

Activities com Listas

Prof. Diogo Soares

Listas



- Grande parte das aplicações para dispositivos móveis usam listas
 - GMail – Lista de e-mails
 - Calendar – Lista de eventos
 - Notes – Lista de notas de texto
 - ToDo – Lista de coisas a fazer
 - Twitter – Lista de tweets
 - Reddit – Lista de conteúdo dos redditors
 - Facebook – Lista de posts dos amigos

- Por este motivo, o Android possui recursos avançados para a criação e gerenciamento de listas

App de Exemplo

- Nesta parte do app `PlainText`, iremos implementar a lista de senhas cadastradas
- Inicialmente, os dados serão armazenados em um `ArrayList`
 - Futuramente, mudaremos a implementação para usar o banco de dados (SQLite)

PlainText



App de Exemplo

- Para isso, as seguintes classes serão criadas/modificadas na implementação:
 - Password
 - *Modelo de uma senha*
 - PasswordDAO
 - *Data Access Object das senhas*
 - ListActivity
 - *Será modificada para conter uma lista de senhas usando o componente RecyclerView*
 - EditActivity
 - *Controlará uma tela para criar uma senha ou ver e editar uma senha existente*

PlainText



Classe Password

- Uma senha será modelada através da classe `Password`, composta pelos seguintes atributos:
 - `id`: identificador (inteiro) da senha
 - `name`: um nome para a senha (e.g., nome do site)
 - `login`: login de acesso da senha
 - `password`: a senha a ser cadastrada
 - `notes`: observações gerais

C	Password
a	id: <i>int</i>
a	name: <i>String</i>
a	login: <i>String</i>
a	password: <i>String</i>
a	notes: <i>String</i>
M ^C	Password(<i>int id</i> , <i>String name</i> , <i>String login</i> , <i>String password</i> , <i>String notes</i>)
M	Getters and Setters ...

Classe Password

■ Código-fonte (parcial) da classe:

```
public class Password {  
    private int id;  
    private String name;  
    private String login;  
    private String password;  
    private String notes;  
  
    Password(int id, String name, String login, String password, String notes) {  
        this.id = id;  
        this.name = name;  
        this.login = login;  
        this.password = password;  
        this.notes = notes;  
    }  
  
    // Getters and Setters  
}
```

Classe PasswordDAO



- Para implementar o armazenamento das senhas, criaremos a classe `PasswordDAO`
 - DAO - *Data Access Objects*, conforme feito nos slides de MySQL
- Inicialmente, esta classe irá salvar os dados em um `ArrayList`
 - Futuramente, mudaremos essa classe para usar banco de dados (SQLite)

Classe PasswordDAO



■ A classe PasswordDAO terá os seguintes métodos:

- `ArrayList<Password> getList()`
 - *Retorna a lista de senhas cadastradas*
- `boolean add(Password password)`
 - *Cadastra uma nova senha*
- `boolean update(Password password)`
 - *Atualiza uma senha já existente*
- `Password get(int id)`
 - *Retorna uma senha cadastrada através do seu id*
- Para simplificar os slides, o método para remover um item não será implementado

Classe PasswordDAO

■ Atributos e Construtor

```
public class PasswordDAO {
```

```
    private Context context;
```

```
    private static ArrayList<Password> passwordsList = new ArrayList<>();
```

```
    public PasswordDAO(Context context) {
```

```
        this.context = context;
```

```
    }
```

```
    // Classe continua com outros métodos
```

Activity usando o DAO.
Usado para toasts e BD.

