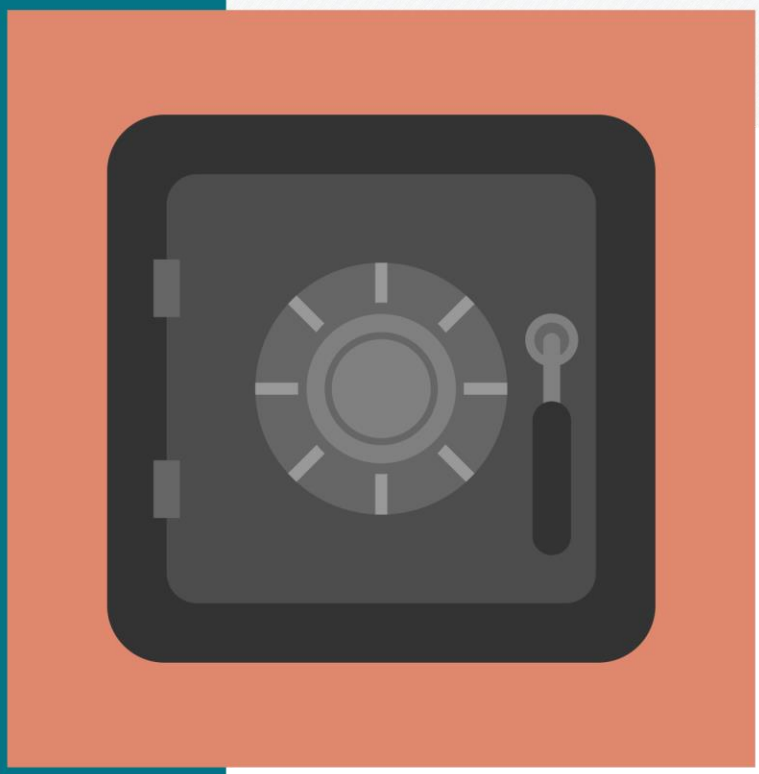
AGENDA 2

PADRÕES DE PROJETO: MVC E O ANDROID



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 II

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...

MVC (Model, View, Controller)

O padrão de desenvolvimento **Model-View-Controller (MVC)** surgiu em 1979 na corporação Xerox PARC, Trygve Reenskaug descreveu no artigo "Applications Programming in Smalltalk-80: How to use Model-View-Controller" o que seria o primeiro protótipo do desenvolvimento MVC.

Reenskaug desenvolveu o padrão dividindo a aplicação em camadas independentes denominadas: Modelo (Model), Visão (View) e Controlador (Controller). O MVC garante ao programador ou equipe de desenvolvimento uma fácil manutenção do projeto, uma vez que suas rotinas são separadas em etapas.

Na camada **modelo (model)** encontramos as classes e atributos que definem as regras de negócio com o qual o sistema vai trabalhar. As classes são responsáveis por trabalhar com os dados do sistema e interagir com Banco de Dados quando necessário.

A camada **visão (view)** é responsável pelas telas do programa, aplicativo ou até mesmo site. Nesta camada apenas o que diz respeito a interação com o usuário é o que deve ser programado, como por exemplo, os botões, caixas de texto, mensagens para o usuário, entre outros componentes.

A camada **controlador (controller)** atua intermediando as camadas View e Model. Essa camada é responsável por tarefas e ações que o software executa. Ela também é responsável por interações com outras camadas, como por exemplo, camadas que trabalham com o acesso a estruturas de Banco de Dados.

Estrutura do projeto "AppCofre"

No desenvolvimento do projeto "AppCofre" vamos utilizar alguns conceitos do MVC, e vamos dividir o projeto em camadas. O objetivo desse processo é organizar nosso código e facilitar a inclusão de uma

estrutura de Banco de Dados, que é responsável pelo armazenamento e recuperação de dados pertinentes a nossa aplicação.

O aplicativo “AppCofre” é desenvolvido para armazenar dados em forma de texto como o nome do serviço, o usuário de acesso a esse serviço e a senha utilizada para a validação do acesso para o serviço em questão. Para esse armazenamento será necessário a interligação desse aplicativo com um Banco de Dados (BD) instalado localmente no dispositivo Android, garantindo assim a integridade dos dados, mesmo quando o aplicativo for encerrado ou até mesmo após uma reinicialização do sistema operacional Android.

