APLICATIVO CADASTRO DE ALUNOS

GRUPO BOLA 7

Integrantes:

Bruno Ferreira RGM: 29523176

Julia Aparecida RGM: 30111056

Gabriel Amancio RGM: 31365973

Rafael de Souza RGM: 29320127

Rafael Ferreira Melo RGM: 30060222

Rhaian Alvarado RGM: 29162831

Rychard Alves RGM: 29690781

ConnectionFactory:

```
public class ConnectionFactory extends SQLiteOpenHelper {
    private static final String NAME = "banco.db";
    private static final int VERSION = 1;

    public ConnectionFactory(@Nullable Context context) {
        super(context, NAME, null, VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE aluno (id INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                "nome TEXT, cpf TEXT, telefone TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        String sql = "DROP TABLE IF EXISTS aluno";
        db.execSQL(sql);
        onCreate(db);
    }
}
```

Classe Javabean(Aluno):

```
package com.example.cadastroalunos.model;

import java.io.Serializable;

public class Aluno implements Serializable {
    private Integer id;
    private String nome;
    private String cpf;
    private String telefone;

    // Construtores
    public Aluno(Integer id, String nome, String cpf, String telefone)
{
        this.id = id;
        this.nome = nome;
        this.cpf = cpf;
        this.telefone = telefone;
    }

    public Aluno() {
    }

    //Formatar retorno
    @Override
    public String toString() {
        return "Nome: " + nome + "\nCPF: " + cpf + "\nTelefone: " +
telefone;
    }

    // Getters e Setters
    public Integer getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}
```

AlunoDAO:

```
public class AlunoDAO {
    private ConnectionFactory conexao;
    private SQLiteDatabase banco;

    public AlunoDAO(Context context) {
        conexao = new ConnectionFactory(context);
        banco = conexao.getWritableDatabase();
    }
    public Aluno obterAlunoPorId(long id) {
        String query = "SELECT * FROM aluno WHERE id = ?";
        Cursor cursor = banco.rawQuery(query, new
String[]{String.valueOf(id)});

        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex("id");
            int nomeIndex = cursor.getColumnIndex("nome");
            int cpfIndex = cursor.getColumnIndex("cpf");
            int telefoneIndex = cursor.getColumnIndex("telefone");

            Aluno aluno = new Aluno();
            aluno.setId(cursor.getInt(idIndex));
            aluno.setNome(cursor.getString(nomeIndex));
            aluno.setCpf(cursor.getString(cpfIndex));
            aluno.setTelefone(cursor.getString(telefoneIndex));

            cursor.close();
            return aluno;
        } else {
            cursor.close();
            return null;
        }
    }


    public long insert(Aluno aluno) {
        ContentValues values = new ContentValues();
        values.put("nome", aluno.getNome());
        values.put("cpf", aluno.getCpf());
        values.put("telefone", aluno.getTelefone());
        return banco.insert("aluno", null, values);
    }

    public void update(Aluno aluno) {
        ContentValues values = new ContentValues();
        values.put("nome", aluno.getNome());
        values.put("telefone", aluno.getTelefone());
        values.put("cpf", aluno.getCpf());

        // Atualiza o registro na tabela onde o ID corresponde ao id
do aluno
        banco.update("aluno", values, "id = ?", new
String[]{String.valueOf(aluno.getId())});
    }
```

```java
    public void deleteByCPF(String cpf) {
        banco.delete("aluno", "cpf = ?", new String[] { cpf });
    }

    public Aluno findByNome(String nome) {
        String query = "SELECT * FROM aluno WHERE nome = ?";
        Cursor cursor = banco.rawQuery(query, new String[]{nome});

        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex("id");
            int nomeIndex = cursor.getColumnIndex("nome");
            int cpfIndex = cursor.getColumnIndex("cpf");
            int telefoneIndex = cursor.getColumnIndex("telefone");

            Aluno aluno = new Aluno();
            aluno.setId(cursor.getInt(idIndex));
            aluno.setNome(cursor.getString(nomeIndex));
            aluno.setCpf(cursor.getString(cpfIndex));
            aluno.setTelefone(cursor.getString(telefoneIndex));

            cursor.close();
            return aluno;
        } else {
            cursor.close();
            return null;
        }
    }
    public Aluno findByCPF(String cpf) {
        String query = "SELECT * FROM aluno WHERE cpf = ?";
        Cursor cursor = banco.rawQuery(query, new String[]{cpf});

        if (cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex("id");
            int nomeIndex = cursor.getColumnIndex("nome");
            int cpfIndex = cursor.getColumnIndex("cpf");
            int telefoneIndex = cursor.getColumnIndex("telefone");

            Aluno aluno = new Aluno();
            aluno.setId(cursor.getInt(idIndex));
            aluno.setNome(cursor.getString(nomeIndex));
            aluno.setCpf(cursor.getString(cpfIndex));
            aluno.setTelefone(cursor.getString(telefoneIndex));

            cursor.close();
            return aluno;
        } else {
            cursor.close();
            return null;
        }
    }



    public List<Aluno> obterTodos() {
        List<Aluno> alunos = new ArrayList<>();
        Cursor cursor = banco.query("aluno", new String[]{"id",
"nome", "cpf", "telefone"},
                null, null, null, null, null);
        if (cursor != null && cursor.moveToFirst()) {
            int idIndex = cursor.getColumnIndex("id");
            int nomeIndex = cursor.getColumnIndex("nome");
            int cpfIndex = cursor.getColumnIndex("cpf");
            int telefoneIndex = cursor.getColumnIndex("telefone");

            do {
                Aluno a = new Aluno();
                a.setId(cursor.getInt(idIndex));
                a.setNome(cursor.getString(nomeIndex));
                a.setCpf(cursor.getString(cpfIndex));
                a.setTelefone(cursor.getString(telefoneIndex));
                alunos.add(a);
            } while (cursor.moveToNext());

            cursor.close();
        }

        return alunos;
    }
}
```

