



android

Prof. Diogo Soares



Adicionando resources



- Agora vamos adicionar outros recursos para deixar nosso app mais bonitinho

- Como resolver?
 - Utilizando imagens, assets
 - Formas
 - Outros tipos de layout (cardview ou linearlayout)

ImageView

- O componente ImageView permite inserir imagens na interface

- Tipos de Imagens

 - Não-Vetoriais

 - *Imagem é composta por um conjunto de “pixels”*
 - *O tamanho/qualidade da imagem depende da densidade de pixels da tela*
 - *PNG, GIF, JPG*

 - Vetoriais

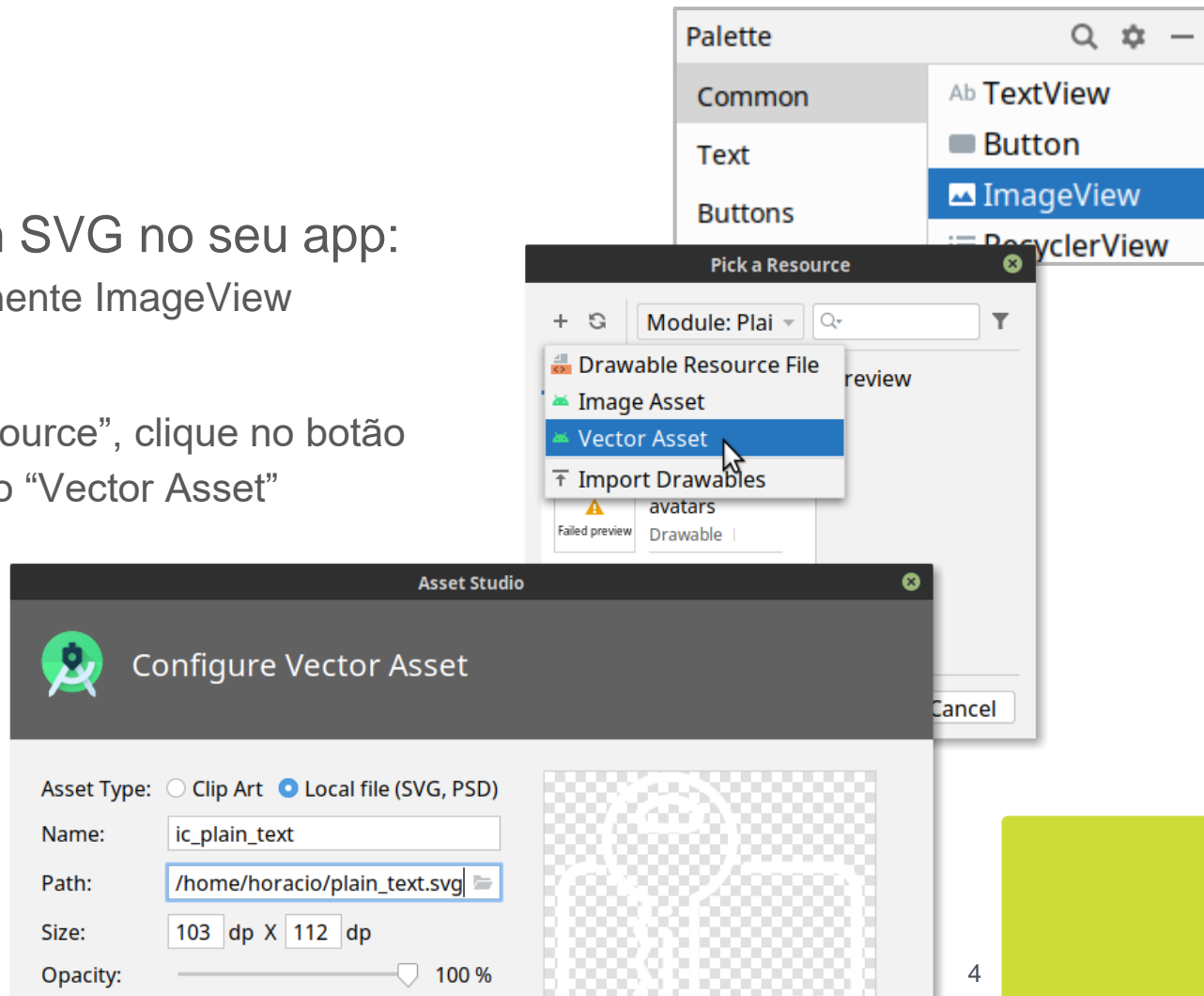
 - *Imagem é composta por um conjunto de linhas, quadrados, paths, etc*
 - *Qualidade é mantida independente do tamanho, zoom, densidade de pixes, etc*
 - *SVG, EPS*

- Como os apps executam em diferentes telas e dispositivos, recomenda-se o uso de imagens vetoriais

 - O Android tem suporte a imagens SVG

ImageView

- Para adicionar um SVG no seu app:
 - Arraste o componente ImageView para a interface
 - Na janela “Pick a Resource”, clique no botão “+” e, depois na opção “Vector Asset”
- Selecione “Local File”
- Indique o Path
- Por fim, clique em “Next” → “Finish”, e selecione a nova imagem na janela “Pick a Resource”



ImageView

Ajuste os
alinhamentos

Ajuste o ID

Palette

- Common
- Text
- Buttons
- Widgets
- Layouts
- Containers

Ab TextView

- Button
- ImageVie...
- Recycler...
- <> <fragme...
- ScrollView
- Switch

Component Tree

- ConstraintLayout
- layoutSlogan
- imageLogo
- Ab textSloganText "@stri...
- Ab textSloganAuthors "@...
- Ab textMessage "@string/m...
- Ab textLogin "@string/login"
- Ab editLogin (E-mail)
- Ab textPassword "@string/p...
- Ab editPassword (Password)
- checkSaveLogin "@string...
- buttonEnter "@string/ent...

Attributes

imageLogo

id imageLogo

Declared Attributes

Layout

Constraint Widget

0 30 0

Start → StartOf parent (30dp)

Top → TopOf parent (0dp)

Bottom → BottomOf parent (...)

layout_width 151dp

layout_height 110dp

visibility

ImageView

Copia e cola imagem na pasta drawable

Adiciona um componente imageView

Podemos adicionar imagens não vetorizadas também!