Para trabalhar com os dados da credencial apresentados anteriormente e no futuro com o BD, nossa aplicação conta com uma classe desenvolvida para estabelecer um modelo para as credenciais armazenadas no aplicativo. Essa classe pertence ao “**Model”** do projeto, que estabelece atributos e métodos para que os dados trafeguem internamente entre os dois extremos do projeto, que são a interface do usuário e o BD. A classe **“credencialModel.java”** apresentada na figura 3 conta com métodos **“set()”** e **“get()”** responsáveis respectivamente por atribuir e recuperar valores dos atributos internos da classe**.**

```
public class credencialModel {
    private int Id;
    private String Nome;
    private String Usuario;
    private String Senha;

    public int getId() { return Id; }

    public void setId(int id) { Id = id; }

    public String getNome() { return Nome; }

    public void setNome(String nome) { Nome = nome; }

    public String getUsuario() { return Usuario; }

    public void setUsuario(String usuario) { Usuario = usuario; }

    public String getSenha() { return Senha; }

    public void setSenha(String senha) { Senha = senha; }
}
```

Figura 3 – Classe “credencialModel” utilizada na camada “Model” do projeto.

Quando o usuário utiliza a interface gráfica do "AppCofre", ocorre uma interação entre ele e a camada **View** do projeto. A interface gráfica desenvolvida através do arquivo XML **"activity_main.xml"** ocupa a função de apresentar os dados provenientes das rotinas e execuções internas do aplicativo e capturar dados informados pelo usuário**.**

Essa interface obrigatoriamente recebe componentes para representar os dados encontrados na camada **"Model"**. A figura 4 exibe a interface gráfica do projeto "AppCofre", é importante observar que todos os atributos encontrados na camada **"Model"** estão presentes na camada **"View",** além dos botões que servem de gatilhos para desempenhar algumas funções do aplicativo.

Nos projetos desenvolvidos seguindo padrões MVC encontramos a camada **"Controller", que é** responsável pelas ações do aplicativo e que trabalha em conjunto com as camadas **"Model"** e **"View".**

No projeto **"AppCofre"** podemos dizer que nossa classe **"MainActivity.java"** recebe algumas funções exercidas pela camada **"Controller",** mesmo sabendo que ela implementa nossa camada **"View".** Essa classe recebe métodos para executar as ações de acordo com as rotinas escolhidas pelo usuário através da interface do aplicativo**.**

Figura 4 – Interface do usuário utilizada na camada "View" do projeto.

No exemplo da figura 5, após o usuário pressionar o botão **"Novo"** da interface do projeto, este botão busca um método através do **android:onClick="clickBtnNovo"** em nossa pseudocamada **"Controller".** Por sua vez o método chamado executa rotinas para remover os valores presentes nos componentes da interface ou tela do usuário.

Encontramos na camada **"Controller"** inúmeras outras funções e rotinas, inclusive as funções para manipulação de dados no BD, que é realizada em conjunto com a camada **"Model"**.

```
public void clickBtnNovo(View v)
{

    limpar();
}

public void limpar()
{
    edtNomeProg.setText("");
    edtUsuarioProg.setText("");
    edtSenhaProg.setText("");
    edtNomeProg.requestFocus();
    txtServicoProg.setText("Serviço:");
    registroAtual = -1;
}
```

Figura 5 – Métodos presentes na classe "MainActivity.java".

Encontramos na camada **“Controller”** inúmeras outras funções e rotinas, inclusive as funções para manipulação de dados no BD, que é realizada em conjunto com a camada **“Model”**.

A figura 6 demonstra o processo de comunicação que ocorre entre as camadas de um projeto que utiliza o MVC. É importante destacar a função da camada **“Controller”** que atua como um mediador entre as camadas **“View”** e **“Model”.**