Main Activity:

```
public class MainActivity extends AppCompatActivity {

    private ListView listView;
    private Button buttonMenu;
    private AlunoDAO alunoDAO;
    private List<Aluno> alunos;
    private ArrayAdapter<Aluno> adapter;

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

        listView = findViewById(R.id.listView);
        buttonMenu = findViewById(R.id.buttonMenu);
        alunoDAO = new AlunoDAO(this);

        buttonMenu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MainActivity.this,
Crud.class));
            }
        });

        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
                long alunoId = alunos.get(position).getId(); // Obtém
o ID do aluno clicado
```

```
                // Cria uma Intent para a tela Crud
                Intent crudIntent = new Intent(MainActivity.this,
Crud.class);
                crudIntent.putExtra("alunoId", alunoId); // Passa o ID
do aluno clicado para a próxima tela
                startActivity(crudIntent); // Inicia a tela Crud
            }
        });
    }


    @Override
    protected void onResume() {
        super.onResume();

        alunos = alunoDAO.obterTodos();
        adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, alunos);
        listView.setAdapter(adapter);
    }
}
```

Classe Crud.java:

```
public class Crud extends AppCompatActivity {

    private EditText editTextNome;
    private EditText editTextCPF;
    private EditText editTextTelefone;
    private Button buttonGravar;
    private Button buttonAlterar;
    private Button buttonExcluir;
    private Button buttonVoltar;
    private Button buttonPdf;
    private AlunoDAO alunoDAO;
    private long alunoId; // Variável para armazenar o ID do aluno
selecionado

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

        alunoDAO = new AlunoDAO(this);

        editTextNome = findViewById(R.id.editTextNome);
        editTextCPF = findViewById(R.id.editTextCPF);
        editTextTelefone = findViewById(R.id.editTextTelefone);
        buttonGravar = findViewById(R.id.buttonGravar);
        buttonAlterar = findViewById(R.id.buttonAlterar);
        buttonExcluir = findViewById(R.id.buttonExcluir);
        buttonVoltar = findViewById(R.id.buttonVoltar);
        buttonPdf = findViewById(R.id.buttonPdf);

        // Obtem o ID do aluno selecionado da MainActivity
        alunoId = getIntent().getLongExtra("alunoId", -1);

        if (alunoId != -1) {
            // carregua os detalhes do aluno
            Aluno alunoSelecionado =
alunoDAO.obterAlunoPorId(alunoId);
            if (alunoSelecionado != null) {
                // Preenche os campos com os detalhes do aluno
selecionado
                editTextNome.setText(alunoSelecionado.getNome());
                editTextCPF.setText(alunoSelecionado.getCpf());

editTextTelefone.setText(alunoSelecionado.getTelefone());
            }
        } else {

        }




        // TextWatchers para os campos
        editTextNome.addTextChangedListener(textWatcher);
        editTextCPF.addTextChangedListener(textWatcher);
        editTextTelefone.addTextChangedListener(textWatcher);

        // OnEditorActionListener para o campo Telefone

editTextTelefone.setOnEditorActionListener(editorActionListener);


        buttonGravar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                gravarAluno();
            }
        });

        buttonAlterar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alterarAluno();
            }
        });

        buttonExcluir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                excluirAluno();
            }
        });

        buttonVoltar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { voltar();}
        });

        buttonPdf.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { gerarPdf();}
        });
```

```java
    }
    @SuppressLint("RestrictedApi")
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);

        // Única maneira de definir icones na opção do menu com action
"never"
        if(menu instanceof MenuBuilder){
            MenuBuilder m = (MenuBuilder) menu;
            m.setOptionalIconsVisible(true);
        }
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.item1) {
            gravarAluno();
            return true;
        } else if (id == R.id.item2) {
            alterarAluno();
            return true;
        } else if (id == R.id.item3) {
            excluirAluno();
            return true;
        } else if (id == R.id.item4) {
            voltar();
            return true;
        } else if (id == R.id.item5) {
        gerarPdf();
        return true;
    }

        return super.onOptionsItemSelected(item);
    }




    // TextWatcher para habilitar o botão de Alterar
    private TextWatcher textWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int
before, int count) {
        }

        @Override
        public void afterTextChanged(Editable s) {
