    EditText edtNomeProg;
    EditText edtUsuarioProg;
    EditText edtSenhaProg;
    int quantidadeRegistros;
    int registroAtual;
    int idCredencialAtual;

    credencialModel credencial = new credencialModel();
```

```
    bdModel bd;

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

        txtServicoProg = (TextView) findViewById(R.id.txtServico);
        edtNomeProg = (EditText) findViewById(R.id.edtServico);
        edtUsuarioProg = (EditText) findViewById(R.id.edtUsuario);
        edtSenhaProg = (EditText) findViewById(R.id.edtSenha);

        carregarRegistroZero();
    }

    public void clickBtnDeletar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public void clickBtnAlterar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public  void clickBtnCadastrar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
        bd = new bdModel(getApplicationContext());
        bd.insert(bdModel.getTabela(), credencial);
    }

    public  void clickBtnNovo(View v)
    {
        limpar();
    }

    public void limpar()
    {
        edtNomeProg.setText("");
        edtUsuarioProg.setText("");
        edtSenhaProg.setText("");
        txtServicoProg.setText("Serviço:");
        edtNomeProg.requestFocus();
    }

    public void carregarDados(int i) {
        bd = new bdModel(getApplicationContext());
        ArrayList<credencialModel> arrayCredencialModel;
        arrayCredencialModel = bd.select();
        quantidadeRegistros = arrayCredencialModel.size();
        if(quantidadeRegistros != 0){
```

```
            credencialModel credencialModel = arrayCredencialModel.get(i);
            txtServicoProg.setText("Serviço " + String.valueOf(credencialModel.getId()) +
":");
            idCredencialAtual = credencialModel.getId();
            edtNomeProg.setText(credencialModel.getNome());
            edtUsuarioProg.setText(credencialModel.getUsuario());
            edtSenhaProg.setText(credencialModel.getSenha());
        }
    }

    public void carregarRegistroZero(){
        registroAtual = 0;
        carregarDados(registroAtual);
    }

}
```

Aproveitando que o conteúdo sobre leitura de dados de um BD está bem fresco, vamos desenvolver a função dos botões **"anterior"** e **"próximo".** No vídeo a seguir encontramos o desenvolvimento dos métodos para as funções dos botões de navegação.



Vídeo 4 – Processo de desenvolvimento dos botões de navegação do aplicativo. Link: https://youtu.be/hEJs6IuVaYY

Verifique após o vídeo como ficou os códigos da classe **"MainActivity.java".**

```
package com.example.appcofre;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
```

```
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    TextView txtServicoProg;
    EditText edtNomeProg;
    EditText edtUsuarioProg;
    EditText edtSenhaProg;
    int quantidadeRegistros;
    int registroAtual;
    int idCredencialAtual;

    credencialModel credencial = new credencialModel();

    bdModel bd;

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

        txtServicoProg = (TextView) findViewById(R.id.txtServico);
        edtNomeProg = (EditText) findViewById(R.id.edtServico);
        edtUsuarioProg = (EditText) findViewById(R.id.edtUsuario);
        edtSenhaProg = (EditText) findViewById(R.id.edtSenha);

        carregarRegistroZero();
    }

    public void clickBtnDeletar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public void clickBtnAlterar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
    }

    public  void clickBtnCadastrar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
        bd = new bdModel(getApplicationContext());
        bd.insert(bdModel.getTabela(), credencial);
        carregarRegistroZero();
    }

    public  void clickBtnNovo(View v)
    {
        limpar();
    }
```

```
    public void clickBtnAnterior(View v)
    {
        if(quantidadeRegistros != 0)
        {
            if(registroAtual > 0)
            {
                registroAtual = registroAtual - 1;
                carregarDados(registroAtual);
            }
        }
    }

    public void clickBtnProximo(View v)
    {
        if(quantidadeRegistros != 0)
        {
            if(registroAtual < quantidadeRegistros - 1)
            {
                registroAtual = registroAtual + 1;
                carregarDados(registroAtual);
            }
        }
    }

    public void limpar()
    {
        edtNomeProg.setText("");
        edtUsuarioProg.setText("");
        edtSenhaProg.setText("");
        txtServicoProg.setText("Serviço:");
        edtNomeProg.requestFocus();
    }