The screenshot displays the Android Studio interface with several key components highlighted by blue callout boxes:

- Top Left:** The **res/drawable** folder is selected in the Project view, indicating where to place non-vector images.
- Top Center:** The **Component Tree** shows the **headerLayout** containing an **imageView** component.
- Center:** The **Design** view shows a login form with fields for "Login" and "Senha", a "Salvar informação do login" checkbox, and an "Entrar" button. A green rectangle with the word "IMAGES" is overlaid on the design.
- Bottom Right:** The **Properties** panel for the selected **imageView** shows the **srcCompat** attribute set to **@drawable/img**.

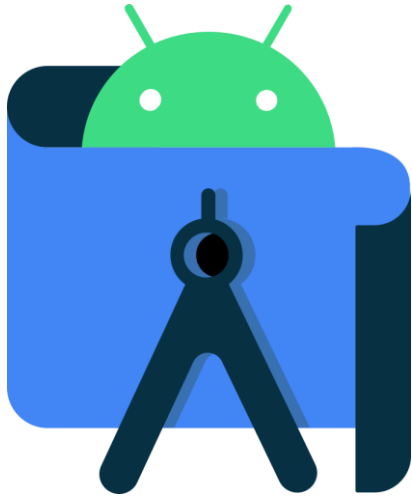
Additional annotations include:

- A blue box pointing to the **imageView** in the Component Tree with the text: "Adiciona um componente imageView".
- A blue box pointing to the **srcCompat** attribute in the Properties panel with the text: "Adiciona imagem".

ImageView



- ▣ Ademais, podemos adicionar imagens como propriedades dos componentes, como planos de fundo, etc.



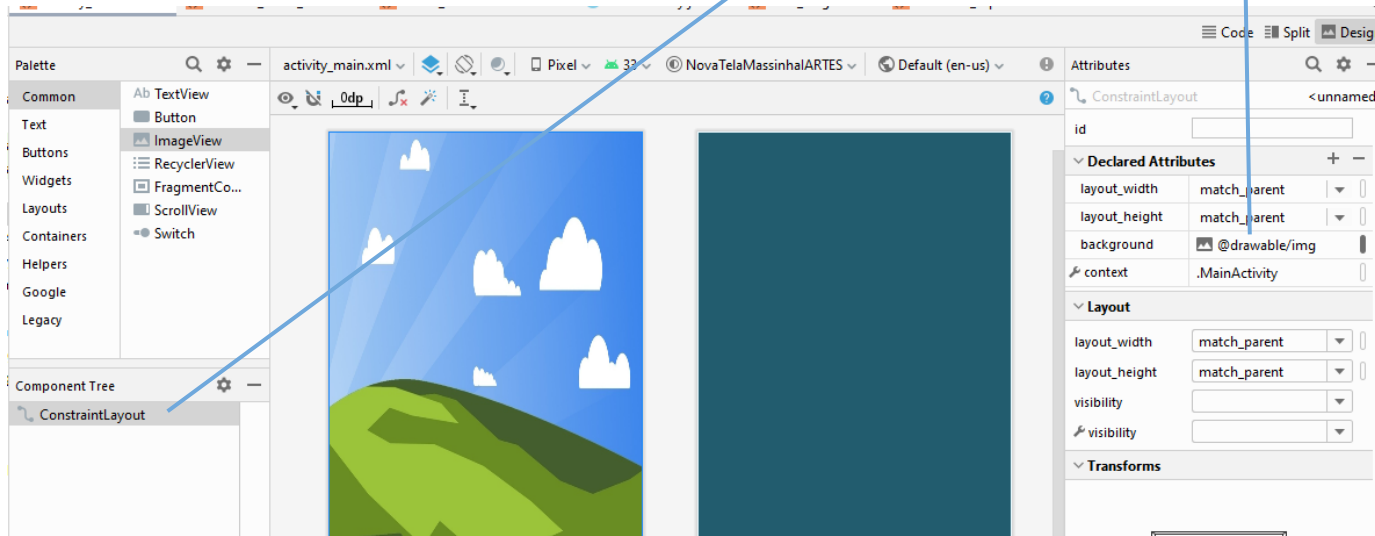
Tela de login 2.0

Prof. Diogo Soares

Melhorando nossa tela de login

Adiciona atributo background com a image

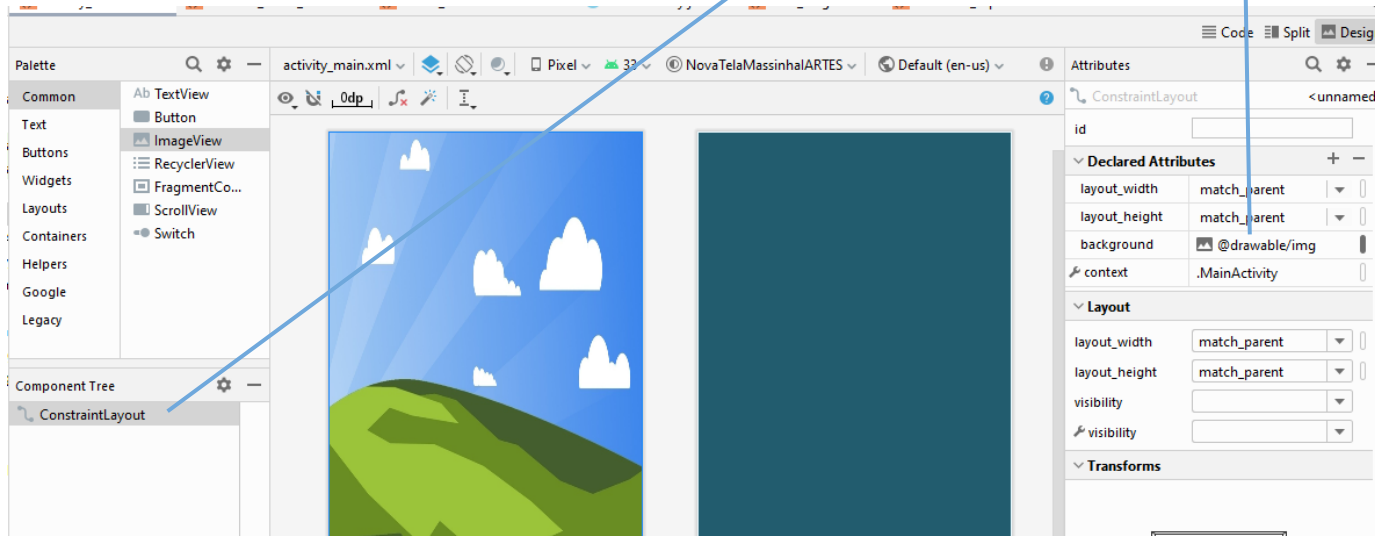
- Agora podemos adicionar resources, incluindo formas para criar versões melhores e mais bonitinhas do nosso app
- 1. Vamos adicionar um plano de fundo ao nosso layout
 - Clique em constraintLayout