Armazenamento das
senhas

Construtor, recebe a activity
(contexto) como parâmetro

Classe PasswordDAO

■ Método getList

Se a lista estiver vazia, insere algumas senhas de teste

```
public ArrayList<Password> getList() {  
    if (passwordsList.size() == 0) {  
        passwordsList.add(new Password(0, "Facebook", "dovahkiin@gmail.com",  
                                         "FusRoDah123", ""));  
        passwordsList.add(new Password(1, "GMail", "dovahkiin",  
                                         "Teste123", ""));  
        passwordsList.add(new Password(2, "IComp", "dfrd@icomp.ufam.edu.br",  
                                         "Java4242", "Mudar a senha!"));  
        passwordsList.add(new Password(3, "Steam", "dovahkiin",  
                                         "FusRoDah123", "Conta do Brasil"));  
    }  
    return passwordsList;  
}
```

Retorna a lista de senhas

Classe PasswordDAO

■ Método add

Seta o id da senha para ser um
“autoincrement”

```
public boolean add(Password password) {  
    password.setId(passwordsList.size());  
    passwordsList.add(password);  
    Toast.makeText(context, "Senha salva!", Toast.LENGTH_SHORT).show();  
    return true;  
}
```

Salva a senha no ArrayList

Classe PasswordDAO

■ Métodos update e get

```
public boolean update(Password password) {  
    passwordsList.set(password.getId(), password);  
    Toast.makeText(context, "Senha atualizada!", Toast.LENGTH_SHORT).show();  
    return true;  
}  
  
public Password get(int id) {  
    return passwordsList.get(id);  
}
```

Voltando para a Lista

- Agora que temos as classes auxiliares, vamos ver como criar uma lista na activity `ListActivity`, que criamos anteriormente
- A forma recomendada de se criar listas no Android é usando a biblioteca *RecyclerView*
 - Ele é mais complexo que as formas anteriores
 - Mas é muito mais poderoso e eficiente

RecyclerView



- ▣ Para se criar a lista usando o *RecyclerView*, deve-se
 1. Criar um componente *RecyclerView* no layout da activity (XML)
 2. Criar o layout de um item da lista (XML)
 3. Na activity, criar uma classe pro adaptador (*Adapter*)
 - O adaptador faz a ligação entre os dados, vindos do *PasswordDAO*, e a lista, que é o componente *RecyclerView*
 4. Na activity, criar uma classe pro *Holder*
 - O *Holder*, armazena as informações de um item da lista

1. RecyclerView no Layout

Arquivo: `res/layout/activity_list.xml`

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

- Palette:** The 'Common' tab is selected, showing various widgets. 'RecyclerView' is highlighted under the 'Containers' category.
- Component Tree:** The 'list_recycler' component is listed under the 'ConstraintLayout'.
- Design View:** A visual representation of the layout showing a list of items (Item 0 to Item 9) being added to the RecyclerView.
- Attributes Panel:** The 'list_recycler' component is selected, showing its attributes. The 'id' attribute is set to 'list_recycler'.
- Annotations:**
 - 'Componente RecyclerView' points to the RecyclerView widget in the Palette.
 - 'Arrasta o componente para o design' points to the design view.
 - 'ID do componente' points to the 'id' attribute in the Attributes panel.
 - 'Alinhamento' points to the 'layout_width' and 'layout_height' attributes, which are both set to 'match_parent'.