    public void carregarDados(int i) {
        bd = new bdModel(getApplicationContext());
        ArrayList<credencialModel> arrayCredencialModel;
        arrayCredencialModel = bd.select();
        quantidadeRegistros = arrayCredencialModel.size();
        if(quantidadeRegistros != 0){
            credencialModel credencialModel = arrayCredencialModel.get(i);
            txtServicoProg.setText("Serviço " + String.valueOf(credencialModel.getId()) +
":");
            idCredencialAtual = credencialModel.getId();
            edtNomeProg.setText(credencialModel.getNome());
            edtUsuarioProg.setText(credencialModel.getUsuario());
            edtSenhaProg.setText(credencialModel.getSenha());
        }
    }

    public void carregarRegistroZero(){
        registroAtual = 0;
        carregarDados(registroAtual);
    }

}
```

Verifique após o vídeo como ficou os códigos do arquivo XML **“activity_main.xml”.**

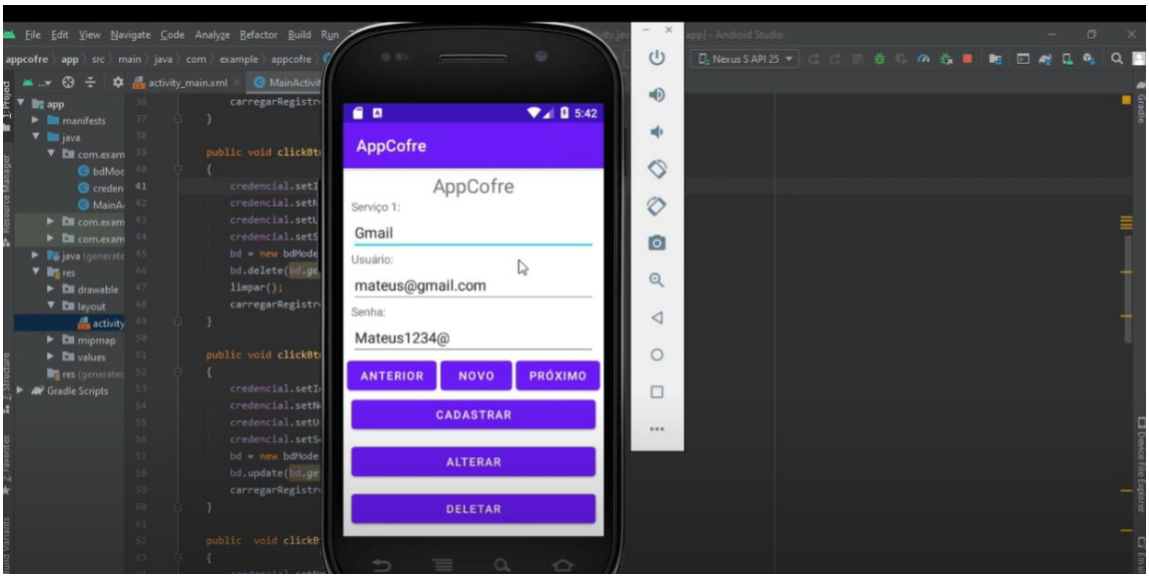
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtAppCofre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="AppCofre"
        android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@id/txtServico"
        android:textAlignment="center"
        />

    <TextView
        android:id="@+id/txtServico"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Serviço:"
        android:layout_marginHorizontal="10dp"
        app:layout_constraintBottom_toTopOf="@id/edtServico"
        />

    <EditText
        android:id="@+id/edtServico"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10dp"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@id/txtUsuario"
        />

    <TextView
        android:id="@+id/txtUsuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Usuário:"
        android:layout_marginHorizontal="10dp"
        app:layout_constraintBottom_toTopOf="@id/edtUsuario"
        />

    <EditText
        android:id="@+id/edtUsuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10dp"
        android:inputType="textPersonName"
```

```
        app:layout_constraintBottom_toTopOf="@id/txtSenha"
        />

    <TextView
        android:id="@+id/txtSenha"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Senha:"
        android:layout_marginHorizontal="10dp"
        app:layout_constraintBottom_toTopOf="@id/edtSenha"
        />

    <EditText
        android:id="@+id/edtSenha"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10dp"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@id/btnAnterior"
        />

    <Button
        android:id="@+id/btnAnterior"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickBtnAnterior"
        android:text="Anterior"
        app:layout_constraintBottom_toTopOf="@id/btnCadastrar"
        app:layout_constraintEnd_toStartOf="@+id/btnNovo"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btnNovo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickBtnNovo"
        android:text="Novo"
        app:layout_constraintBottom_toTopOf="@id/btnCadastrar"
        app:layout_constraintEnd_toStartOf="@+id/btnProximo"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnAnterior" />

    <Button
        android:id="@+id/btnProximo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickBtnProximo"
        android:text="Próximo"
        app:layout_constraintBottom_toTopOf="@id/btnCadastrar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnNovo" />

    <Button
        android:id="@+id/btnCadastrar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_margin="10dp"
        android:onClick="clickBtnCadastrar"
        android:text="Cadastrar"
        app:layout_constraintBottom_toTopOf="@id/btnAlterar"
        tools:layout_editor_absoluteX="10dp" />

    <Button
        android:id="@+id/btnAlterar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:onClick="clickBtnAlterar"
        android:text="Alterar"
        app:layout_constraintBottom_toTopOf="@id/btnDeletar"
        tools:layout_editor_absoluteX="10dp" />

    <Button
        android:id="@+id/btnDeletar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:onClick="clickBtnDeletar"
        android:text="Deletar"
        app:layout_constraintBottom_toBottomOf="parent"
        tools:layout_editor_absoluteX="10dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Para alterar ou deletar uma credencial vamos utilizar alguns comandos anteriormente apresentados no processo de inserir dados na tabela do BD. O que vai mudar são os comandos utilizados na camada **“Model”** mais precisamente no **“SQLiteDataBase”.**

No vídeo a seguir encontramos o desenvolvimento dos métodos para as funções dos botões de alterar e deletar do projeto.



Vídeo 5 – Processo de desenvolvimento dos botões de alterar e deletar. Link: https://youtu.be/_A_giTR3goc

Verifique após o vídeo como ficou os códigos da classe **“bdModel”.**