Melhorando nossa tela de login

Adiciona atributo background com a image

- Agora podemos adicionar resources, incluindo formas para criar versões melhores e mais bonitinhas do nosso app
- 1. Vamos adicionar um plano de fundo ao nosso layout
 - Clique em constraintLayout



Melhorando nossa tela de login

Adiciona uma forma retangular, com cor de fundo branca, linhas pretas com 1dp de grossura e raio na borda

2. Vamos adicionar uma forma

The screenshot shows the Android Studio interface with the 'New Resource File' dialog open. The dialog fields are filled with the following information:

- File name: `forma_arredond`
- Root element: `shape`
- Source set: `main src/main/`
- Directory name: `drawable`
- Available qualifiers: (empty)

The XML code for the shape is displayed in the editor:

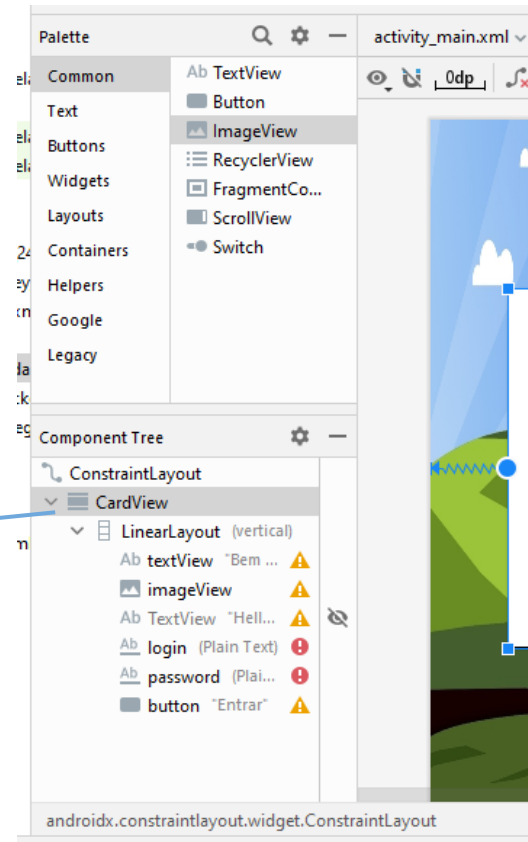
```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/white"></solid>
    <stroke android:color="@color/black"
        android:width="1dp"></stroke>
    <corners android:radius="10dp"></corners>
</shape>
```

On the right side of the editor, a preview of the resulting rounded rectangle is shown, illustrating the visual output of the XML code.

Melhorando nossa tela de login

- 3. De volta a tela principal, vamos arrumar nosso layout para um formato de cartão (CardView) com elementos alinhados um embaixo do outro (LinearLayout)

O CardView cria um formato no estilo cartão (ou container) com elementos alinhados um embaixo do outro usando LinearLayout



Melhorando nossa tela de login

No exemplo usamos gravity para centralizar todos os componentes, padding para não ficar próximo da borda

- Atenção as propriedades de cada componente (testem mesmo)

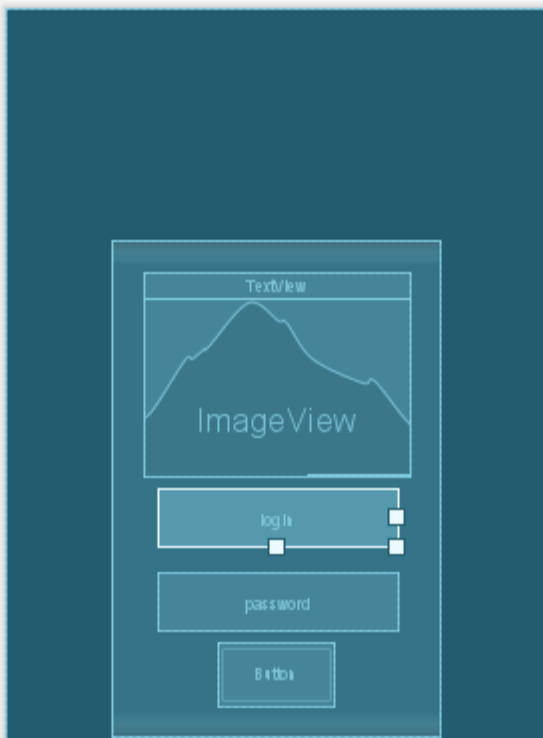
The screenshot displays the Android Studio IDE with the following components:

- Component Tree (Left):** Shows a hierarchy starting with `ConstraintLayout` containing a `CardView`, which in turn contains a `LinearLayout (vertical)`. The `LinearLayout` contains a `TextView` ("Bem vinda!"), an `imageView`, a `TextView` ("Digite seu login"), a `TextView` ("Digite sua senha"), and a `button` ("Entrar").
- Design Canvas (Center):** Shows two preview modes of the login screen. The left preview shows the final design with a blue sky background and a white login card. The right preview shows the underlying UI components (TextView, ImageView, EditText, EditText, Button) overlaid on a dark blue background.
- Properties Panel (Right):** Displays the attributes for the selected `View` component. A red circle highlights the **Declared Attributes** section, which includes:
 - `layout_width`: `match_parent`
 - `layout_height`: `match_parent`
 - `background`: `drawable/forma_arredondada`
 - `gravity`: `center` (highlighted by a red circle and a blue arrow from the text box above)
 - `orientation`: `vertical`
 - `padding`: `24dp`