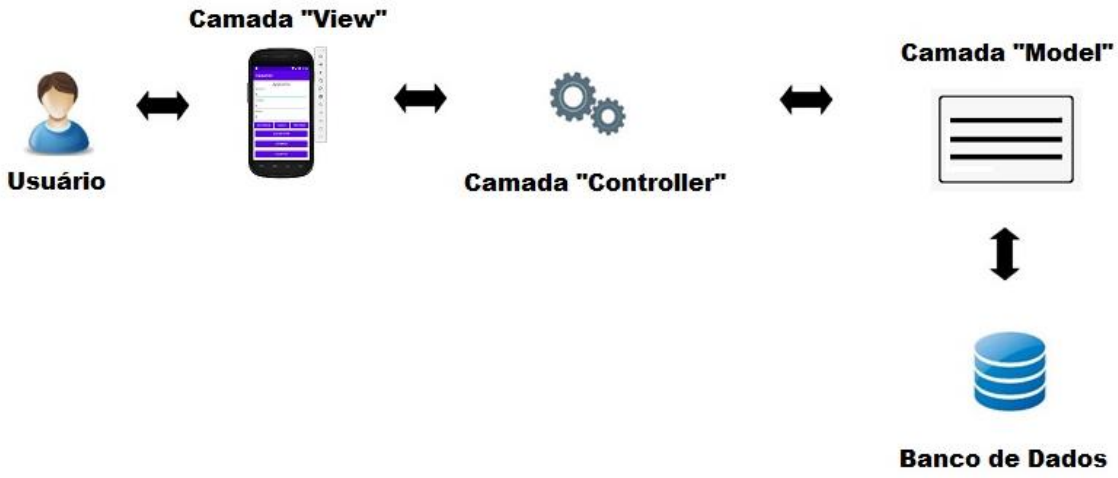


Figura 6 – Estrutura MVC.

VOCÊ NO COMANDO

AppCofre

Desenvolva um novo projeto com o nome de “AppCofre” utilizando a API 24, verifique as características na figura 7.

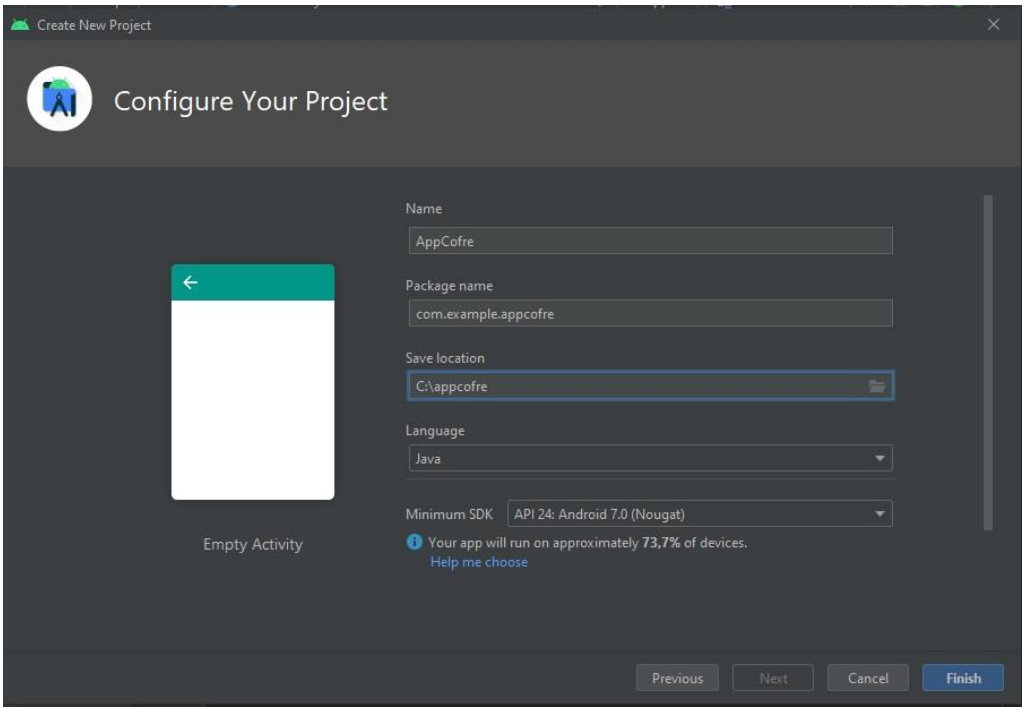


Figura 7 – Novo projeto “AppCofre”.

