UFN
Universidade Franciscana
Somos universidade. Somos você.

Curso: Ciência da Computação

COMPUTAÇÃO MÓVEL

aula 01 – Introdução ao ANDROID

Professor: André Flores dos Santos.

Santa Maria – RS 2026

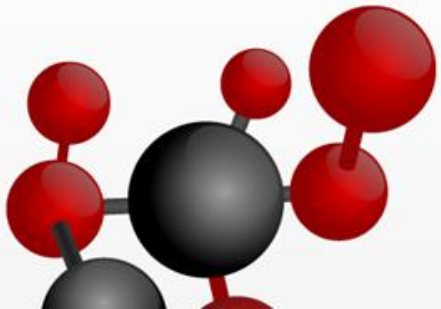
Plano de Ensino da Disciplina

APRESENTAÇÃO DOS ALUNOS

> Cada um falar o seu nome, cidade, se já tem experiência na área de computação, etc.

> Quais os planos para o seu futuro, se pretende ser programador, desenvolvedor de jogos, etc.

> E o que acha sobre programação para Aplicativos móveis, onde é aplicado e de que forma?



Apresentação sobre Android

O sistema operacional Android, uma plataforma móvel líder no mercado mundial. Vamos explorar sua história, evolução, ferramentas de desenvolvimento e recursos essenciais para construir aplicativos extraordinários.

Unidade 1 - Introdução ao Desenvolvimento Android

O Ecossistema Android

* **Origem e Governança:** Criado em 2003 e impulsionado pela aquisição do Google em 2005, o Android consolidou-se através da *Open Handset Alliance (consórcio de várias empresas)*, unindo hardware, software e operadoras em um padrão aberto.

- **Arquitetura Evolutiva:** Mais do que simples atualizações, a evolução das APIs (da 1.0 à atual) reflete a mudança de paradigma: da computação básica para a integração profunda com IA, biometria e segurança de dados.
- * **Dominância de Mercado:** Com mais de 70% do mercado global, o Android não é apenas um SO, mas a principal plataforma de entrega de serviços digitais e inovação tecnológica mobile no mundo.

História e Evolução do Android

Trajetória Tecnológica

1. **Fundação (Android Inc.):** O foco inicial era o desenvolvimento de sistemas avançados para câmeras digitais, alternando para smartphones para competir com Symbian e Windows Mobile.

2. **Maturação do SDK:** A cada versão, o Android refinou o gerenciamento de processos (LifeCycle) e a renderização de interface (usuário), passando do Dalvik para o ART (Android Runtime) para otimizar performance (código executado).

3. **Open Source e Alcance:** O licenciamento via AOSP (Android Open Source Project) permitiu que fabricantes personalizassem o sistema, garantindo a presença única da plataforma em diversos perfis de hardware.

História e Evolução do Android

Lifecycle (Ciclo de Vida)

O **Lifecycle** é o conjunto de estados por onde um componente Android (como uma Activity) passa, desde o momento em que é criado até ser destruído pelo sistema.
Diferente de um programa de PC onde você clica em "X" para fechar, no celular as coisas são voláteis: o usuário recebe uma ligação, muda de app ou a bateria acaba. O Lifecycle serve para o desenvolvedor gerenciar essas transições e não perder dados.

Os principais "Gatilhos" (Callbacks):
onCreate(): O app nasce. Aqui você infla o layout e prepara os dados.
onStart() / onResume(): O app fica visível e pronto para interação.
onPause() / onStop(): O usuário saiu do app (mas ele ainda está na memória). É hora de salvar rascunhos ou pausar vídeos.
onDestroy(): O sistema mata o app para liberar memória.

Android

- Nas versões mais recentes do SDK, a IDE recomendada para o desenvolvimento de aplicativos móveis para o sistema operacional Android é o **Android Studio**;

- O Android Studio contempla todas as ferramentas necessárias, tais como:
  - Android Studio IDE;
  - Android SDK;
  - Plataforma Android
  - Emulador para Android com Google APIs;

Instalação e Configuração do Android Studio

Baixe o Android Studio

Visite o site oficial do Android Developers e faça o download da última versão estável do Android Studio, a principal ferramenta de desenvolvimento para a plataforma Android.

Instale e Configure

Siga as instruções de instalação para o seu sistema operacional e configure o Android Studio com os SDKs, emuladores e outras ferramentas necessárias.

Atualize Regularmente

Mantenha o Android Studio atualizado para ter acesso às últimas melhorias, correções de bugs e suporte para novas versões do Android.

Android

- O computador deve ter o JDK 21 ou superior instalado para nossas aulas, pois apenas o JRE não é suficiente.

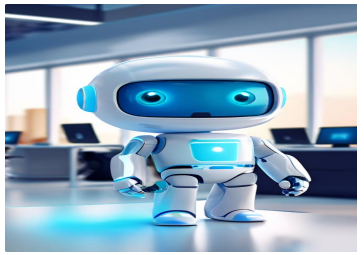
Link:https://www.oracle.com/java/technologies/downloads/?er=221886

- Download do Android Studio para Windows https://developer.android.com/studio?hl=pt-br

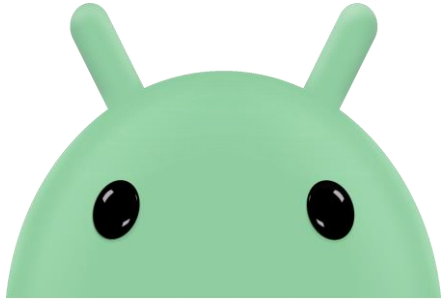
Este pacote já contém todas as ferramentas necessárias;

Comparação com outras tecnologias:

Característica	Android Nativo (Java)	Flutter	React Native
Linguagem	Java (ou Kotlin)	Dart	JavaScript / TypeScript
Criado por	Google	Google	Meta (Facebook)
Interface (UI)	Componentes Nativos (XML)	Renderização Própria (Widgets)	Componentes Nativos via "Bridge"
Performance	Máxima (acesso direto ao SO)	Alta (compilado para nativo)	Boa (depende da ponte JS)



Comparação com outras tecnologias:



Android Nativo (O que veremos agora)

•**Vantagens:** Acesso total ao hardware (sensores, câmera), melhor performance e suporte imediato a novas APIs do Google.

•**Desvantagens:** Desenvolvimento exclusivo para Android; se quiser iOS, precisa refazer o código em outra linguagem.

•**Uso:** Ideal para apps que exigem muito do hardware ou performance extrema.

Comparação com outras tecnologias:

Flutter

Vantagens: "Um código, todas as telas" (Android, iOS, Web). UI idêntica em qualquer dispositivo e desenvolvimento muito rápido com *Hot Reload*.

Desvantagens: O app final costuma ser um pouco mais pesado (em MB) e usa uma linguagem menos comum (Dart).

Uso: Apps com foco em design personalizado e rapidez de entrega.

Comparação com outras tecnologias:

React Native

Vantagens: Reuso de conhecimento de quem já sabe Web (React). Grande ecossistema de bibliotecas e facilidade de atualização.

Desvantagens: Pode sofrer lentidão em animações complexas devido à "ponte" entre o JavaScript e o código nativo.

Uso: Apps que precisam de integração constante com serviços web e atualizações frequentes.

Instalação

Link para baixar a versão utilizada no laboratório da UFN.
https://developer.android.com/studio/archive?hl=pt-br

- Aceitar os termos

Procurar por:
Android Studio Ladybug | 2024.2.1 Patch 3 December 2, 2024

Utilizar esta versão em casa, pois é a mesma do laboratório.

Android

O Android Studio vem preparado com o SDK de desenvolvimento para a versão 15 do Android ou também Android API 35;

Caso desejem trabalhar com uma versão anterior (mais leve), clique em: Tools>SDk Manager (Tela no próximo slide)

Android

SDK Manager

d Run Tools VCS Window Help

m example exemplocrud MainActivity salvar

Settings

Languages & Frameworks › Android SDK

Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
C/C++
Schemas and DTDs
Android SDK
Kotlin
Markdown
Template Data Languages
Tools
Advanced Settings
Kotlin Compiler
Experimental

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\Andre\AppData\Local\Android\Sdk Edit Optimize disk space

SDK Platforms SDK Tools SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
Android API 35	35	1	Not installed
Android VanillaIceCream Preview	VanillaIceCream	4	Not installed
Android UpsideDownCakePrivacySandbox Preview	UpsideDownCakePrivacySandbox	3	Not installed
Android API 34	34	2	Update available
Android API 34	34-ext8	1	Not installed
Android API 34	34-ext12	1	Not installed
Android TiramisuPrivacySandbox Preview	TiramisuPrivacySandbox	9	Not installed
Android 13.0 ("Tiramisu")	33	3	Partially installed
Android 13.0 ("Tiramisu")	33-ext4	1	Not installed
Android 13.0 ("Tiramisu")	33-ext5	1	Not installed
Android 12L ("Sv2")	32	1	Not installed
Android 12.0 ("S")	31	1	Not installed
Android 11.0 ("R")	30	3	Not installed
Android 10.0 ("Q")	29	5	Not installed
Android 9.0 ("Pie")	28	6	Not installed
Android 8.1 ("Oreo")	27	3	Not installed
Android 8.0 ("Oreo")	26	2	Not installed
Android 7.1.1 ("Nougat")	25	3	Not installed

Hide Obsolete Packages Show Package Details

OK Cancel Apply

Dicas:

- Para configurar as versões do SDK no projeto Android, é necessário editar o arquivo build.gradle.kts(:app). Nessa configuração, definimos o SDK que será utilizado para a compilação do aplicativo.
- No exemplo, foi utilizado o SDK 35, correspondente ao Android 15. Caso deseje utilizar versões anteriores, é fundamental verificar a compatibilidade do SDK com as bibliotecas e funcionalidades que serão usadas no projeto.
- Essa escolha é de responsabilidade do programador e depende da necessidade de suportar versões mais atuais ou antigas do Android. Além disso, é importante garantir que o emulador utilizado seja compatível com a versão do SDK configurada.

```
android {
    namespace = "com.example.passagemdados"
    compileSdk = 35  // SDK para compilação (Android 14)

    defaultConfig {
        applicationId = "com.example.passagemdados"
        minSdk = 24  // Versão mínima do Android suportada (Android 7.0)
        targetSdk = 33  // App otimizado para Android 13
        versionCode = 1
        versionName = "1.0"
    }
}
```

Dicas:

- Atenção: O targetSdk = 33 significa que o aplicativo foi testado e otimizado para rodar no Android 13. No entanto, ele poderá rodar em versões anteriores do Android, contanto que sejam compatíveis com o minSdk configurado (neste exemplo, **Android 7.0).** Isso permite que o app funcione em uma variedade de dispositivos, mas seu comportamento será garantido especialmente no Android 13.

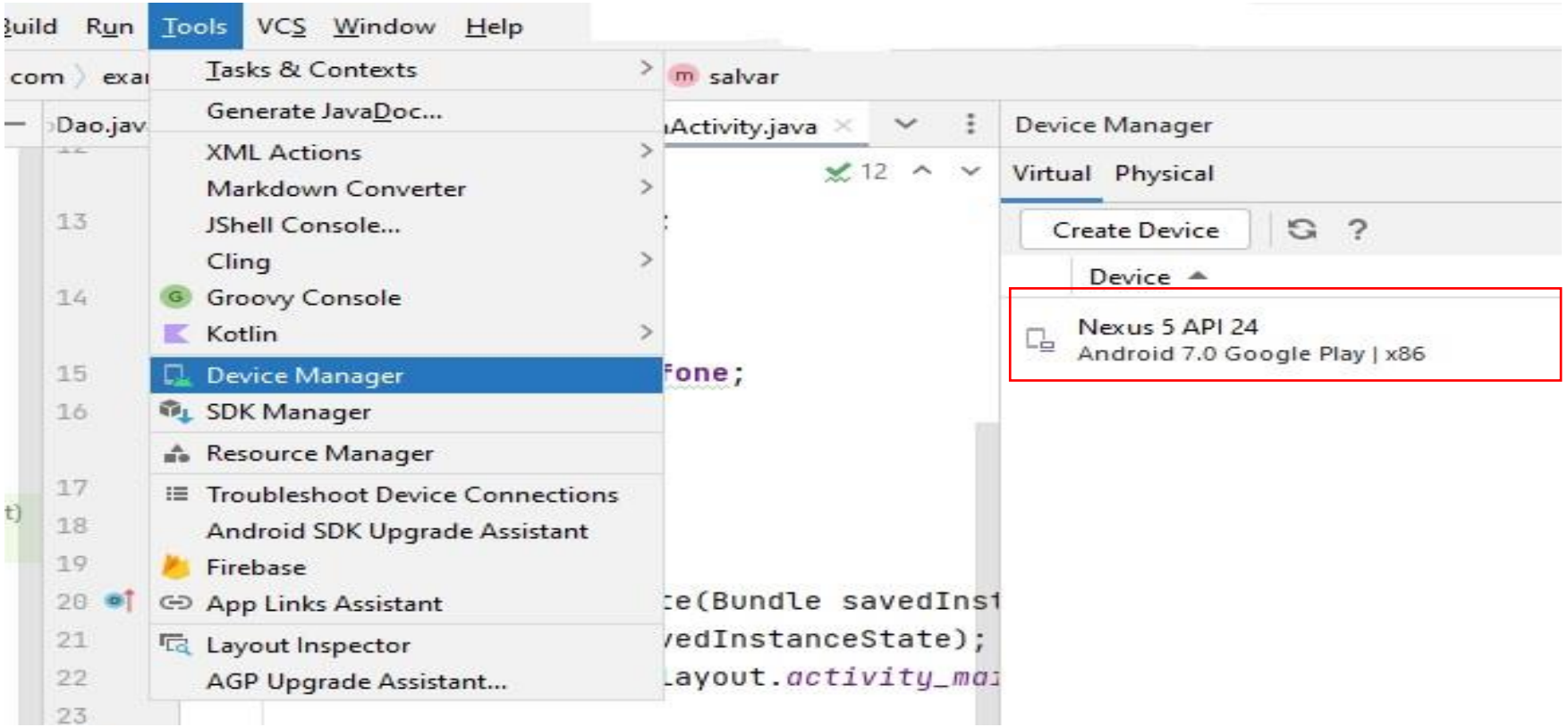
- **No AndroidManifest.xml**, o atributo tools:targetApi tem uma função semelhante ao targetSdk definido no arquivo de build do projeto, mas ele é usado principalmente para fins de ferramentas e compatibilidade durante a compilação do projeto. Neste caso, o valor '33' indica que você está destinando o comportamento do seu app para ser compatível com o Android 13 (API 33).