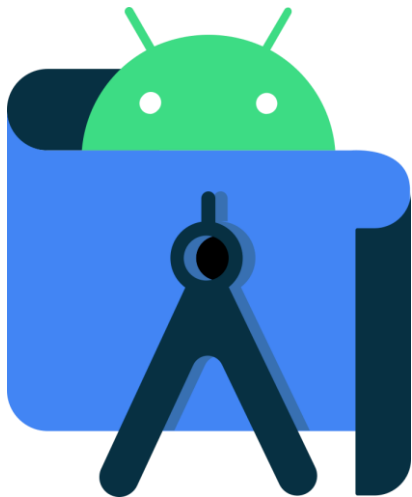
Melhorando nossa tela de login

Componente do login possui nossa forma arredondada, um ícone a direita (vector asset) e até uma dica do que escrever

- 4. Brinque com as propriedades de cada componente



id	login
Declared Attributes	
layout_width	wrap_content
layout_height	wrap_content
layout_margin	10dp
background	@drawable/forma_arr
drawableRight	@drawable/baseline_
ems	10
hint	Digite seu login
id	login
inputType	text
padding	10dp
Layout	
layout_width	wrap_content
layout_height	wrap_content



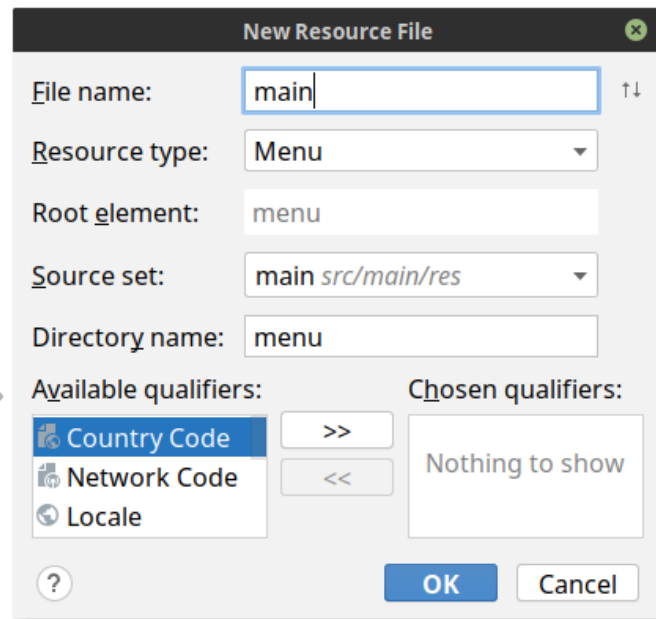
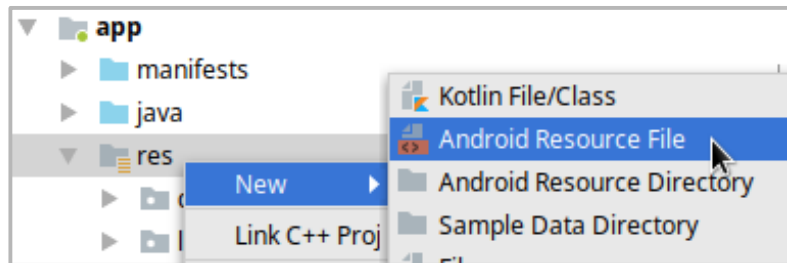
Menu

Prof. Diogo Soares



Criando um Menu

- Toda tela pode ter um menu
- Um menu é definido a partir de um arquivo XML
- Para criar um novo menu:
 - Botão direito no diretório “res”
 - New
 - Android Resource File



Criando um Menu

- Será criado um arquivo XML - res/menu/main.xml
 - Este arquivo deve ser editado para adicionar os itens do menu
 - Você pode editar o arquivo na janela de design, arrastando os componentes do menu, ou editar o XML direto.

Lembre-se de
ajustar os IDs



Criando um Menu

▣ XML resultante do slide anterior

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

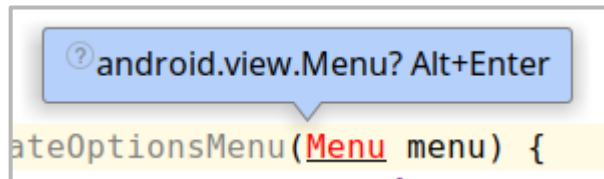
    <item android:id="@+id/configs" android:title="@string/configs" />
    <item android:id="@+id/about" android:title="@string/sobre" />
</menu>
```

Criando um Menu

- Por fim, modifique classe `MainActivity`, para usar o menu
 - Após o método `onCreate`, inclua o método abaixo:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

- Se você copiar e colar o código acima, provavelmente dará erro na classe `Menu`, pois o pacote dele não foi importado
- O Android Studio pode corrigir esse erro automaticamente, para isso, basta clicar no erro (texto vermelho sublinhado) e digitar `Alt+Enter`



Interface Gráfica

□ Resultado Final

PlainText



"The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar


Login:

Senha:

☐ Salvar informação de login

ENTRAR

PlainText



"secure password manager"
Bob and Alice

Configurações

Sobre

Digite suas credenciais para continuar

Login:

Senha:

☐ Salvar informação de login

ENTRAR

PlainText



"The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar

Login:

Senha:

☒ Salvar informação de login

ENTRAR

Interface Gráfica

- Caso não dê certo, verifique a implementação do tema (res->values->themes->themes.xml)
- Remova o “NoActionBar” do tema, as versões mais recentes do Android costumam remover o actionBar por padrão