Desenvolva a camada **“View”** do projeto, representada melo arquivo XML **“activity_main.xml”**, utilizando como base a codificação a seguir.

```
<?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"
```

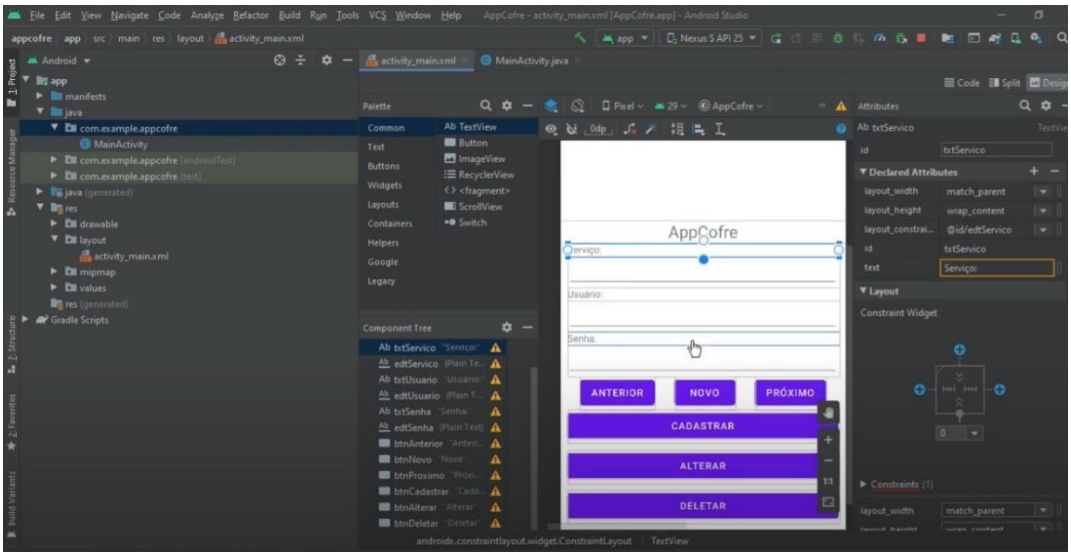
```
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@id/txtSenha"
        />

    <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: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: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: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: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: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:text="Deletar"
        app:layout_constraintBottom_toBottomOf="parent"
        tools:layout_editor_absoluteX="10dp" />

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

O resultado da codificação anterior está presente na figura 8.

Figura 8 – Tela do aplicativo "AppCofre", desenvolvida através do arquivo "activity_main.xml".

Veja o vídeo a seguir para auxiliar o desenvolvimento da classe **“credencialModel”** que representa a primeira camada “Model”. Consulte o código disponível após o vídeo para completar seu projeto.

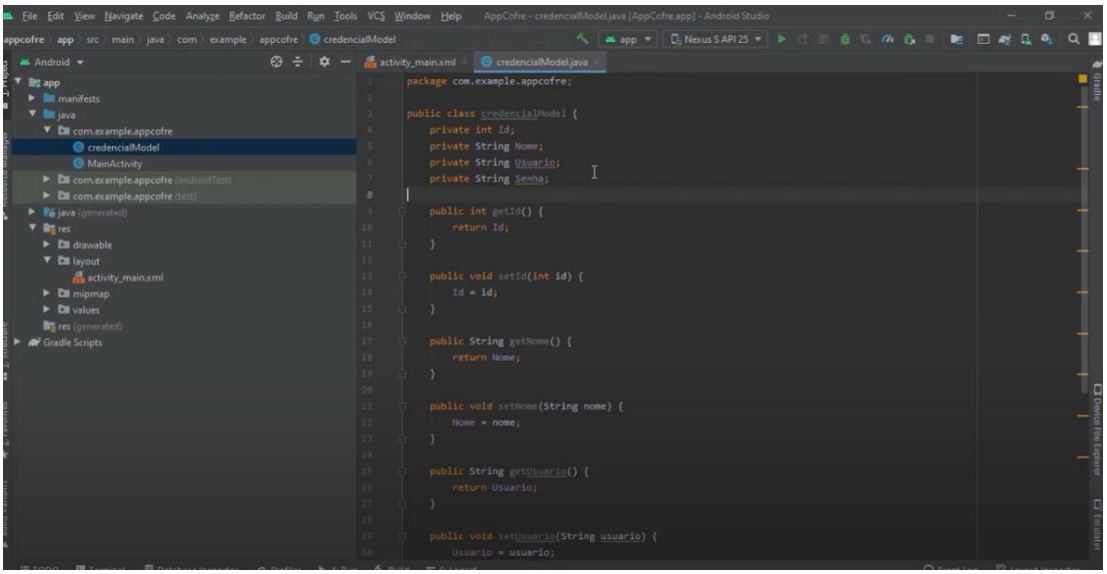


Vídeo 1 – Desenvolvimento da classe credencialModel. Link: https://youtube/ncUY9pSOL00