1. *RecyclerView* no Layout

■ XML do slide anterior

```
<?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=".ListActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list_recycler"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```


2. Layout do Item da Lista

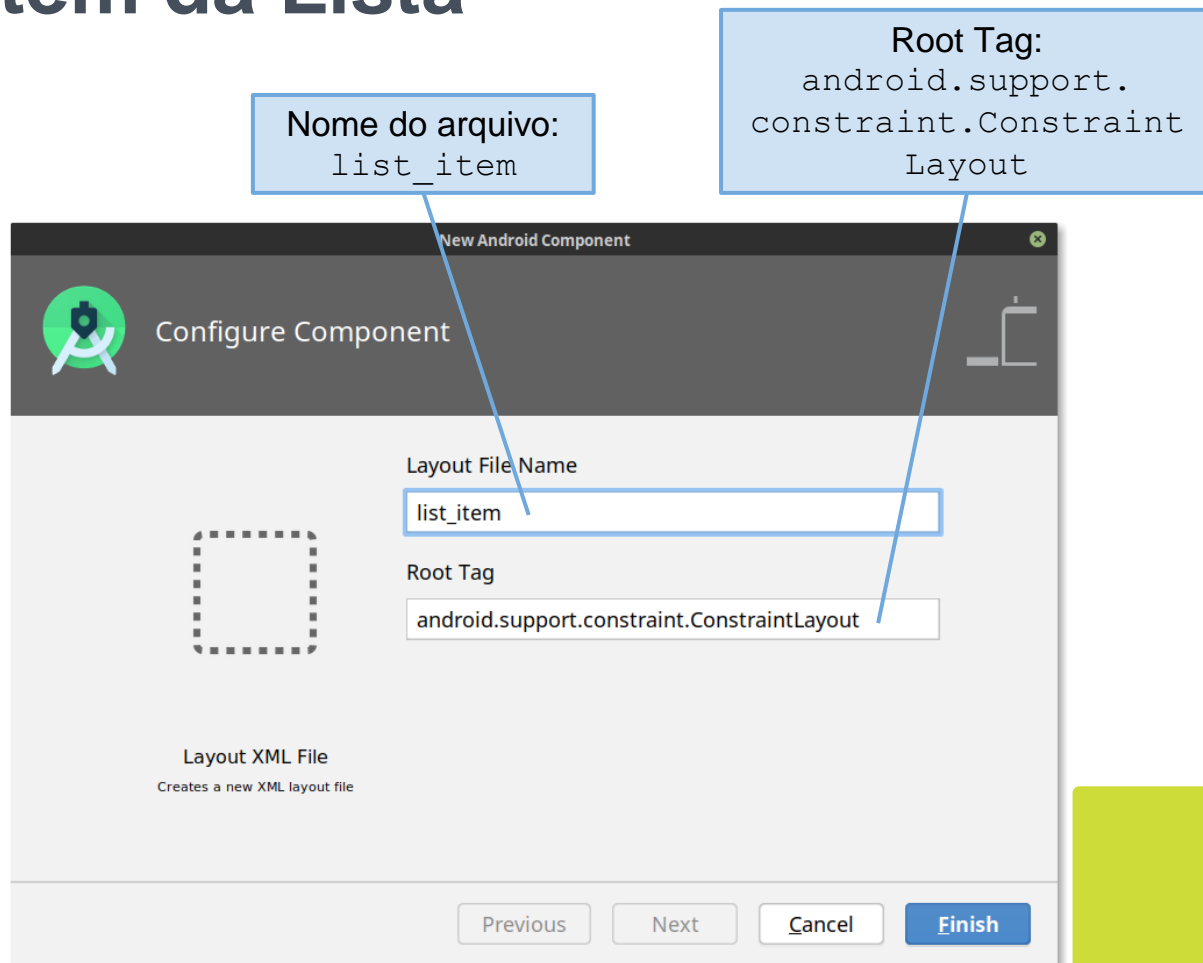


- Uma lista é composta por um conjunto de itens
- Todos os itens possuem a mesma formatação (layout)
 - Mas com “valores” diferentes
- O próximo passo para criar a lista é criar uma formatação para os itens dessa lista
 - Essa formatação é feita através de um layout (XML)

2. Layout do Item da Lista

■ Para criar o novo layout dos itens:

- File
 - New
 - XML
 - Layout XML File



2. Layout do Item da Lista

■ Design do layout:

□ Arquivo `res/layout/list_item.xml`

Altura de cada item

Note os IDs dos campos de texto

Imagem

Nome

Login

Seta

Attributes

`<unnamed>` ConstraintLayout

id

Declared Attributes

Layout

layout_width match_parent

layout_height 70dp

visibility

visibility

Transforms

2. Layout do Item da Lista (XML)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent" android:layout_height="70dp"
    android:background="?attr/selectableItemBackground">
    <ImageView
        android:id="@+id/itemIcon" android:layout_width="50dp" android:layout_height="50dp"
        android:layout_marginStart="20dp" app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_item_key" />

    <TextView
        android:id="@+id/itemName" android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_marginStart="20dp" android:text="TextView" android:textColor="#39393a" android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/itemLogin" app:layout_constraintStart_toEndOf="@+id/itemIcon"
        app:layout_constraintTop_toTopOf="@+id/itemIcon" />

    <TextView
        android:id="@+id/itemLogin" android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="TextView" android:textColor="#39393a" android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="@+id/itemIcon" app:layout_constraintStart_toStartOf="@+id/itemName" />

    <ImageView
        android:id="@+id/itemArrow" android:layout_width="55dp" android:layout_height="55dp"
        app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" app:srcCompat="@drawable/ic_right_arrow" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Adaptador



- Um adaptador **conecta dados** a um **componente**
 - Dados: no nosso exemplo, os dados vêm do método `getList` da classe `PasswordDAO`, explicado nos slides anteriores
 - Componente: É o *RecyclerView* adicionado no layout da activity
 - Componente com ID `list_recycler` do arquivo `res/layout/activity_list.xml`
- Vamos primeiro mostrar o uso do adaptador no `onCreate` da `activity` para, em seguida, implementá-lo

3. Uso do Adaptador