- O valor do targetApi no AndroidManifest.xml serve para garantir que ferramentas como o lint (que verifica erros e boas práticas no código) saibam qual versão do Android o código deve ser compatível.

- Este atributo não substitui o targetSdk definido no build.gradle.kts, que é o parâmetro principal para determinar a compatibilidade e otimização do app em versões específicas do Android. Idealmente, o tools:targetApi no Manifest deve estar alinhado com o targetSdk do projeto.

Device Manager

No device Manager escolhemos qual aparelho e qual android escolheremos para rodar no emulador e testar o App, inclusive podemos conectar o cabo-usb do nosso celular e testar o app diretamente nele (não indicado em fase de testes porque talvez seja necessário ficar instalando e desinstalando o app, ou alterando o banco de dados).



Estrutura de um Projeto Android

1 **Manifesto**

O arquivo **AndroidManifest.xml** define as principais características do seu aplicativo, como permissões, atividades e serviços.

2 **Código-fonte**

O diretório '**/src**' contém os arquivos Java ou Kotlin que implementam a lógica do seu aplicativo.

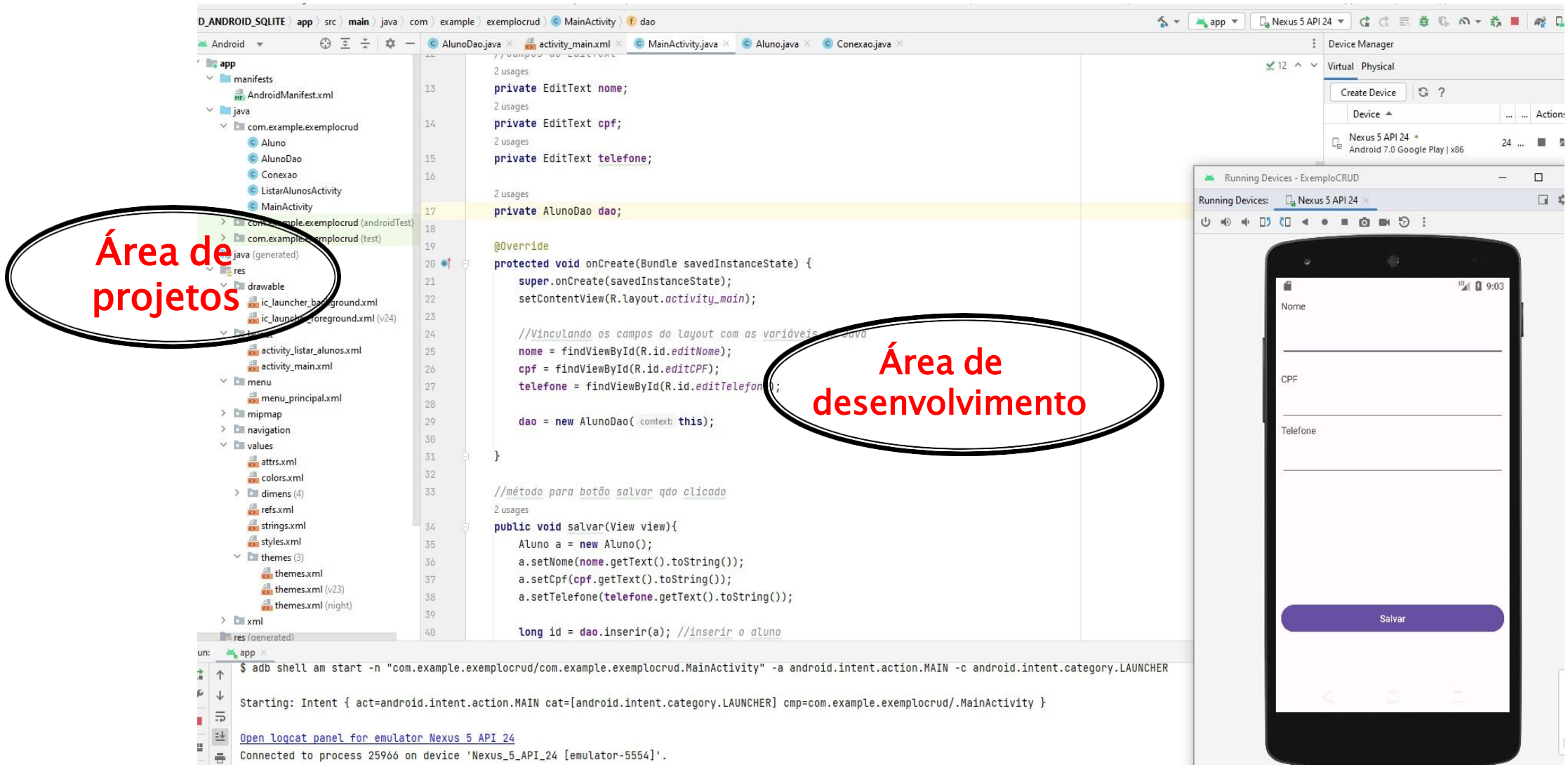
3 **Recursos**

O diretório **'/res'** armazena os recursos estáticos, como layouts, imagens, strings e arquivos de configuração.

4 **Dependências**

O arquivo **'build.gradle'** configura as dependências e bibliotecas utilizadas no seu projeto.

Android Studio



Android

- A **aplicação** é composta **por duas partes** que se integram:

Interface Gráfica
.xml

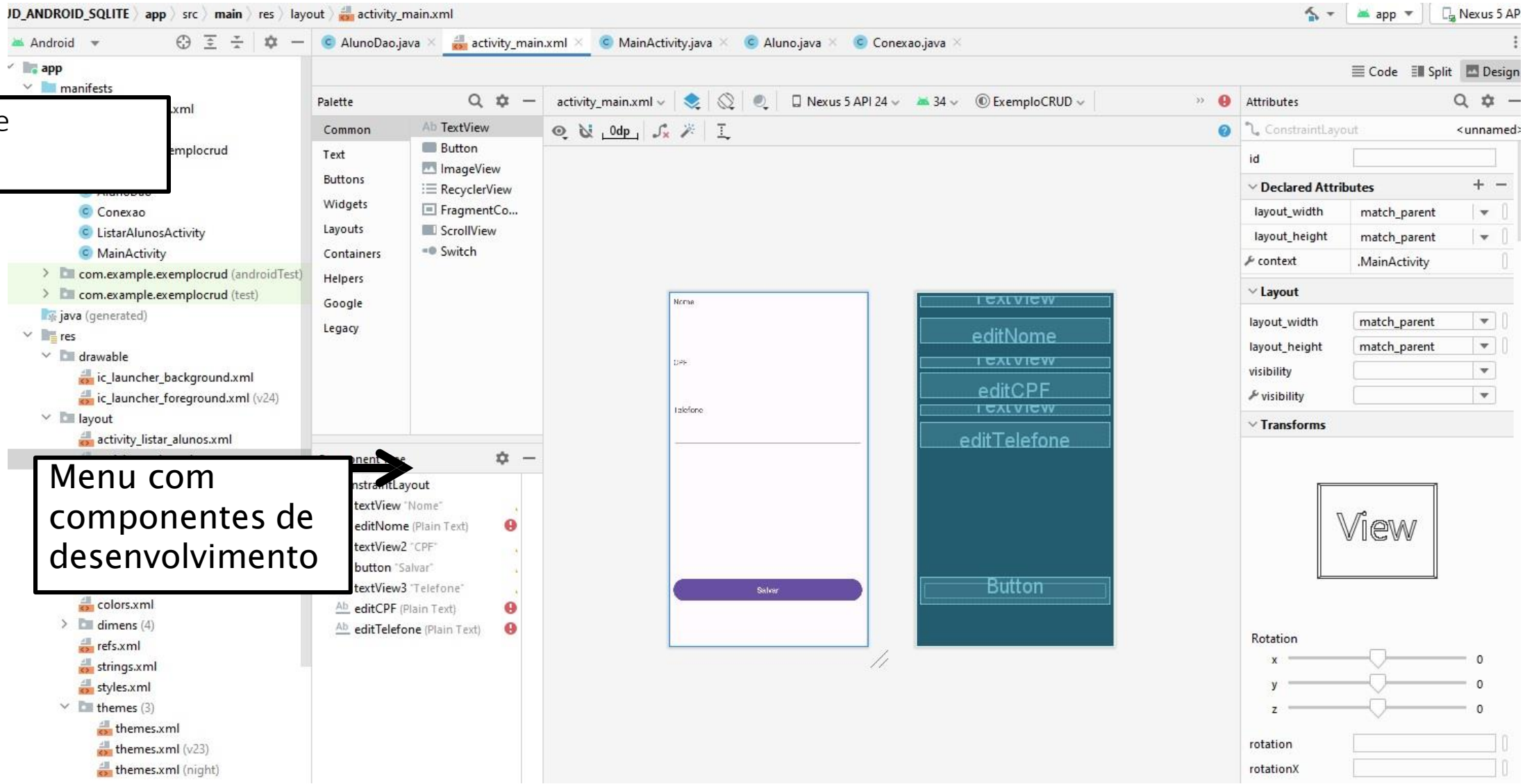
+

Código-fonte Java
.java

=

Aplicação

Interface Gráfica



Componentes Básicos da Interface Gráfica

TextView

O TextView é um componente simples para exibir texto estático na interface do usuário.

EditText

O EditText permite que o usuário insira e edite texto, como entradas de login, pesquisas e formulários.

Button

O Button é um componente interativo que dispara ações quando clicado pelo usuário, como iniciar uma nova atividade ou enviar um formulário.