```
package com.example.appcofre;

public class credencialModel {
    private int Id;
    private String Nome;
    private String Usuario;
    private String Senha;

    public int getId() {
        return Id;
    }

    public void setId(int id) {
        Id = id;
    }

    public String getNome() {
        return Nome;
    }

    public void setNome(String nome) {
        Nome = nome;
    }

    public String getUsuario() {
        return Usuario;
```

```
    }

    public void setUsuario(String usuario) {
        Usuario = usuario;
    }

    public String getSenha() {
        return Senha;
    }

    public void setSenha(String senha) {
        Senha = senha;
    }
}
```

A construção da nossa pseudocamada **"Controller"** utilizando a classe "MainActivity.java" está disponível no vídeo 2.



Vídeo 2 – Desenvolvimento da classe "MainActivity". Link: https://youtu.be/wpDozBuX6oA

Como "Depurar" seu projeto

Durante o desenvolvimento de um projeto, é natural surgir a necessidade de vistoriar os códigos que estão sendo programados. O desenvolvedor pode analisar seu código para procurar as causas de um erro, ou simplesmente para monitorar se o algoritmo e/ou lógica de programação estão gerando os resultados esperados.

O Android Studio oferece a ferramenta de depuração para o desenvolvedor gerar pontos de interrupções durante a execução do seu projeto, e com essas pausas visualizar o conteúdo das variáveis e expressões em tempo real.

Para gerar uma interrupção na execução, localize a linha de código em que você quer pausar a execução e clique com o "botão esquerdo do mouse" na margem esquerda dessa linha ou posicione o cursor na linha e pressione Ctrl + F8. Para sinalizar que a interrupção foi adicionada, o Android Studio insere uma bola vermelha do lado esquerdo da linha, veja a figura 9.

Figura 9 – Gerando uma interrupção durante a execução da linha 36.

Para executar o aplicativo e gerar as interrupções solicitadas, o desenvolvedor deve escolher a opção "Debug app", como na figura 10.

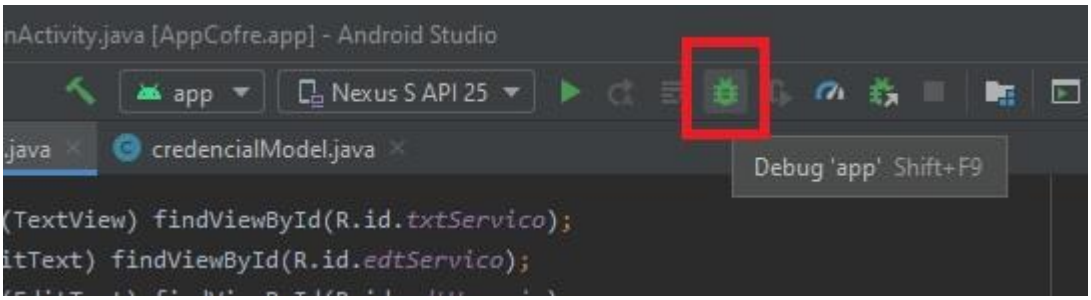
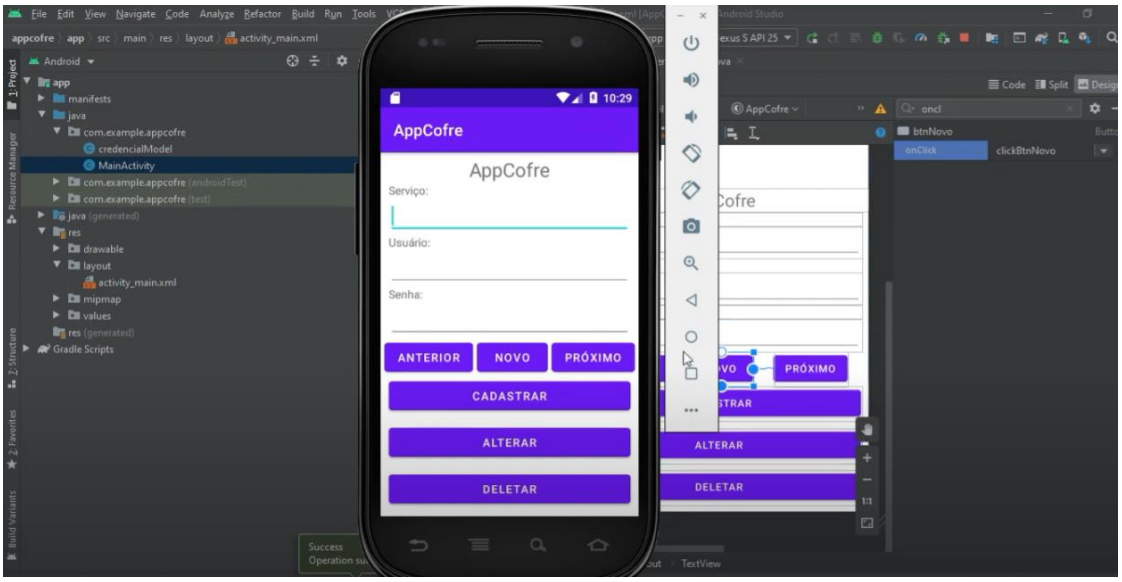


Figura 10 – Execução pelo método "Debug app".

O vídeo 3 demonstra o desenvolvimento de mais uma etapa do "MainActivity" e como gerar uma interrupção durante a execução para monitorar o conteúdo de uma variável e/ou objeto.



Vídeo 3 – Desenvolvimento da classe “MainActivity”. Link: https://youtu.be/SGljuSy2ZHE

Quando o usuário desejar alterar ou remover uma credencial, o mesmo processo utilizado para cadastrar será feito. Vamos enviar novamente os dados que estão na tela para nossa “credencialModel”.

Desenvolva com auxílio do código a seguir as rotinas para os botões de alterar e deletar uma credencial. Compare o seu projeto com o código da classe “MainActivity”.