```
package com.example.appcofre;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

public class bdModel extends SQLiteOpenHelper {

    SQLiteDatabase dataBase;

    public bdModel(Context context) {
        super(context, getDbNome(), null, getDbVersao());
        dataBase = getWritableDatabase();
    }

    private static String dbNome = "dbCredencial";
    private static int dbVersao = 1;
    private static String Tabela = "tblCredencial";
    private static String Id = "idCredencial";
    private static String Nome = "nomeCredencial";
    private static String Usuario = "usuarioCredencial";
    private static String Senha = "senhaCredencial";
    private String CmdSQL = "";


    public static String getDbNome() {
        return dbNome;
    }

    public static int getDbVersao() {
        return dbVersao;
    }

    public static String getTabela() {
        return Tabela;
    }

    public static String getId() {
        return Id;
    }

    public static String getNome() {
        return Nome;
    }

    public static String getUsuario() {
        return Usuario;
    }

    public static String getSenha() {
        return Senha;
    }
```

```
    public String getCmdSQL() {
        return CmdSQL;
    }

    public void setCmdSQL(String cmdSQL) {
        CmdSQL = cmdSQL;
    }

    public String criarTabela(){
        setCmdSQL("CREATE TABLE " + getTabela() + " (" +
                getId() + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                getNome() + " TEXT, " +
                getUsuario() + " TEXT, " +
                getSenha() + " TEXT" +
                ")"
        );
        return getCmdSQL();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(criarTabela());
    }

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

    }

    public void insert(String tabela, credencialModel credencial)
    {
        ContentValues dados = new ContentValues();
        dados.put(getNome(), credencial.getNome());
        dados.put(getUsuario(), credencial.getUsuario());
        dados.put(getSenha(), credencial.getSenha());
        dataBase.insert(tabela,null, dados);
    }

    public ArrayList<credencialModel> select(){
        String[] colunas = {getId(), getNome(), getUsuario(), getSenha()};
        Cursor cursor = dataBase.query(getTabela(), colunas,null, null, null, null, null,
null);
        ArrayList<credencialModel> arrayCredencialModel = new ArrayList<>();
        while(cursor.moveToNext()){
            credencialModel credencialModel = new credencialModel();
            credencialModel.setId(cursor.getInt(0));
            credencialModel.setNome(cursor.getString(1));
            credencialModel.setUsuario(cursor.getString(2));
            credencialModel.setSenha(cursor.getString(3));
            arrayCredencialModel.add(credencialModel);
        }
        return  arrayCredencialModel;
    }

    public void update(String tabela, credencialModel credencial){
        ContentValues dados = new ContentValues();
        dados.put(getNome(), credencial.getNome());
```