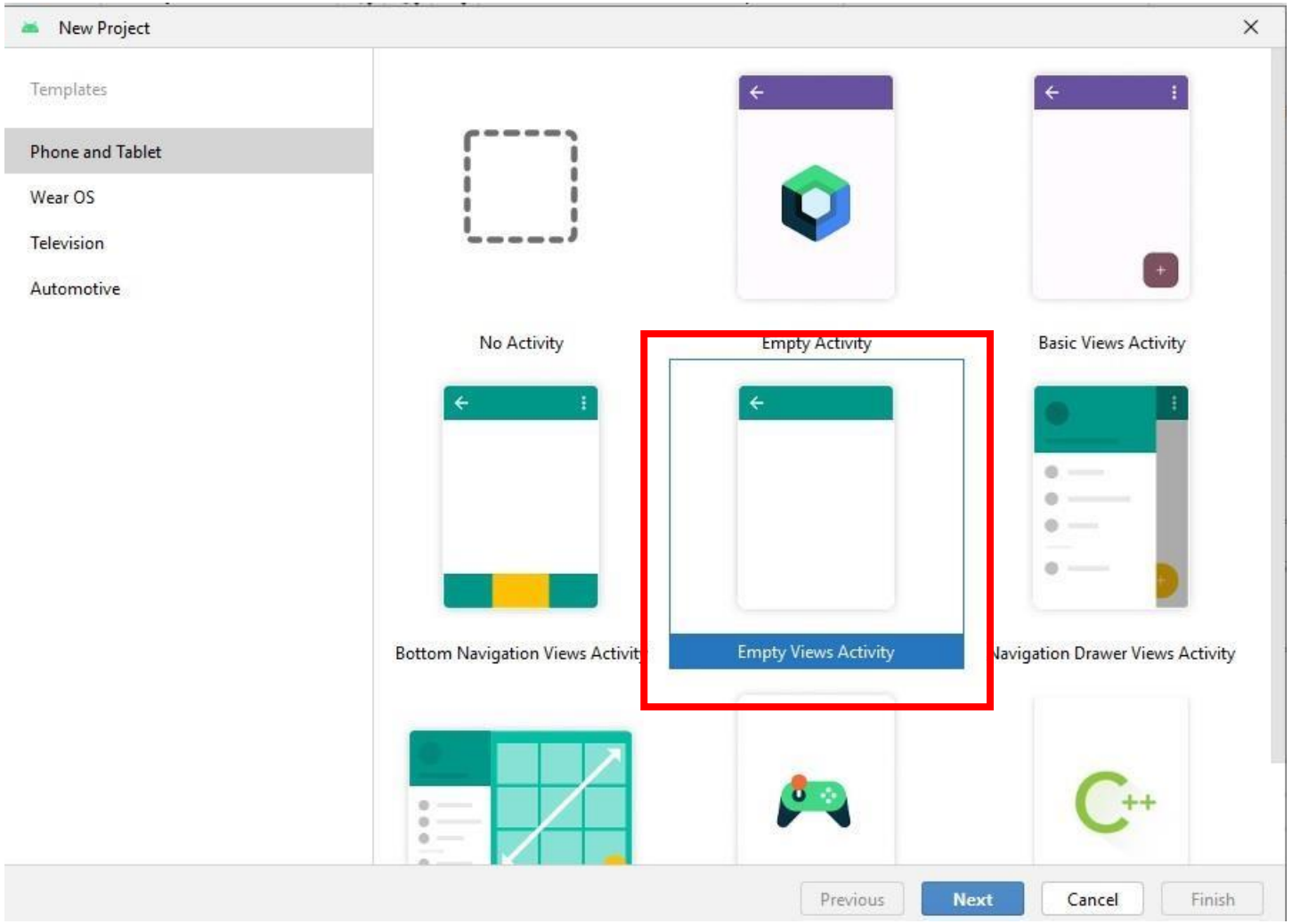
Universidade Franciscana
Somos universidade. Somos você.

CRUD ANDROID SQLITE CADASTRO ALUNO

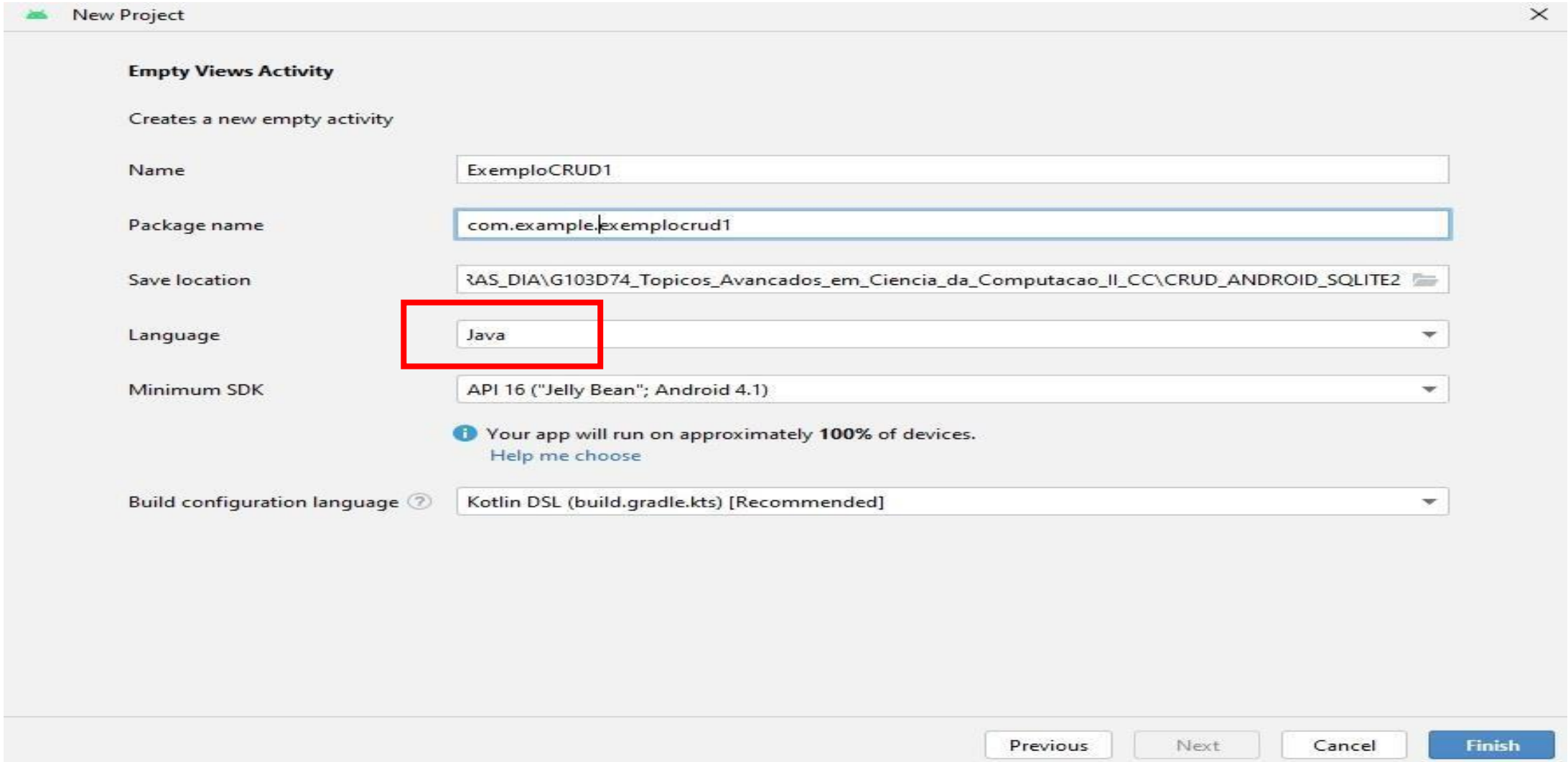
Part I

Professor: André Flores dos Santos

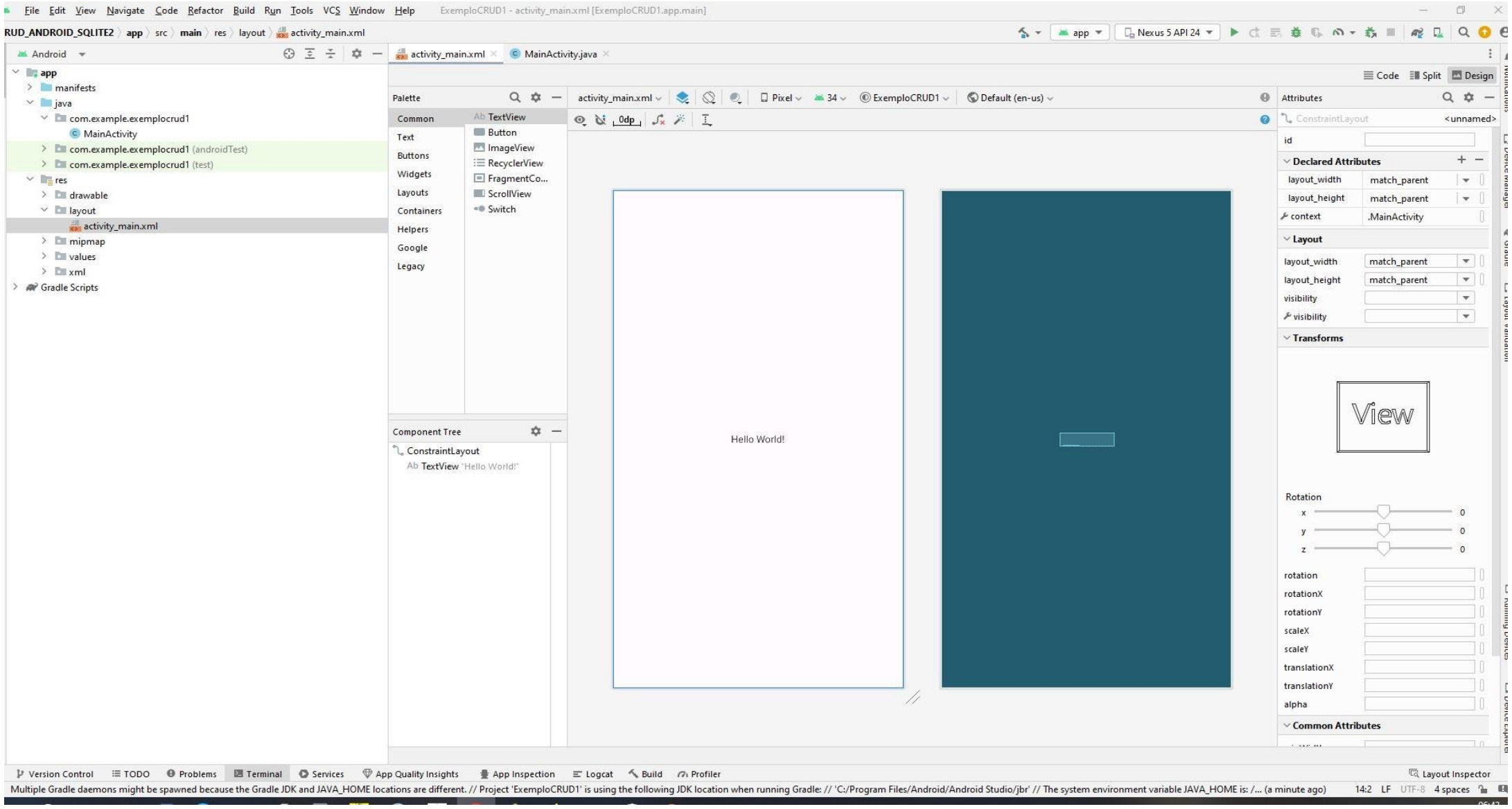
CRIAR PROJETO NOVO



CRIAR PROJETO NOVO



PROJETO CRIADO



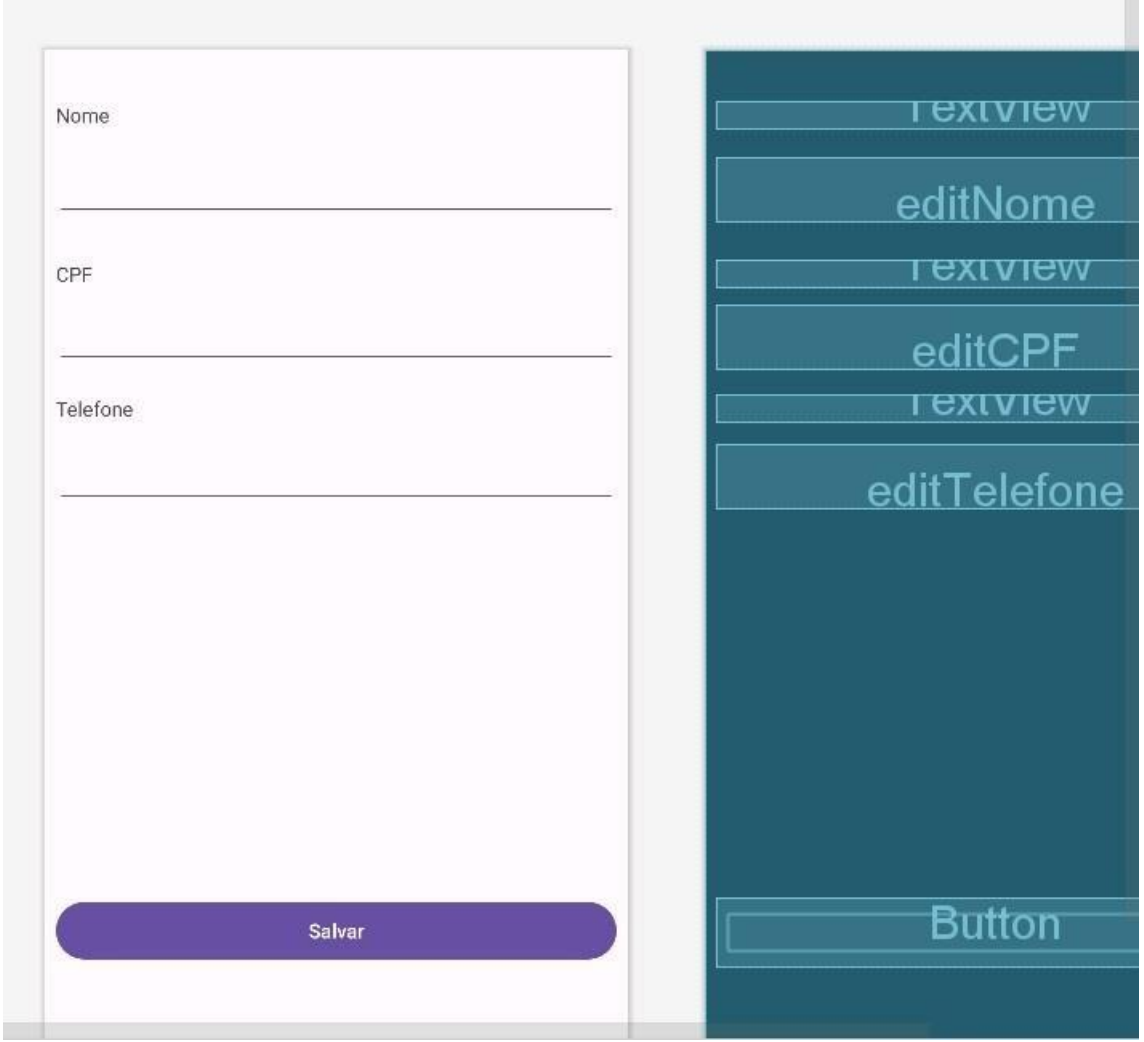
Criar Layout

Após a criação do projeto, devemos clicar na raiz do projeto app> res> layout > activity_main.xml (ou o nome que for definido para o arquivo de atividade principal.