```
public class ListActivity extends AppCompatActivity {  
    private RecyclerView recyclerView;  
    private PasswordsAdapter adapter;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_list);  
  
        recyclerView = findViewById(R.id.list_recycler);  
        recyclerView.setLayoutManager(new LinearLayoutManager(this));  
  
        adapter = new PasswordsAdapter(this);  
        recyclerView.setAdapter(adapter);  
    }  
  
    @Override  
    protected void onRestart() {  
        super.onRestart();  
        adapter.update();  
        adapter.notifyDataSetChanged();  
    }  
}
```

Acessa o *RecyclerView* do Layout (XML)

Seta o Layout do *RecyclerView*

Seta o adaptador (criado nos slides seguintes)

Atualiza os dados da lista quando a activity for reiniciada

Recapitulando: Classe PasswordDAO

■ Atributos e Construtor

```
public class PasswordDAO {  
  
    private Context context;  
    private static ArrayList<Password> passwordsList = new ArrayList<>();  
  
    public PasswordDAO(Context context) {  
        this.context = context;  
    }  
  
    // Classe continua com outros métodos  
}
```

Activity usando o DAO.
Usado para toasts e BD.

Armazenamento das
senhas

Construtor, recebe a activity
(contexto) como parâmetro

Recapitulando: Classe PasswordDAO

■ Métodos update e get

```
public boolean update(Password password) {  
    passwordsList.set(password.getId(), password);  
    Toast.makeText(context, "Senha atualizada!", Toast.LENGTH_SHORT).show();  
    return true;  
}  
  
public Password get(int id) {  
    return passwordsList.get(id);  
}
```


3. Criação do Adaptador

- No nosso app de exemplo, a classe do adaptador que criaremos se chamará `PasswordsAdapter`
 - Como essa classe só será usada pela `ListActivity`, criaremos essa classe no mesmo arquivo (`ListActivity.java`)

- Para implementar um adaptador para o *RecyclerView*, herda-se a classe `RecyclerView.Adapter`
 - Essa classe é uma classe genérica (assim como as coleções genéricas)
 - E tem como parâmetro a classe que será usada para representar os itens da lista.
 - *Esta classe é conhecida como Holder*
 - *Será implementada futuramente*

3. Criação do Adaptador

- A classe do adaptador implementará os seguintes métodos:
 - Construtor
 - update
 - *atualiza os dados do adaptador*
 - onCreateViewHolder
 - *chamada quando uma linha da lista for criada pela primeira vez*
 - onBindViewHolder
 - *chamada quando uma linha já existente for “reciclada”, para conter informações de outro elemento da lista*
 - getItemCount
 - *retorna a quantidade total de elementos da lista*

4. Criação do Adaptador