```
<resources xmlns:tools="http://schemas.android.com/tools">
  <!-- Base application theme. -->
  <style name="Base.Theme.App1DevTitans" parent="Theme.Material3.DayNight.NoActionBar">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>
```

Interface Gráfica

- Resultado Final
- MVC
 - Note como geramos a interface inteira sem (quase) necessidade de escrever códigos Java
- Vamos agora
 - Programar ações
 - *Controllers*
 - Acessar BD
 - *Models*
 - Gerar outras telas
 - Outros

PlainText



"The most secure password manager"
Bob and Alice

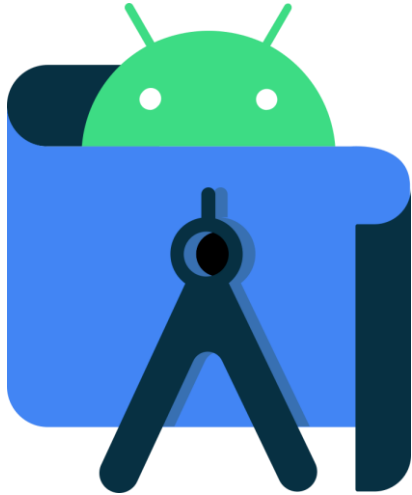
Digite suas credenciais para continuar

Login:

Senha

☐ Salvar informação de login

ENTRAR



Activities e Intents

Prof. Diogo Soares

Activity



- Activity é uma parte da aplicação que permite:
 - acesso a uma instância da tela (interface) em que os componentes UI podem ser acessados e controlados
 - interação do usuário com o aplicativo e resposta a eventos
 - instanciar classes (criar objetos) dos modelos implementados

Activity

- Normalmente, uma aplicação em Android é composta por diversas Activities
 - Cada nova tela da aplicação é uma activity
 - Uma das activities será a “main”, aberta quando a aplicação inicia
 - Uma activity chama a outra
 - *Ao fazer isso, a activity anterior para de executar, mas seu estado é guardado e, ao pressionar o botão de “voltar”, ela volta ao estado anterior*

Activity 1



PlainText

 "The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar

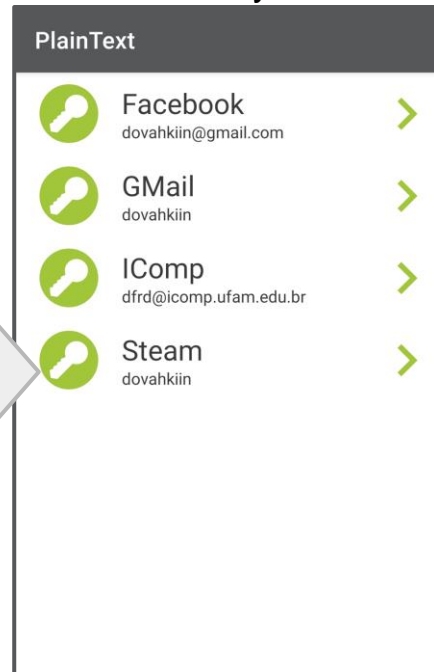
Login:

Senha:





☐ Salvar informação de login

ENTRAR

Activity 2

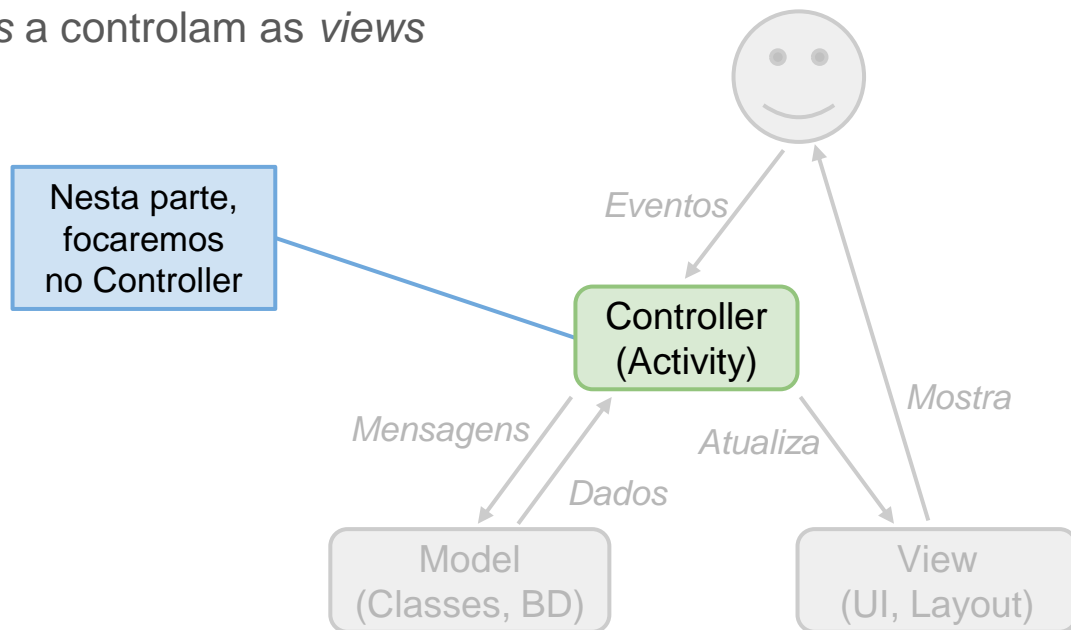


PlainText

	Facebook dovahkiin@gmail.com	>
	GMail dovahkiin	>
	IComp dfrd@icomp.ufam.edu.br	>
	Steam dovahkiin	>

Activity

- No modelo MVC, activity provê a parte do controller
 - Implementados em Java, instanciam os *modelos* a controlam as *views*



Activity

■ Toda activity estende a classe Activity (ou AppCompatActivity)

■ Principais métodos:

- ☐ onCreate
- ☐ onStart
- ☐ onPause
- ☐ onResume
- ☐ onStop

Mostra a tela
activity_main, que
criamos na parte anterior

Cria e mostra o menu,
criado na parte anterior

```
package br.edu.ufam.icomp.plaintext;

// imports ...