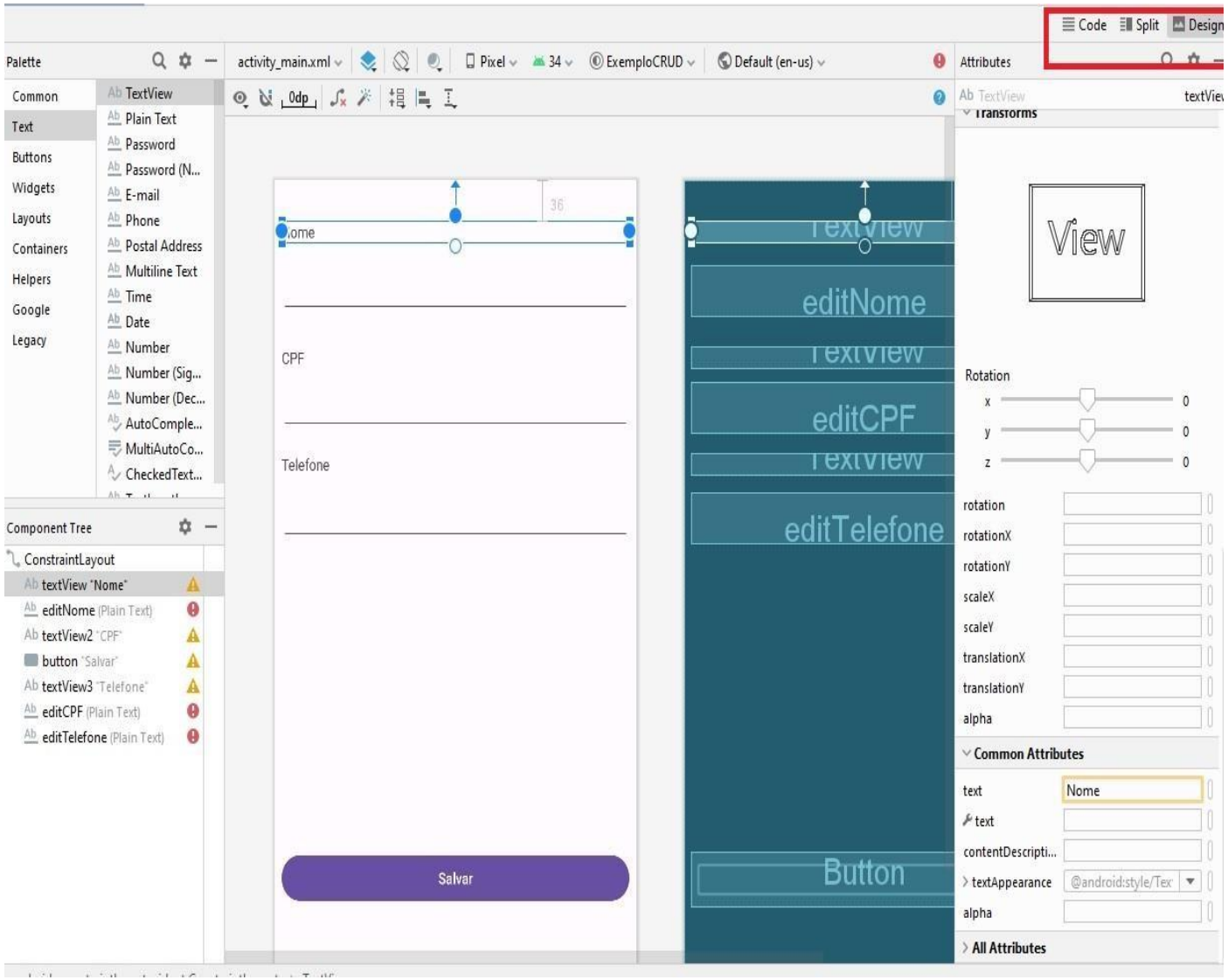
Existem 3 opções para exibir o layout:
- Code (código)
- Split
- Design (visual gráfico como temos a direita)

Essas opções geralmente ficam no canto superior direito acima do layout.

Na opção gráfica podemos arrastar diferentes tipos de objetos de texto e botões para a tela.

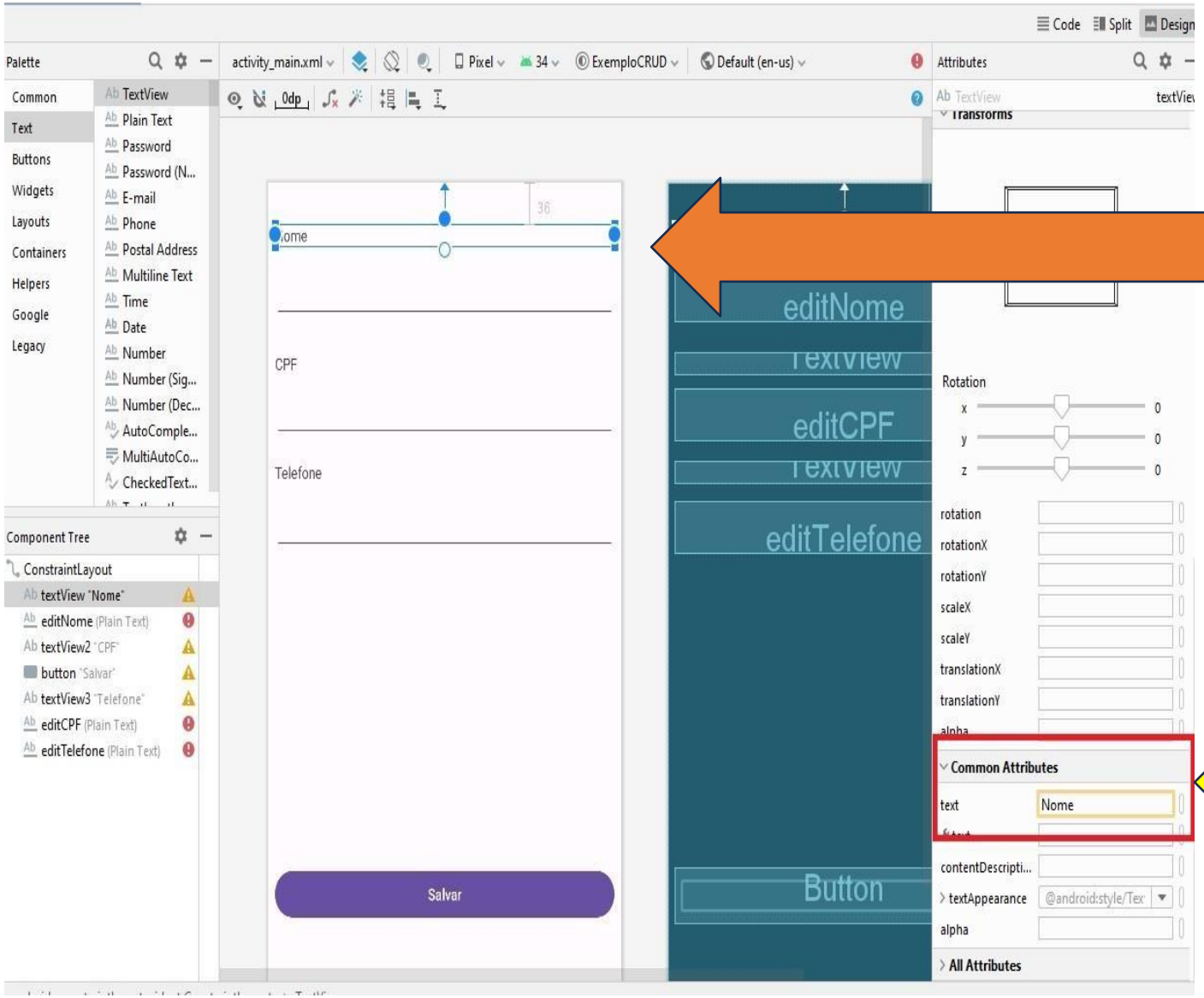


Opções de Layout



Existem 3 opções para exibir o layout:
- Code (código)
- Split
- Design (visual gráfico como temos a direita)

Criar Layout



Neste Layout usamos
- 3 Text > TextView (Nome, CPF, Telefone)
- 3 Text > Plain Text (campo de texto em branco)
- 1 Buttons > button

Essas opções ficam abertas a esquerda do layout na parte 'Design' ou visual.

Para mudar o texto dentro dos objetos, primeiro selecionamos eles e depois vamos em Attributes > Common atttributes > Text e mudamos para o valor que queremos.

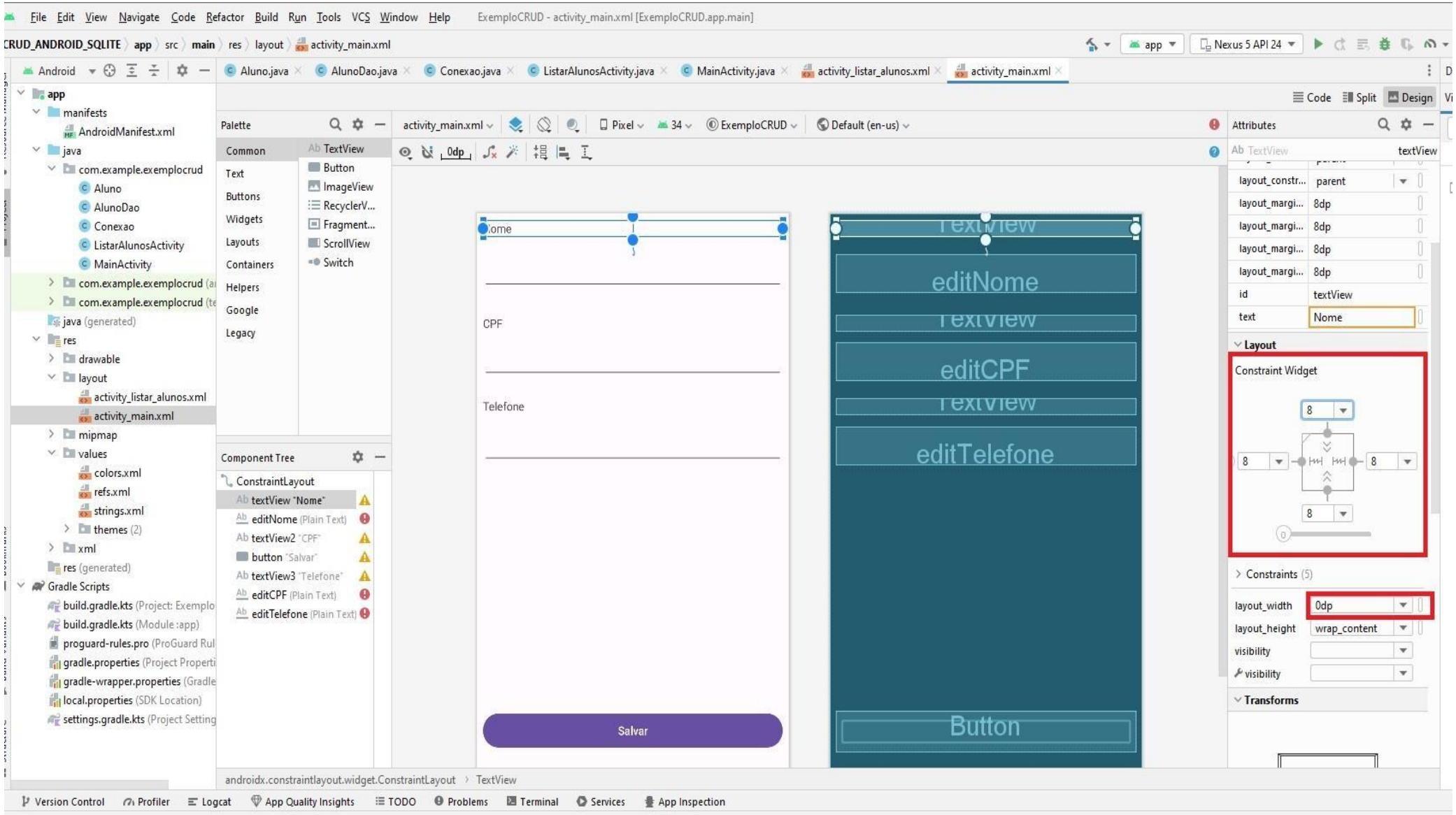
Criar Layout

Para os TextView iremos mudar o atributo ‘id’ para textView, textView2, textView3 que correspondem ao (Nome, CPF, Telefone)

Para os campos ‘Plain Text’ ou campo de texto iremos definir o ‘id’ como editNome, editCPF e editTelefone.