```
class PasswordsAdapter extends RecyclerView.Adapter<PasswordsViewHolder> {  
    private Context context;  
    private ArrayList<Password> passwords;  
    PasswordDAO passwordDAO;  
  
    public PasswordsAdapter(Context context) {  
        this.context = context;  
        passwordDAO = new PasswordDAO(context);  
        update();  
    }  
  
    public void update() { passwords = passwordDAO.getList(); }  
  
    public PasswordsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        ConstraintLayout v = (ConstraintLayout) LayoutInflater  
            .from(parent.getContext())  
            .inflate(R.layout.list_item, parent, false);  
        PasswordsViewHolder vh = new PasswordsViewHolder(v, context);  
        return vh;  
    }  
  
    public void onBindViewHolder(PasswordsViewHolder holder, int position) {  
        holder.name.setText(passwords.get(position).getName());  
        holder.login.setText(passwords.get(position).getLogin());  
        holder.id = passwords.get(position).getId();  
    }  
  
    public int getItemCount() { return passwords.size(); }  
}
```

Classe Holder, explicado
no próximo slide

Activity, necessário para
acessar o BD, Toasts, e
abrir uma nova activity

Pega a lista de senhas
cadastradas

Chamado quando um
item é criado pela 1a vez

Infla o layout do item
(list_item)

Cria e retorna um objeto
da classe
PasswordsViewHolder

Atualiza os textos de um
item (holder) de acordo
com sua posição na lista

4. Criação do *Holder*

- Como mostrado no slide anterior, a classe *holder* se chama

`PasswordsViewHolder`

- Assim como no adaptador, como essa classe só será usada pela `ListActivity`, criaremos ela no mesmo arquivo (`ListActivity.java`)

- Para implementar um *holder*, herda-se a classe

`RecyclerView.ViewHolder`

- Como será possível “clicar” em um item, esta classe irá implementar a interface `View.OnClickListener`
- Neste caso, deve-se implementar o método `onClick`

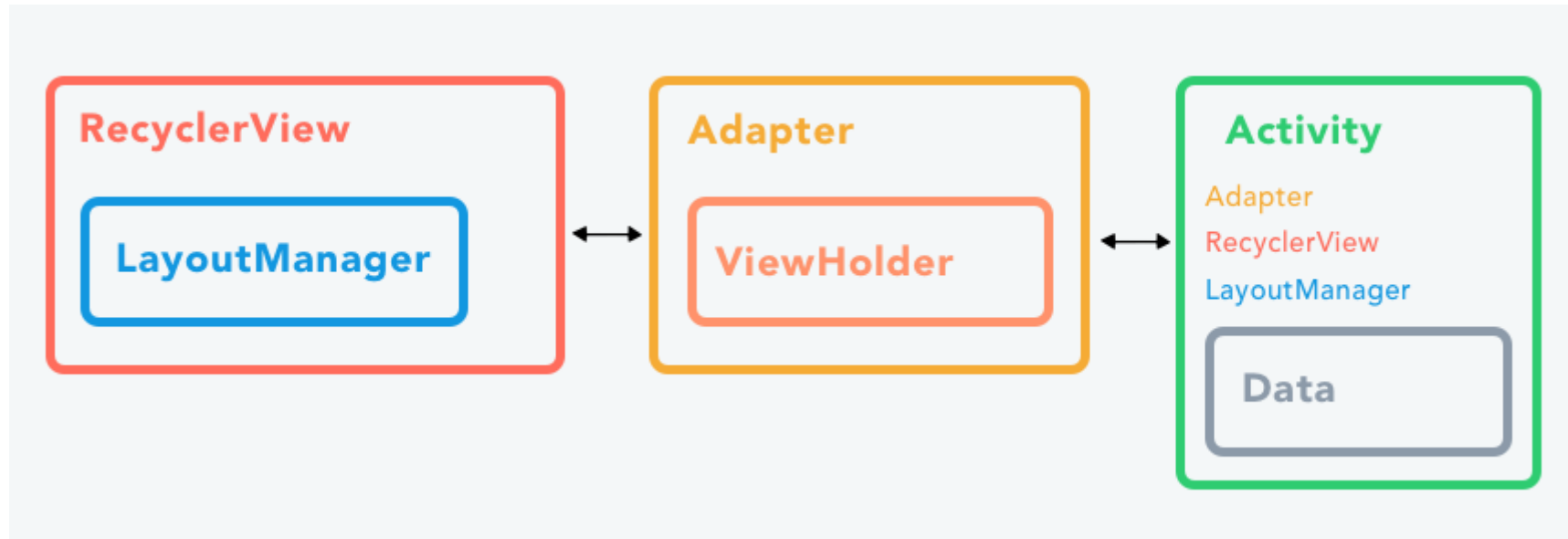
4. Criação do *Holder*