            enableAlterarButton();
        }
    };
    // Listener para a tecla "Done" no teclado
    private TextView.OnEditorActionListener editorActionListener = new
TextView.OnEditorActionListener() {
        @Override
        public boolean onEditorAction(TextView v, int actionId,
KeyEvent event) {
            if (actionId == EditorInfo.IME_ACTION_DONE) {

                gravarAluno();
                return true;
            }
            return false;
        }
    };
    //Gerar pdf
    private void gerarPdf() {
        Intent intent = new Intent(Crud.this, PdfActivity.class);
        startActivity(intent);
    }


    // Método para formatar o CPF
    private String formatCPF(String cpf) {
        return cpf.replaceFirst("(\\d{3})(\\d{3})(\\d{3})(\\d{2})",
"$1.$2.$3-$4");
    }

    // Método para formatar o telefone (adiciona parênteses e hífen)
    private String formatPhone(String phone) {
        return phone.replaceFirst("(\\d{2})(\\d{4})(\\d{4})", "($1)
$2-$3");
    }




    // Voltar
    private void voltar() {
        finish();

    }
    // Método para habilitar/desabilitar o botão de Alterar
    private void enableAlterarButton() {
        String nome = editTextNome.getText().toString();
        String cpf = editTextCPF.getText().toString();
        String telefone = editTextTelefone.getText().toString();

        if (!nome.isEmpty() && !cpf.isEmpty() && !telefone.isEmpty())
{
            buttonAlterar.setEnabled(true);
        } else {
            buttonAlterar.setEnabled(false);
        }
    }


    private void gravarAluno() {
        String nome = editTextNome.getText().toString();
        String cpf = editTextCPF.getText().toString();
        String telefone = editTextTelefone.getText().toString();
        if (!nome.isEmpty() && !cpf.isEmpty() && !telefone.isEmpty())
{
            Aluno cpfExistente = alunoDAO.findByCPF(cpf);

            if (cpfExistente == null) {
                // Não existe um aluno com o mesmo cpf, então podemos
gravar o novo aluno
                Aluno novoAluno = new Aluno(null, nome, cpf,
telefone);
                long resultado = alunoDAO.insert(novoAluno);

                if (resultado != -1) {
                    // Gravação bem-sucedida, obtem o ID do novo aluno
                    Aluno alunoInserido = alunoDAO.findByCPF(cpf);
                    long alunoId = alunoInserido.getId();

                    Toast.makeText(this, "Aluno gravado com sucesso!",
Toast.LENGTH_SHORT).show();
                    limparCampos();

                    // Cria uma Intent para a tela MainActivity
                    Intent mainIntent = new Intent(Crud.this,
MainActivity.class);

                    // Passa o ID do novo aluno como um extra na
Intent
                    mainIntent.putExtra("alunoId", alunoId);

                    // Inicia a tela MainActivity
                    startActivity(mainIntent);
                }
            } else {
                Toast.makeText(this, "Já existe um aluno com esse
CPF.", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Preencha todos os campos.",
Toast.LENGTH_SHORT).show();
        }
    }


    private void alterarAluno() {
        String nome = editTextNome.getText().toString();
        String novoCPF = editTextCPF.getText().toString();
        String telefone = editTextTelefone.getText().toString();

        if (!nome.isEmpty() && !telefone.isEmpty() &&
!novoCPF.isEmpty() && alunoId != -1) {
            // Verifica se um aluno foi selecionado na MainActivity
            if (alunoId != -1) {
                // Cria um objeto Aluno com os novos dados
                Aluno alunoAtualizado = new Aluno((int) alunoId, nome,
novoCPF, telefone);

                // Atualiza o aluno no banco de dados
                alunoDAO.update(alunoAtualizado);

                Toast.makeText(this, "Aluno alterado com sucesso!",
Toast.LENGTH_SHORT).show();
                limparCampos();
            } else {
                Toast.makeText(this, "Aluno não encontrado. Selecione
um aluno da lista.", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Preencha todos os campos",
Toast.LENGTH_SHORT).show();
        }
    }

    private void excluirAluno() {
        String cpf = editTextCPF.getText().toString();

        if (!cpf.isEmpty()) {
            Aluno alunoParaExcluir = alunoDAO.findByCPF(cpf);

            if (alunoParaExcluir != null) {
                alunoDAO.deleteByCPF(cpf);
                Toast.makeText(this, "Aluno excluído com sucesso!",
Toast.LENGTH_SHORT).show();
                limparCampos();
            } else {
                Toast.makeText(this, "CPF não encontrado. Verifique o
CPF.", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Preencha o campo de CPF para excluir
o aluno.", Toast.LENGTH_SHORT).show();
        }
    }


    private void limparCampos() {
        editTextNome.setText("");
        editTextCPF.setText("");
        editTextTelefone.setText("");
    }
}
```