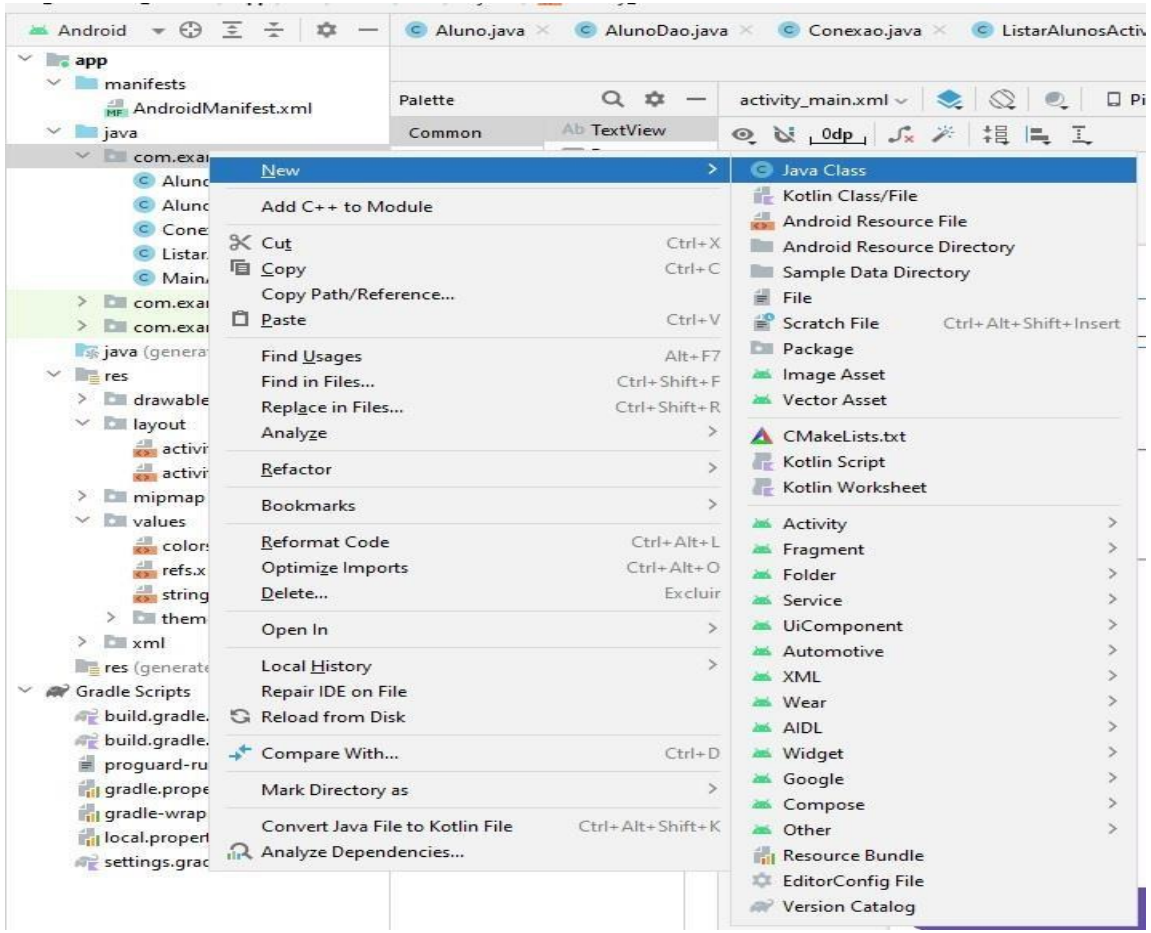
No botão iremos definir o atributo ‘text’ como ‘Salvar’.

Ancoragem dos componentes com a tela Layout



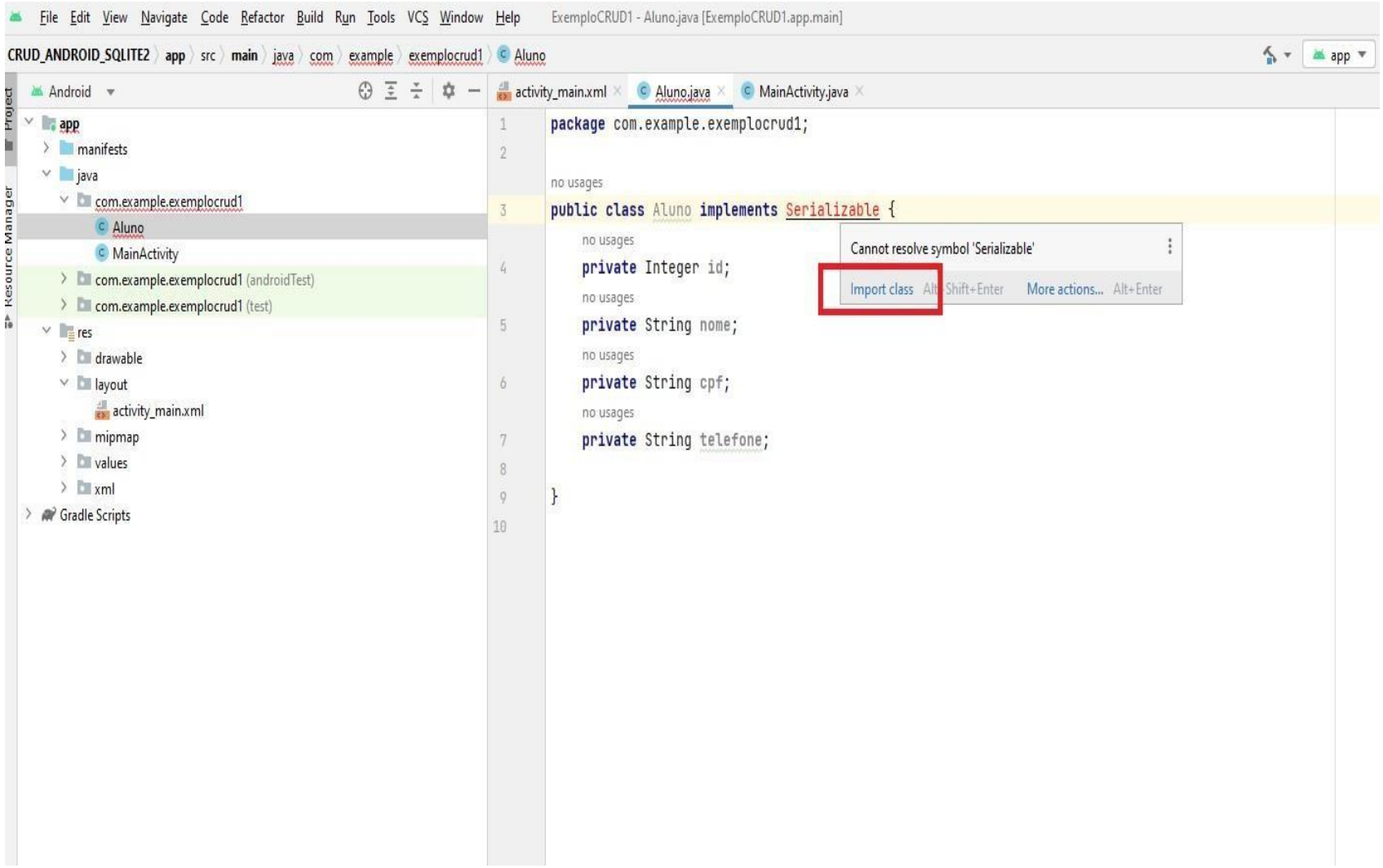
NOVA CLASSE

Clicar com o botão direito em cima do nosso projeto app>java> com.example.exemplocrud1 (ou no nome que foi definido para o pacote) Escolher New>JavaClass e dar o nome para a classe desejada



Iremos criar as seguintes classes:
- MainActivity (já criada, p cadastrar)
- Aluno .java
- Conexão .java
- AlunoDao .java

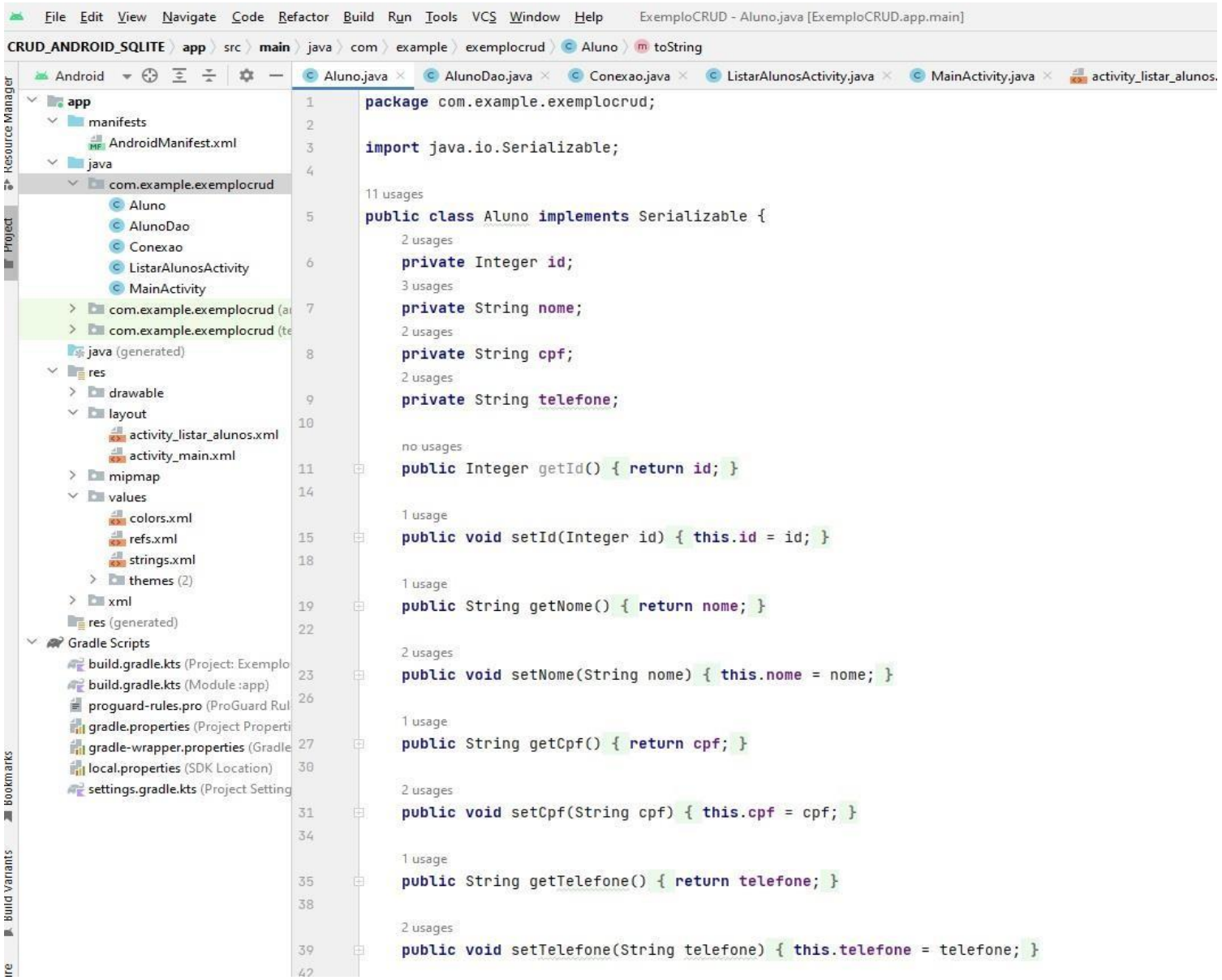
CLASSE ALUNO



O aluno terá os seguintes campo:
Id, nome, cpf, telefone.