```
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();


    @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());
    }

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

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

Compare o seu projeto com o código do arquivo “activity_main.xml” a seguir.

```
<?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: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: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>
```

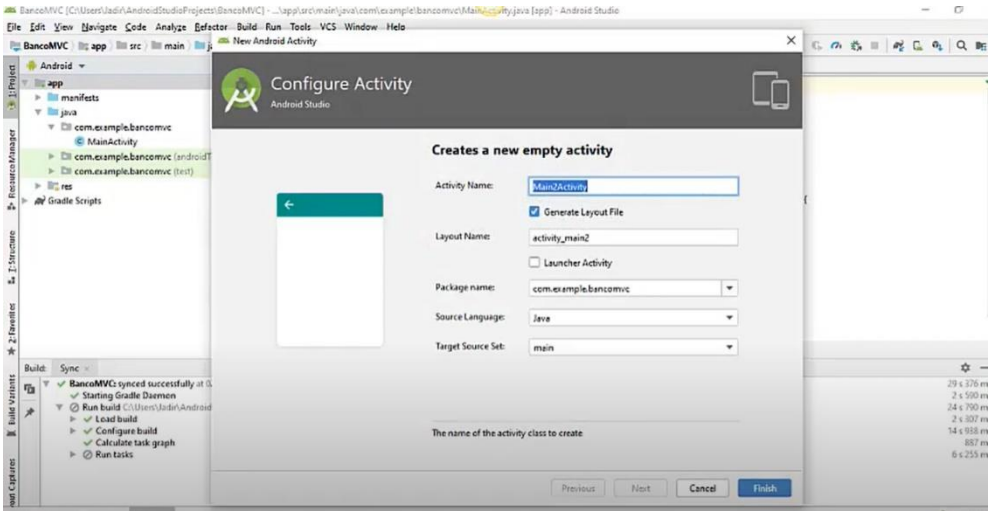
ATIVIDADE ONLINE

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.

AMPLIANDO HORIZONTES

Vídeos:



Vídeo 4 – Android MVC com SQLite - Parte 1 Canal: Jadir Mendonca. Link: https://www.youtube.com/watch?v=RHfXBnOAJZw&t=125s