Pdf_activity(Classe para exibir pdf):

```
public class PdfActivity extends AppCompatActivity {
    private WebView webView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pdf);

        webView = findViewById(R.id.webView);
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true);

        // Carrega o link do PDF na webview
        webView.loadUrl("https://drive.google.com/file/d/1p1Ir6ksjQ-
DSA84V4nnOuRpXJtFtMstD/view?usp=sharing");
    }


}
```

Activity Main XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- Título -->
    <TextView
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#2196F3"
        android:text="@string/titulo"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:padding="16dp"
        android:gravity="center" />

    <!-- ListView para exibir os alunos -->
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

    <!-- Botão "Menu" -->
    <Button
        android:id="@+id/buttonMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/cadastrar"
        android:textSize="18sp"
        android:background="#2196F3"
        android:textColor="#FFFFFF"
        android:padding="16dp" />
</LinearLayout>
```

Crud XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="#7359A1"
        android:elevation="4dp"
        android:theme="@style/MyToolbarStyle"
        app:title="Menu" />

    <!-- Campo de texto para Nome -->
    <EditText
        android:id="@+id/editTextNome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/hint_nome" />

    <!-- Campo de texto para CPF -->
    <EditText
        android:id="@+id/editTextCPF"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/hint_cpf" />

    <!-- Campo de texto para Telefone -->
    <EditText
        android:id="@+id/editTextTelefone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/hint_telefone" />

    <!-- Botão para Gravar -->
    <Button
        android:id="@+id/buttonGravar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/menu_gravar" />

    <!-- Botão para Alterar -->
    <Button
        android:id="@+id/buttonAlterar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/menu_alterar"/>
    <!-- Botão para Excluir -->
    <Button
        android:id="@+id/buttonExcluir"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/menu_excluir" />

    <!-- Botão para Voltar -->
    <Button
        android:id="@+id/buttonVoltar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/menu_voltar" />

    <!-- Botão para Gerar PDF -->
    <Button
        android:id="@+id/buttonPdf"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_gerar_pdf" />
</LinearLayout>
```

Menu XML:

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#800080"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/item1"
        android:icon="@android:drawable/ic_menu_save"
        android:title="@string/menu_gravar"
        android:textColor="#000000"
        app:showAsAction="never" />

    <item
        android:id="@+id/item2"
        android:icon="@android:drawable/ic_menu_edit"
        android:title="@string/menu_alterar"
        android:textColor="#000000"
        app:showAsAction="never" />

    <item
        android:id="@+id/item3"
        android:icon="@android:drawable/ic_menu_delete"
        android:title="@string/menu_excluir"
        android:textColor="#000000"
        app:showAsAction="never" />

    <item
        android:id="@+id/item4"
        android:icon="?attr/actionModeCloseDrawable"
        android:title="@string/menu_voltar"
        android:textColor="#000000"
        app:showAsAction="never" />


    <item
        android:id="@+id/item5"
        android:icon="@drawable/pdf"
        android:textColor="#000000"
        android:title="@string/button_gerar_pdf"
        app:showAsAction="never" />


</menu>
```

activity_pdf(Webview):

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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=".PdfActivity">
    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />


</RelativeLayout>
```

AndroidManifest :

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/cadastro"
        android:label="Cadastro"
        android:roundIcon="@drawable/cadastro"
        android:supportsRtl="true"
        android:theme="@style/Theme.CadastroAlunos"
        tools:targetApi="31">

            <provider
                android:name="androidx.core.content.FileProvider"
                android:authorities="${applicationId}.provider"
                android:exported="false"
                android:grantUriPermissions="true">
                <meta-data
                    android:name="android.support.FILE_PROVIDER_PATHS"
                    android:resource="@xml/file_paths" />
            </provider>


        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
```

```xml
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Crud"
            android:exported="true">
        </activity>


    </application>

</manifest>
```