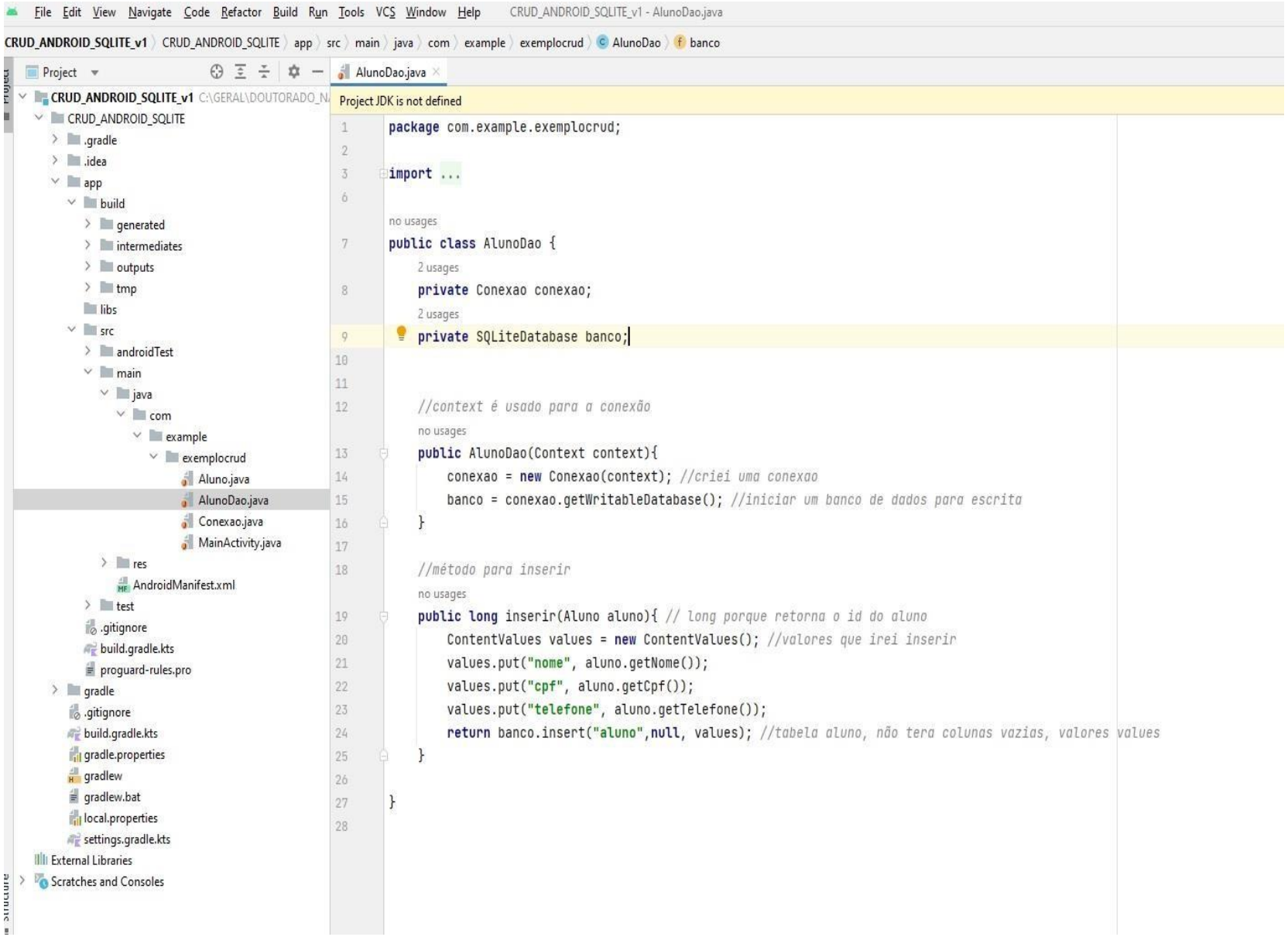
Se houver problemas com bibliotecas o texto irá ficar vermelho. Para resolver devemos parar o mouse em cima e irá abrir uma opção ‘import class’. Após clicar nessa opção o problema é resolvido e a biblioteca é importada.

CLASSE ALUNO



Mandar gerar getter and setter automaticamente, clicando com o botão direito dentro do código Opção generate>getter and setter, e marcar todos os campos. Deverá ficar como a tela ao lado.

CLASSE ALUNO DAO



CLASSE CONEXAO

```
package com.example.exemplocrud;

import ...

no usages
public class Conexao extends SQLiteOpenHelper {

    1 usage
    private static final String name = "banco.db";
    1 usage
    private static final int version = 1;

    no usages
    public Conexao(Context context) { super(context, name, null, version); }


    no usages
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table aluno(id integer primary key autoincrement, " +
                "nome varchar(50), cpf varchar(50), telefone varchar(50))");
    }

    no usages
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

CLASSE MainActivity

```
public class MainActivity extends AppCompatActivity {

    //campos do EditText
    2 usages
    private EditText nome;
    2 usages
    private EditText cpf;
    2 usages
    private EditText telefone;

    2 usages
    private AlunoDao dao;

    no usages
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Vinculando os campos do layout com as variáveis do Java
        nome = findViewById(R.id.editNome);
        cpf = findViewById(R.id.editCPF);
        telefone = findViewById(R.id.editTelefone);

        dao = new AlunoDao(this);

    }

    //método para botão salvar qdo clicado
    public void salvar(View view){
        Aluno a = new Aluno();
        a.setNome(nome.getText().toString());
        a.setCpf(cpf.getText().toString());
        a.setTelefone(telefone.getText().toString());
        long id = dao.inserir(a); //inserir o aluno
        Toast.makeText(this,"Aluno inserido com id: " +id, Toast.LENGTH_SHORT).show();
    }

}
```

Activity_main.xml

AlunoDao.java | Conexao.java | MainActivity.java | activity_main.xml