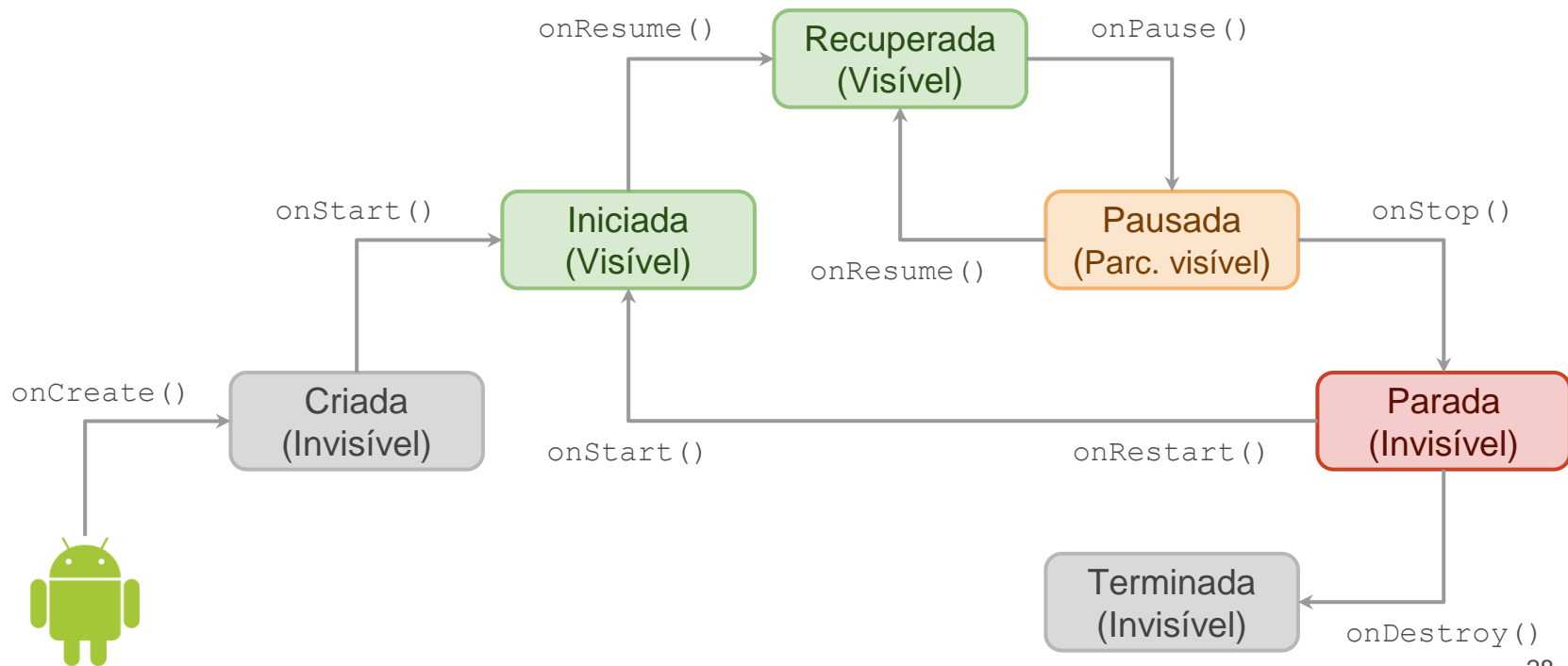
public class MainActivity extends AppCompatActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

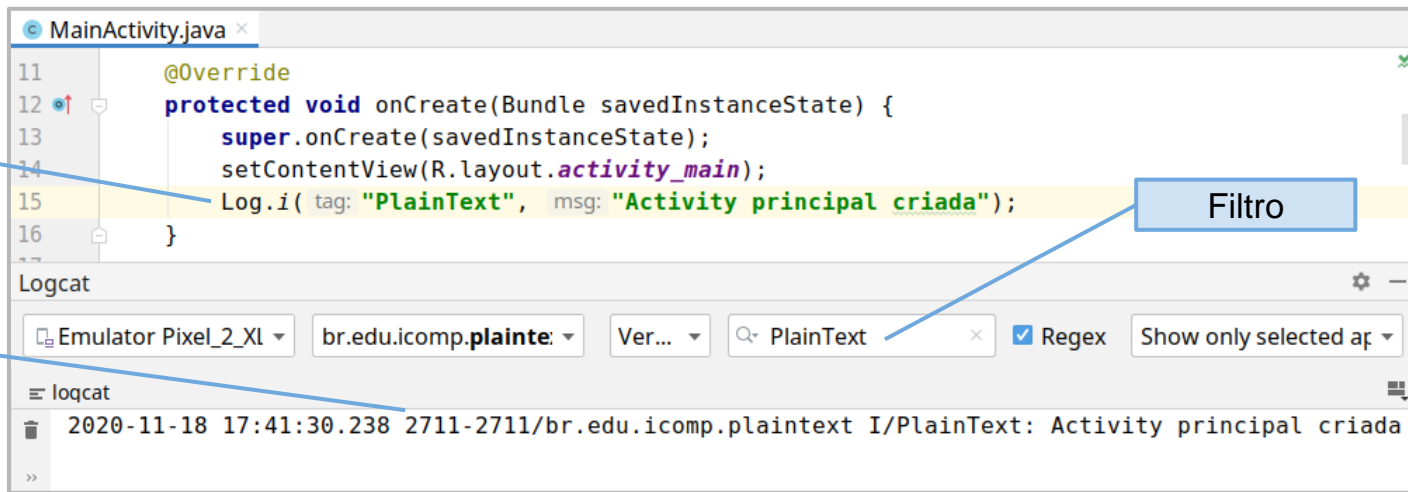
Ciclo de Vida

- Toda activity segue um ciclo de vida



Mensagens de Debug

- LogCat permite receber mensagens de debug da sua aplicação
 - Para escrever: `Log.i(String tag, String msg);`
 - Para ler: *View* → *Tool Windows* → *Android Monitor*
 - Para facilitar a leitura, crie um filtro para mostrar apenas as suas Tags
 - Os erros do Java (incluindo as Exceções), são também mostradas no LogCat, mas é recomendável também criar um filtro para os erros



Debug e Ciclo de Vida

■ Mensagens de debug nas fases do ciclo de vida

```
24 @Override
25 protected void onStart() {
26     super.onStart();
27     Log.i( tag: "PlainText", msg: "Método onStart executado");
28 }
29
30 @Override
31 protected void onResume() {
32     super.onResume();
33     Log.i( tag: "PlainText", msg: "Método onResume executado");
34 }
```