```
        dados.put(getUsuario(), credencial.getUsuario());
        dados.put(getSenha(), credencial.getSenha());
        dataBase.update(tabela, dados, getId() + "=" + credencial.getId(), null);
    }

    public void delete(String tabela, credencialModel credencial)
    {
        dataBase.delete(tabela,getId() + "=" + credencial.getId(),null);
    }
}
```

Verifique após o vídeo como ficou os códigos da classe **“MainActivity.java”.**

```
package com.example.appcofre;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    TextView txtServicoProg;
    EditText edtNomeProg;
    EditText edtUsuarioProg;
    EditText edtSenhaProg;
    int quantidadeRegistros;
    int registroAtual;
    int idCredencialAtual;

    credencialModel credencial = new credencialModel();

    bdModel bd;

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

        txtServicoProg = (TextView) findViewById(R.id.txtServico);
        edtNomeProg = (EditText) findViewById(R.id.edtServico);
        edtUsuarioProg = (EditText) findViewById(R.id.edtUsuario);
        edtSenhaProg = (EditText) findViewById(R.id.edtSenha);

        carregarRegistroZero();
    }

    public void clickBtnDeletar(View v)
    {
        credencial.setId(idCredencialAtual);
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
```

```
        bd = new bdModel(getApplicationContext());
        bd.delete(bd.getTabela(), credencial);
        limpar();
        carregarRegistroZero();
    }

    public void clickBtnAlterar(View v)
    {
        credencial.setId(idCredencialAtual);
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
        bd = new bdModel(getApplicationContext());
        bd.update(bd.getTabela(), credencial);
        carregarRegistroZero();
    }

    public  void clickBtnCadastrar(View v)
    {
        credencial.setNome(edtNomeProg.getText().toString());
        credencial.setUsuario(edtUsuarioProg.getText().toString());
        credencial.setSenha(edtSenhaProg.getText().toString());
        bd = new bdModel(getApplicationContext());
        bd.insert(bdModel.getTabela(), credencial);
        carregarRegistroZero();
    }

    public  void clickBtnNovo(View v)
    {
        limpar();
    }

    public void clickBtnAnterior(View v)
    {
        if(quantidadeRegistros != 0)
        {
            if(registroAtual > 0)
            {
                registroAtual = registroAtual - 1;
                carregarDados(registroAtual);
            }
        }
    }

    public void clickBtnProximo(View v)
    {
        if(quantidadeRegistros != 0)
        {
            if(registroAtual < quantidadeRegistros - 1)
            {
                registroAtual = registroAtual + 1;
                carregarDados(registroAtual);
            }
        }
    }

    public void limpar()
    {
```

```
        edtNomeProg.setText("");
        edtUsuarioProg.setText("");
        edtSenhaProg.setText("");
        txtServicoProg.setText("Serviço:");
        edtNomeProg.requestFocus();
    }

    public void carregarDados(int i) {
        bd = new bdModel(getApplicationContext());
        ArrayList<credencialModel> arrayCredencialModel;
        arrayCredencialModel = bd.select();
        quantidadeRegistros = arrayCredencialModel.size();
        if(quantidadeRegistros != 0){
            credencialModel credencialModel = arrayCredencialModel.get(i);
            txtServicoProg.setText("Serviço " + String.valueOf(credencialModel.getId()) +
":");
            idCredencialAtual = credencialModel.getId();
            edtNomeProg.setText(credencialModel.getNome());
            edtUsuarioProg.setText(credencialModel.getUsuario());
            edtSenhaProg.setText(credencialModel.getSenha());
        }
    }

    public void carregarRegistroZero(){
        registroAtual = 0;
        carregarDados(registroAtual);
    }

}
```

Envie para o seu tutor um vídeo mostrando a tela do celular com o projeto realizado por você na etapa “Você no comando”. É importante demonstrar as funcionalidades desenvolvidas até aqui.

Envie também os códigos utilizados. Aproveite para customizar o projeto **“AppCofre”** com suas cores e uma imagem escolhida por você.