```
46          android:layout_marginTop="8dp"
47          android:layout_marginEnd="8dp"
48          android:layout_marginRight="8dp"
49          android:layout_marginBottom="8dp"
50          android:text="CPF"
51          app:layout_constraintBottom_toBottomOf="parent"
52          app:layout_constraintEnd_toEndOf="parent"
53          app:layout_constraintHorizontal_bias="0.0"
54          app:layout_constraintStart_toStartOf="parent"
55          app:layout_constraintTop_toBottomOf="@+id/editNome"
56          app:layout_constraintVertical_bias="0.034" />
57
58      <Button
59          android:id="@+id/button"
60          android:layout_width="0dp"
61          android:layout_height="wrap_content"
62          android:layout_marginStart="8dp"
63          android:layout_marginLeft="8dp"
64          android:layout_marginTop="8dp"
65          android:layout_marginEnd="8dp"
66          android:layout_marginRight="8dp"
67          android:layout_marginBottom="8dp"
68          android:text="Salvar"
69
70          android:onClick="salvar"
71          app:layout_constraintBottom_toBottomOf="parent"
72          app:layout_constraintEnd_toEndOf="parent"
73          app:layout_constraintHorizontal_bias="0.0"
74          app:layout_constraintStart_toStartOf="parent"
75          app:layout_constraintTop_toBottomOf="@+id/textView2"
76          app:layout_constraintVertical_bias="0.842" />
77
78      <TextView
79          android:id="@+id/textView3"
80          android:layout_width="0dp"
81          android:layout_height="wrap_content"
82          android:layout_marginStart="8dp"
83          android:layout_marginLeft="8dp"
84          android:layout_marginTop="8dp"
```

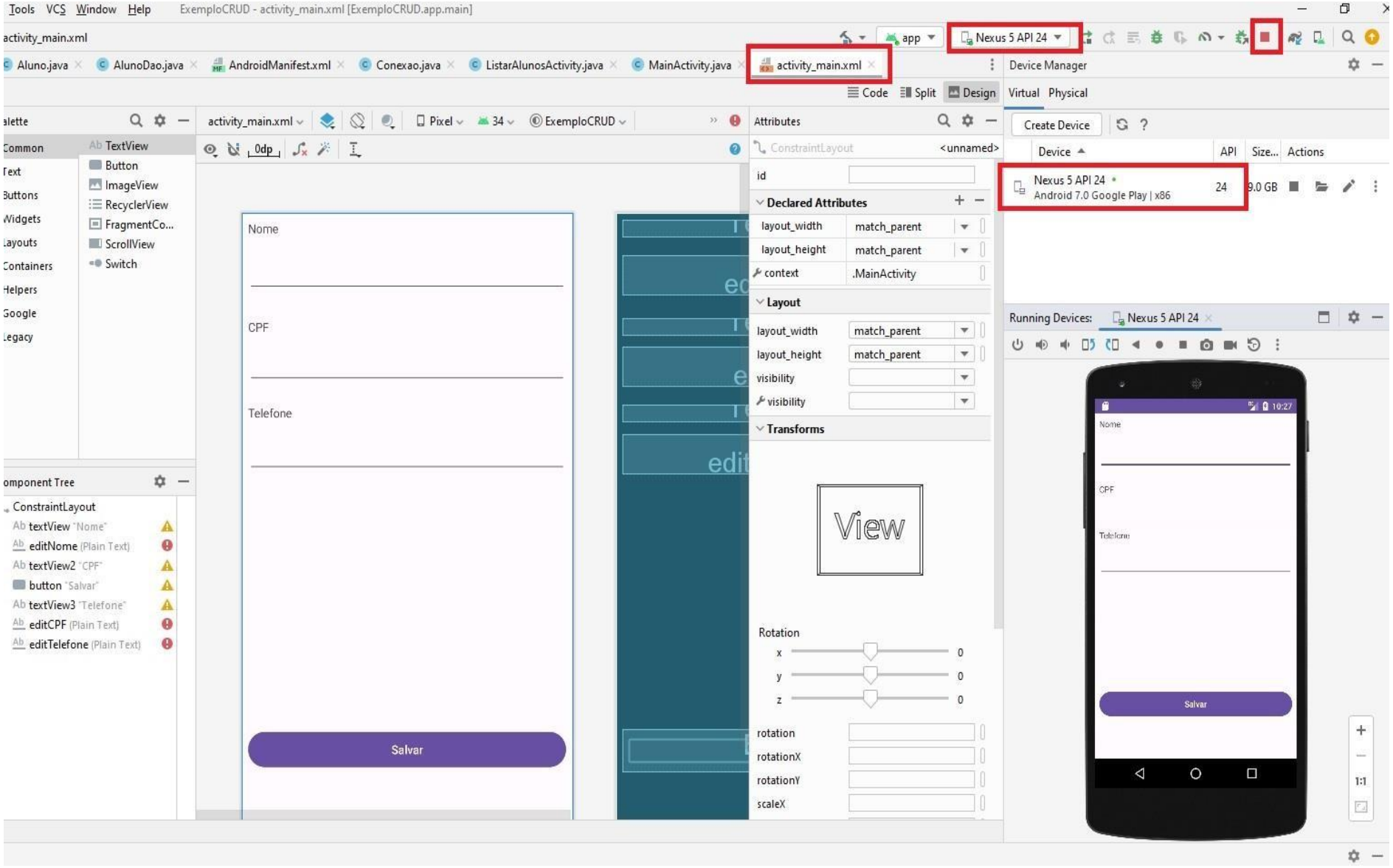
Devemos configurar o botão ‘salvar’ para o evento ‘onClick’, então ele irá executar o método ‘salvar’ da ‘MainActiviy’ quando for clicado.
Existem duas formas de fazer isso.

1) Editamos esse campo dentro do arquivo res > activity_main.xml

Ou

2) selecionar o botão com o layout aberto e ir no atributo onClick e digitar ‘salvar’ na caixa de texto. Isso será o nome do método. Logo após verificar no código se foi inserido, conforme a imagem.

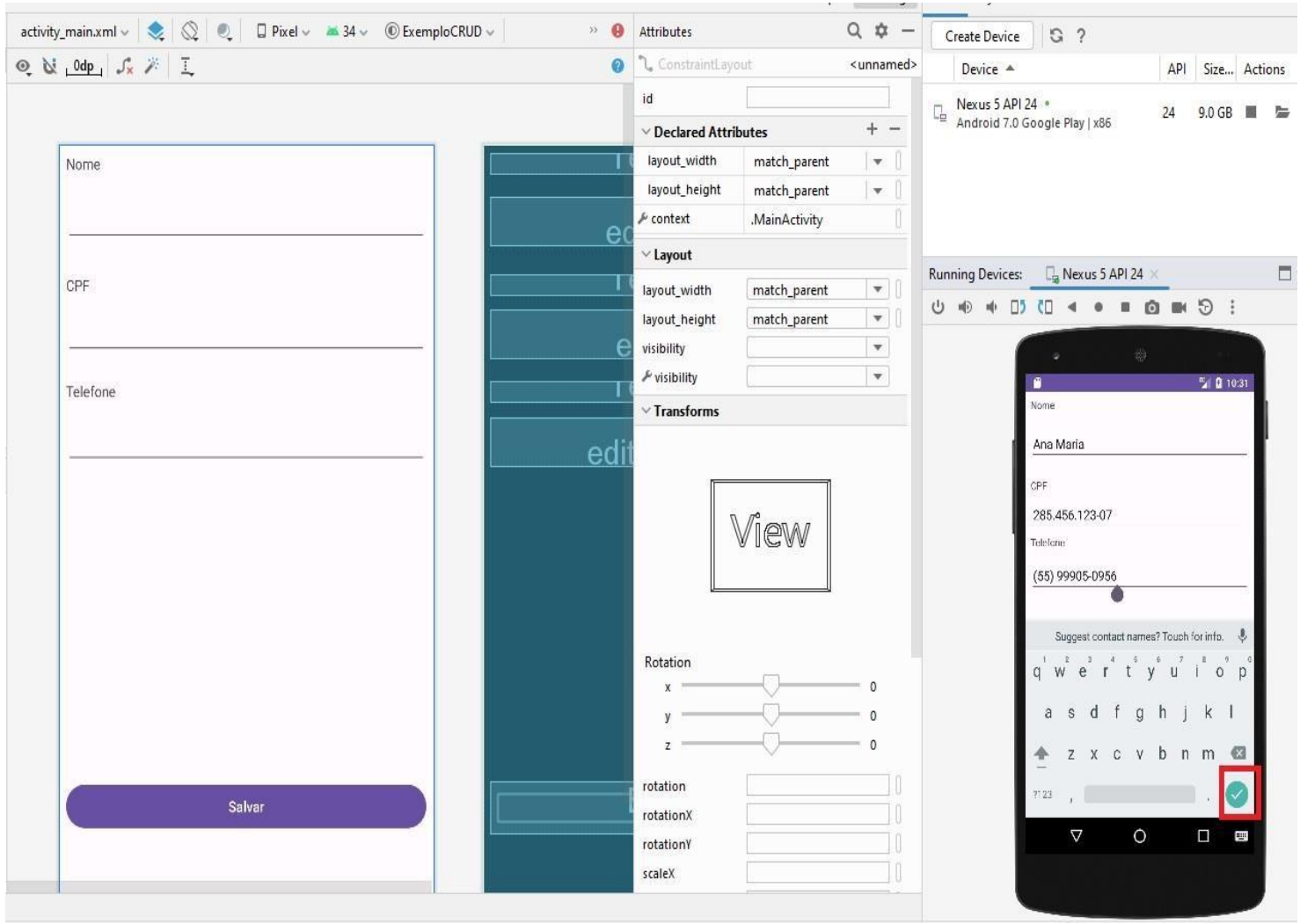
Testar o app no emulador



Clicar no ícone de ‘Play’ para rodar o aplicativo.
Se tudo der certo, a imagem do app deverá aparecer para nós cadastrarmos um aluno no aplicativo.

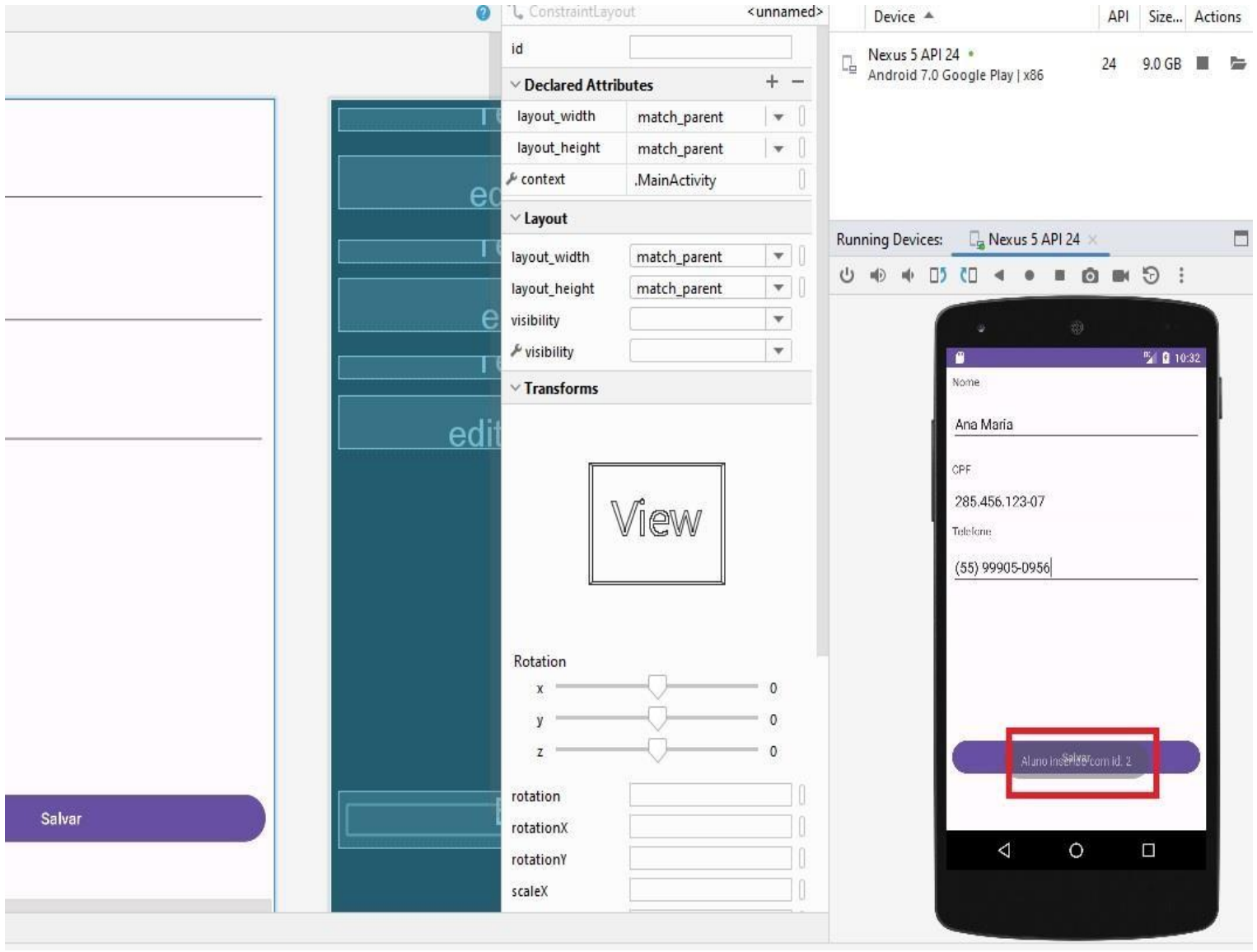
Devemos preencher os campos (ainda sem validação) e clicar no botão ‘salvar’.

Testar o app no emulador



Após preencher os campos, clicar no 'enter', que nos levará ao 'salvar'.

Testar o app no emulador



Se tudo der certo, irá aparecer uma mensagem ‘Aluno inserido com sucesso id *.

Arquivos bases para consulta: https://github.com/andreflores2009/AplicativosMoveis_26/tree/main/Aula01

Nome do pacote no projeto:
com.example.exemplocrud;

Se o nome no seu projeto for diferente usar o comando no início dos arquivos:
package com.nomedopacote

Recursos e Ferramentas Adicionais que podemos explorar

Diretrizes para Aprendizado Contínuo

- **Documentação Oficial:** A "bíblia" do desenvolvedor é o portal *Android Developers (https://developer.android.com/?hl=pt-br)*, onde as referências de API e guias de design (Material Design) são atualizados em tempo real.

- * **Prática em Hardware:** Embora os emuladores sejam eficientes, o teste em dispositivos físicos é indispensável para validar o consumo de bateria, latência de sensores e usabilidade sob luz solar ou toque real.

* **Comunidade e Bibliotecas:** O domínio de bibliotecas de terceiros e repositórios (como o Maven Central) é essencial para acelerar o desenvolvimento e seguir as melhores práticas da indústria.

Tarefas

- Navegar nas opções de interface
- Inserir objetos (elementos gráficos)
- Compilar o projeto
- Emular aplicações com os objetos inseridos
- **Objetivo**: *familiarizar-se com o ambiente*

Parte II – O que aprendemos na aula 01?

Arquitetura do Projeto CRUD

Explicação macro de como as classes se comunicam.

•**Modelo (Aluno.java):** Nossa "forma de bolo". Define quais dados do aluno vamos guardar (Nome, CPF, Telefone).

•**Persistência (Conexao.java & AlunoDao.java):** O "mecanismo de memória". A Conexão abre a porta do banco SQLite, e o AlunoDao executa as tarefas (Salvar, Deletar, Listar).

•**Interface (MainActivity.java):** O "mestre de cerimônias". É quem ouve o clique do usuário e manda os dados para o DAO salvar.

Parte II – O que aprendemos na aula 01?

Classe Aluno (A Entidade)
Foco em: Encapsulamento e Objeto de Valor.

Atributos Privados: Protegemos os dados (id, nome, cpf, telefone) para que não sejam alterados sem controle.

Getters e Setters: As "portas de acesso". Usamos para ler e gravar informações no objeto.

Serializable: Marcamos a classe para que o Android consiga "empacotar" esse objeto e enviá-lo entre telas, se necessário.

Parte II - O que aprendemos na aula 01?

Persistência com SQLite (Conexao e DAO)
Foco em: Como os dados não somem ao fechar o app.

SQLiteOpenHelper (Conexao): Classe nativa do Android que gerencia a criação do banco de dados e a versão das tabelas.

O "Contrato" (AlunoDao):

SQLiteDatabase: Objeto que usamos para escrever (getWritableDatabase()) ou ler (getReadableDatabase()) no disco.

ContentValues: Um "pacotinho" de dados (chave e valor) usado para inserir linhas na tabela sem precisar escrever SQL complexo.

Parte II - O que aprendemos na aula 01?

O Coração da Interface (MainActivity)
Foco em: Ciclo de vida e Vínculo com XML.

setContentView(R.layout.activity_main): Comando que diz ao Java qual "desenho" (XML) ele deve carregar na tela.

findViewById(R.id.id_do_componente): A ponte de ligação. Conecta o componente visual do XML (EditText, Button) a uma variável no Java para podermos manipular via código.

setOnClickListener: O "ouvido" do botão. Fica esperando o toque do usuário para disparar a ação de salvamento.

Parte II – O que aprendemos na aula 01?

Resumo de Comandos "Mágicos"

Comando	O que faz na prática?
findViewById	Vincula o design (XML) ao comportamento (Java).