Logcat

Emulador 4.65_720p | br.edu.icomp.plaintext | Ver... | PlainText | ☒ Regex | Show only selected ap

logcat

- 2020-11-18 18:24:13.295 3449-3449/br.edu.icomp.plaintext I/PlainText: Activity principal criada
- 2020-11-18 18:24:13.308 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onStart executado
- 2020-11-18 18:24:13.312 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onResume executado
- 2020-11-18 18:25:01.589 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onPause executado
- 2020-11-18 18:25:04.981 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onStop executado
- 2020-11-18 18:25:24.011 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onRestart executado
- 2020-11-18 18:25:24.345 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onStart executado
- 2020-11-18 18:25:24.771 3449-3449/br.edu.icomp.plaintext I/PlainText: Método onResume executado

App Iniciada
(onCreate)

Botão "Home"
pressionado

Retornando à
App

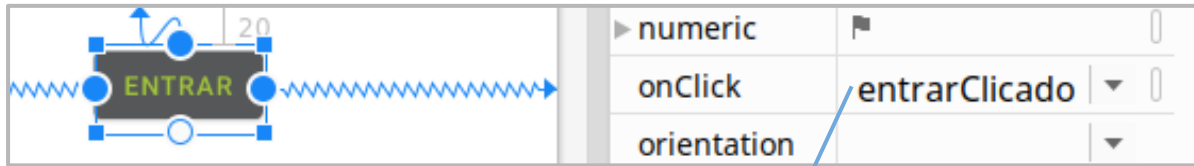
Manipulando Eventos



- Cada um dos componentes UI possui uma série de eventos
 - Botões podem ser clicados, campos de texto podem ser modificados, etc
- Alguns eventos são especificados no próprio XML do componente e executam um determinado método implementado na Activity
 - Interface do método: `public void nomeDoMetodo(View view)`
 - `View view` *contém o objeto do componente UI que gerou o evento*

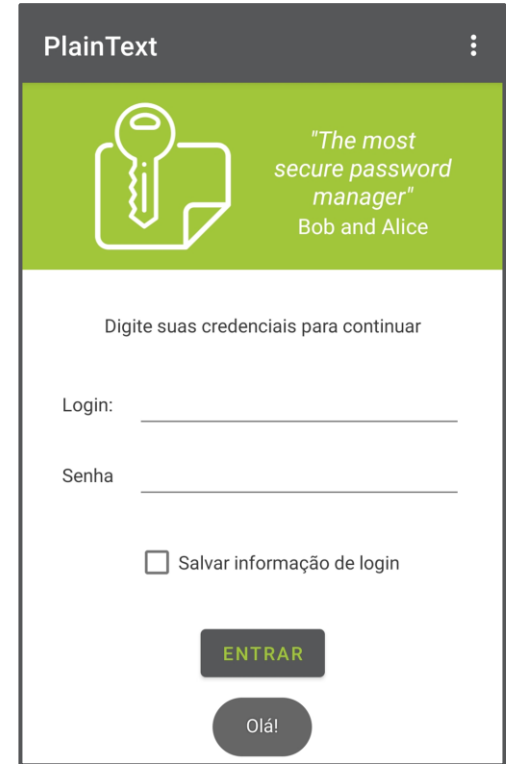
Manipulando Eventos

■ Exemplo: mostra “Olá” ao clicar no “Entrar”



Método executado
ao clicar

```
public class MainActivity extends AppCompatActivity {  
    // onCreate, onCreateOptionsMenu ...  
  
    public void entrarClicado(View view) {  
        Toast.makeText(this, "Olá!",  
        Toast.LENGTH_SHORT).show();  
    }  
}
```



Eventos do Menu

- Quando uma opção do Menu é clicada, `onOptionsItemSelected` é executado passando, como argumento, o item que foi clicado

Qual o ID do item clicado?

Item do menu que foi clicado

Foi o item "sobre"?

Constrói e mostra uma Caixa de Alerta

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.about:  
            AlertDialog.Builder alert = new  
            AlertDialog.Builder(this);  
            alert.setMessage("PlainText Password Manager v1.0")  
                .setNeutralButton("Ok", null)  
                .show();  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

Eventos do Menu

PlainText

 "The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar

Login: _____

Senha _____

☐ Salvar informação de login

ENTRAR



PlainText

 secure password manager"
Bob and Alice

Configurações

Sobre

Digite suas credenciais para continuar

Login: _____


Senha _____

☐ Salvar informação de login

ENTRAR



PlainText

 "The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar

PlainText Password Manager v1.0

OK

☐ Salvar informação de login

ENTRAR

Acessando os Componentes



- No XML dos Layouts, os componentes (botões, imagens, textos, etc) são acessados pelos seus IDs usando o “@”
 - Por isso, enfatizamos a necessidade de se atribuir os IDs aos componentes
- Mas para acessar tais componentes no Java, usa-se a classe “R”
- A classe R é gerada automaticamente e contém, dentre várias coisas, constantes para cada ID nos XMLs
 - Usamos estas constantes para referenciar os componentes a partir do Java

Acessando os Componentes

- Para acessar o componente, usamos o método `findViewById`, que tem como parâmetro o ID do componente (usando a classe R)

Acessa o componente

ID do componente

Objeto da classe
EditText

```
public void entrarClicado(View view) {  
    EditText editText = findViewById(R.id.editLogin);  
    String login = editText.getText().toString();  
    String msg = "Olá " + login + " !!";  
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();  
}
```

PlainText



"The most secure password manager"

Bob and Alice

Digite suas credenciais para continuar

Login:

Senha

☐ Salvar informação de login

ENTRAR

Olá dovahkiin !!