getText().toString()	Pega o que o aluno digitou na caixinha de texto.
dao.inserir(aluno)	Pega o objeto preenchido e grava no banco de dados.
Toast.makeText	Mostra aquela mensagem rápida na parte inferior da tela.

O XML é como o interruptor na parede, e o `findViewById` é o fio que liga esse interruptor à lâmpada (o nosso código Java)

Parte II – O que aprendemos na aula 01?

O "this" (A Identidade da Tela)
Foco: Por que usamos this na MainActivity?

•**Definição:** O this é uma palavra-chave que referencia a própria classe onde você está escrevendo o código.

•**No nosso projeto:** Quando fazemos new AlunoDao(this), estamos dizendo ao banco de dados: "Eu, a MainActivity, sou a dona desta operação".

•**O "RG" da Activity:** O Android exige saber qual tela está ativa para gerenciar a memória. O this é o documento de identidade que você apresenta ao sistema.

•**Analogia:** É como se você estivesse assinando um contrato; o this é a sua assinatura no final da página.

Parte II – O que aprendemos na aula 01?

O "context" (O Cordão Umbilical)
Foco: Como as classes Conexao e AlunoDao acessam o sistema.

O que é o Context: É uma "entidade" do Android que permite acessar recursos globais (banco de dados, strings, temas).

Por que o DAO precisa dele? Classes como Conexao e AlunoDao não são "telas", por isso elas não têm acesso direto ao disco do celular. Elas precisam do context que foi enviado pela Activity para "pedir permissão" ao Android.

No Código:

Na Activity: Enviamos o this.

No DAO/Conexao: Recebemos como Context context no construtor.

Analogia: O context é como o oxigênio ou a rede elétrica da sala; sem estar conectado a ele, nada funciona.

Parte II - O que aprendemos na aula 01?

A "view" (O Gatilho da Ação)
Foco: O parâmetro nos métodos de clique.

Definição: A view representa o componente visual (Botão, Campo de Texto, Imagem) que o usuário acabou de tocar.

Por que aparece no método salvar(View view)?

O Android "avisa" ao método quem foi o responsável pelo clique.
Mesmo que você não use a variável view dentro do código, ela precisa estar lá para o Android entender que aquele método é um "ouvinte de cliques" (onClick).

Poder Oculto: Se você tiver 3 botões usando o mesmo método, você usa a view para descobrir qual deles foi apertado através do view.getId().

Analogia: É como o "remetente" de uma carta. O método é a ação de ler, e a view é o envelope dizendo quem enviou.

Parte II - O que aprendemos na aula 01?

O que é o ContentValues?
Foco: O "Dicionário" de dados.

Definição: É uma classe do Android usada para armazenar um conjunto de dados em pares de **Chave e Valor**.

A Chave: É o nome da coluna da tabela no banco (ex: "nome", "cpf").

O Valor: É a informação real do aluno que queremos salvar (ex: "André", "123.456...").

No Código: No AlunoDao, usamos o método .put("coluna", valor) para carregar esse "pacotinho" antes de enviar para o banco.

Analogia: Imagine uma ficha de cadastro de papel. A "Chave" é o rótulo impresso no campo (Nome: _____) e o "Valor" é o que você escreve à caneta.

Parte II – O que aprendemos na aula 01?

O Fluxo do Salvamento (Passo a Passo)
Foco: O caminho do dado do XML até o Disco.

Captura: O MainActivity pega o texto dos EditText usando getText().toString().

Objeto: Esses textos são guardados dentro do objeto Aluno via métodos *Setters*.

Empacotamento: O AlunoDao recebe esse objeto e transfere os dados para o **ContentValues**.

Gravação: O método db.insert("aluno", null, values) finalmente entrega o pacote ao SQLite para ser gravado permanentemente.

Parte II – O que aprendemos na aula 01?

Como verificar o que esta salvo no banco de dados sem implementar o 'ListarAluno?

Software 'SQLite Studio: https://sqlitestudio.pl/'

Baixar o arquivo do banco de dados no AndroidStudio e carregar no SQLite Studio.

Caminho:

Caminho pelo Menu Superior:

Clique em **View**.

Vá em **Tool Windows**.

Escolha **Device Explorer (com o celular iniciado no emulador)**.

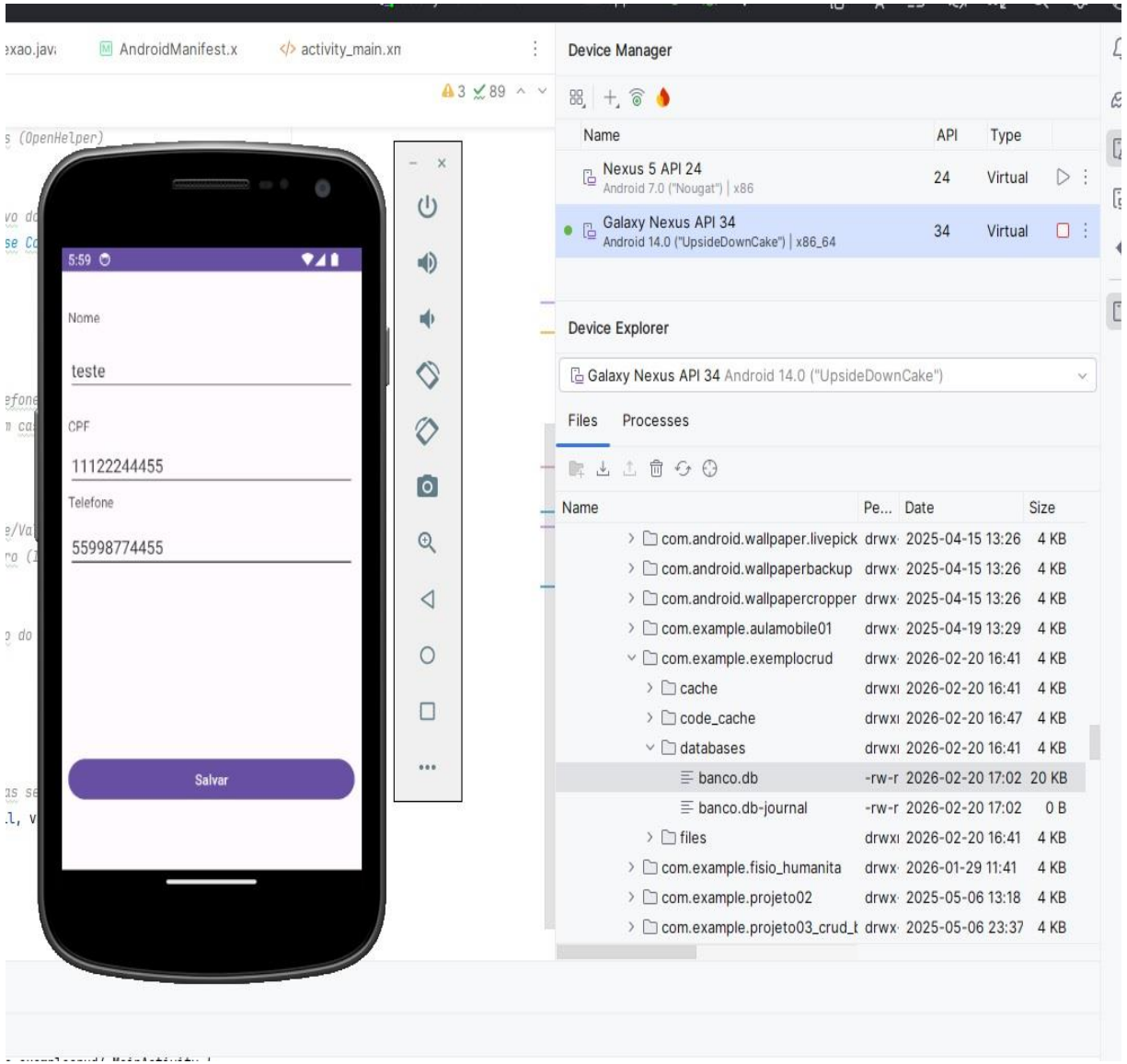
Parte II – Baixar o arquivo do banco de dados

Procurar nas pastas pelo caminho:

•**Selecione o seu Emulador:** No topo da janelinha, verifique se o emulador correto está selecionado na lista suspensa.

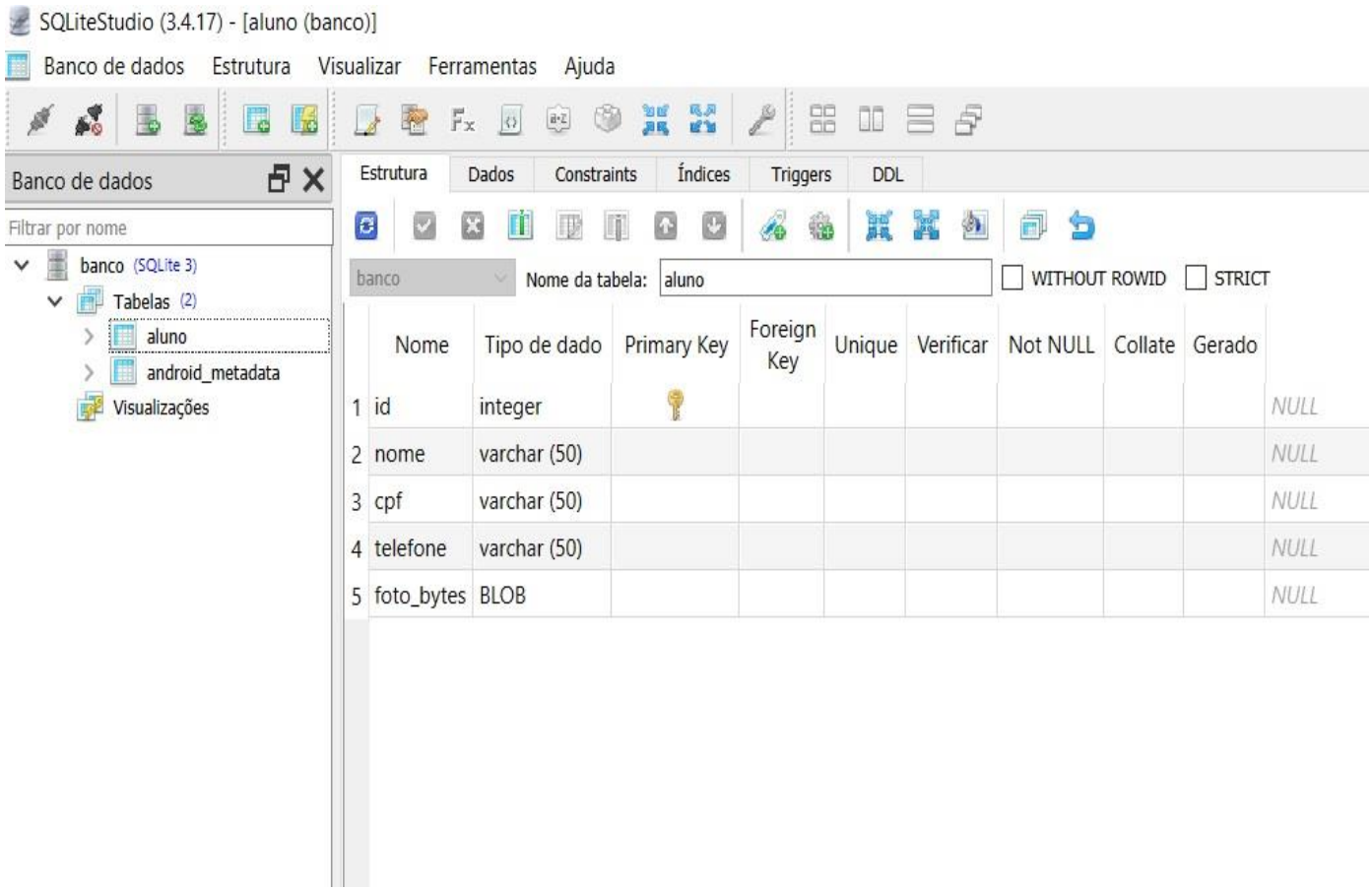
•**Navegue nas Pastas:**
•Abra a pasta **data**.
•Dentro dela, abra **outra pasta data**.
•Procure pelo nome do seu pacote (geralmente **com.example.exemplocrud**).

•Abra a pasta **databases**.
•**Extração:** Clique com o botão direito no arquivo **banco.db** e selecione **Save As...** para salvá-lo em seu computador e abrir no SQLite Studio.



Parte II – Baixar o arquivo do banco de dados

Clique na aba estrutura para ver a estrutura do banco de dados e na aba dados para ver o que foi cadastrado.



Exercício para entregar: Exercício 01

Desenvolva tudo que foi proposto em aula e entregue os códigos das suas classes.java (zip)

Desenvolver o que foi proposto em aula e entregar:

- MainActivity
- Aluno .java
- Conexão .java
- AlunoDao .java

Caminho que ficam salvas no projeto:
app>src>main>java>com>nomepacote>

Referências:

LECHETA, Ricardo R. Google android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. São Paulo, SP: Novatec, 2009. 576 p.

YAVAGAL, Roopa R.; TALUKDER, Asoke K. Mobile computing: technology, applications, and service creation. New York, NY: McGraw-Hill, 2007. 668 p.

Organizador Diego Silva. Desenvolvimento para dispositivos móveis, 2017. (Biblioteca Digital)

ANDROID. Android Developers. Disponível em: https://developer.android.com

MIKKONEN, Tommi. Programming mobile devices: an introduction for practitioners. Chichester, England: John Wiley & Sons, c2007. 222 p.

ROGERS, Rick. Desenvolvimento de aplicações Android. São Paulo, SP: Novatec, 2009. 376 p.

Organizador Everaldo Leme da Silva; Organizador Rafael Félix. Arquitetura para computação móvel – 2ª edição, 2019. (Biblioteca Digital)

LECHETA, Ricardo R. Google Android: aprenda a criar aplicações para dispositivos móveis com a Android SDK. 2. ed. rev. ampl. São Paulo, SP: Novatec, 2010. 608 p.

Obrigado pela atenção!!

Contato:
Email: **andre.flores@ufn.edu.br**