Abrindo outras Activities

- Um `Intent` provê uma ligação entre dois componentes que, em geral, são duas activities
 - Indica uma intenção de fazer alguma coisa

Intenção

```
public void entrarClicado(View view) {  
    Intent intent = new Intent(this, ListaActivity.class);  
    startActivity(intent);  
}
```

Execução

- A partir de agora, o botão “Entrar”, irá abrir uma nova activity
 - Entretanto, precisamos criar essa nova activity (`ListaActivity`)

Criando a Nova Activity

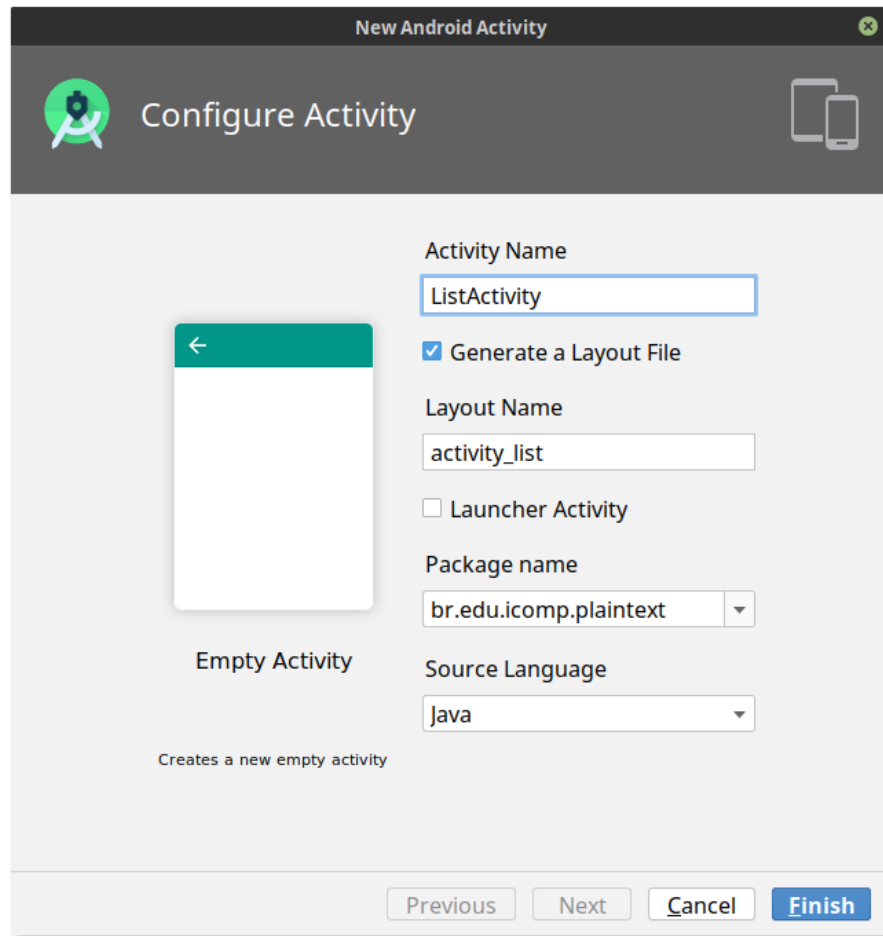
■ No Android Studio

□ File

→ New

→ Activity

→ Empty Activity

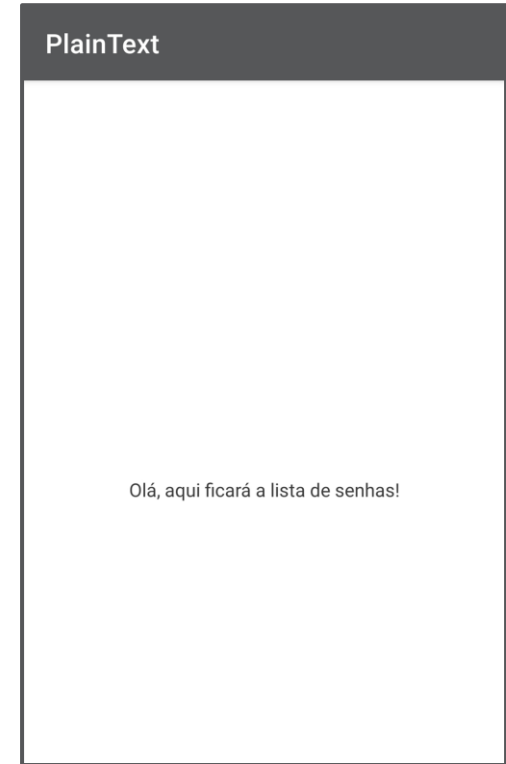


Testando a Nova Activity

- Ao clicar no botão “Entrar”, a nova activity é aberta



The screenshot shows the login screen of an application titled "PlainText". The header is dark gray with the title and a menu icon. Below the header is a green banner with a white key icon and the text: "The most secure password manager" and "Bob and Alice". The main content area is white and contains the text "Digite suas credenciais para continuar". There are two input fields: "Login:" with the value "dovahkiin" and "Senha" (empty). Below the fields is a checkbox labeled "Salvar informação de login" which is unchecked. At the bottom is a dark gray button with the text "ENTRAR" in green.



The screenshot shows the main screen of the application, titled "PlainText". The header is dark gray. The main content area is white and contains the text "Olá, aqui ficará a lista de senhas!" (Hello, here will be the list of passwords!).

Passando Dados entre Activities

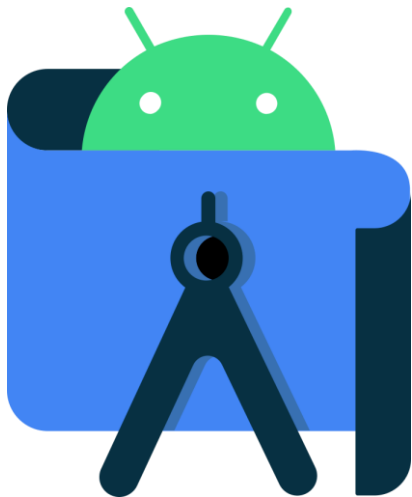
- Testando novamente o botão “Entrar”:



The login screen of the PlainText app. It features a dark grey header with the title 'PlainText' and a menu icon. Below the header is a green banner with a white key icon and the text: "The most secure password manager" and "Bob and Alice". The main content area is white and contains the text "Digite suas credenciais para continuar". There are two input fields: "Login:" with the text "dovahkiin" and "Senha" (password). Below the password field is a checkbox labeled "Salvar informação de login". At the bottom is a dark grey button with the text "ENTRAR" in green.



The welcome screen of the PlainText app. It features a dark grey header with the title 'PlainText'. The main content area is white and contains the text "Olá dovahkiin!".



Challenging

Prof. Diogo Soares



Tela de login e logado



- Tentem criar uma classe que verifica se uma senha está certa e importe no código da tela de login
 - Seu código deve retornar um alerta de senha errada na tela de login, ou passar para a tela de logado, caso a senha esteja correta