```
class PasswordsViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
    public Context context;  
    public TextView login, name;  
    public int id;  
  
    public PasswordsViewHolder(ConstraintLayout v, Context context) {  
        super(v);  
        this.context = context;  
        name = v.findViewById(R.id.itemName);  
        login = v.findViewById(R.id.itemLogin);  
        v.setOnClickListener(this);  
    }  
  
    public void onClick(View v) {  
        Toast.makeText(context, "Olá " + this.login.getText().toString(), Toast.LENGTH_LONG)  
            .show();  
    }  
}
```

Acessa os dados da view (list_item.xml) do item atual

Ao clicar em um item, o método onClick dele será chamado

Resumindo em imagem:



Resultado

- ▣ Testando o *RecyclerView*

Testando o *RecyclerView*

□ Resultado

PlainText



"The most secure password manager"
Bob and Alice

Digite suas credenciais para continuar


Login:


Senha:


☐ Salvar informação de login


ENTRAR

PlainText


 Facebook
dovahkiin@gmail.com


 GMail
dovahkiin


 IComp
dfrd@icomp.ufam.edu.br


 Steam
dovahkiin

PlainText

 Facebook
dovahkiin@gmail.com

 GMail
dovahkiin

 IComp
dfrd@icomp.ufam.edu.br

 Steam
dovahkiin

Olá dovahkiin

Finalizando o App

- Para finalizar as funcionalidades do App, será implementada a parte para adicionar uma nova senha
 - Até o momento, as senhas foram adicionadas no código-fonte
- Para isso, iremos:
 1. Modificar a `ListActivity` para:
 - *Ter um botão de adicionar senha (irá abrir uma nova activity)*
 - *Abrir a nova activity para visualizar a senha ao clicar em um item da lista*
 2. Criar uma nova activity para adicionar, editar e visualizar as senhas
 - *A mesma activity será usada para adicionar, editar e visualizar uma senha*

Adicionando o Botão de Adicionar

□ FloatingActionButton no ListActivity

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

- Palette:** The 'Buttons' category is selected, and 'FloatingActionButton' is highlighted.
- Component Tree:** The 'buttonAdd' widget is selected under the 'list_recycler'.
- Design View:** A visual representation of the UI with a list of items (Item 0 to Item 9) and a floating action button at the bottom right.
- Attributes Panel:** The 'FloatingActionButton' attributes are shown, including 'id' (buttonAdd), 'Declared Attributes', 'Layout', 'Constraint Widget', 'nextFocusUp', 'onClick' (set to 'buttonAddClick'), and 'orientation'.

Annotations in the image include:

- A blue box labeled 'FloatingActionButton' pointing to the widget in the Component Tree and the design view.
- A blue box labeled 'Método executado ao clicar no botão' (Method executed when clicking the button) pointing to the 'onClick' attribute in the Attributes panel.

Adicionando o Botão de Adicionar

XML do slide anterior

```
<?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=".ListActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list_recycler" android:layout_width="match_parent" android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/buttonAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:onClick="buttonAddClick"
        app:fabCustomSize="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:srcCompat="@android:drawable/ic_input_add" />

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

Adicionando o Botão de Adicionar

- Implementação do evento ao clicar no botão
 - Arquivo: `ListActivity.java`
 - Inicia a nova activity para adicionar/editar/visualizar (`EditActivity`)

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

Evento de Clique no Item da Lista

- Implementação do evento ao clicar em um item da lista
 - Arquivo: `ListActivity.java`
 - Dentro da classe `PasswordsViewHolder`
 - Inicia a nova activity para adicionar/editar/visualizar (`EditActivity`)

```
public void onClick(View v) {  
    Intent intent = new Intent(context, EditActivity.class);  
    intent.putExtra("passwordId", this.id);  
    context.startActivity(intent);  
}
```

Criando a Nova Activity

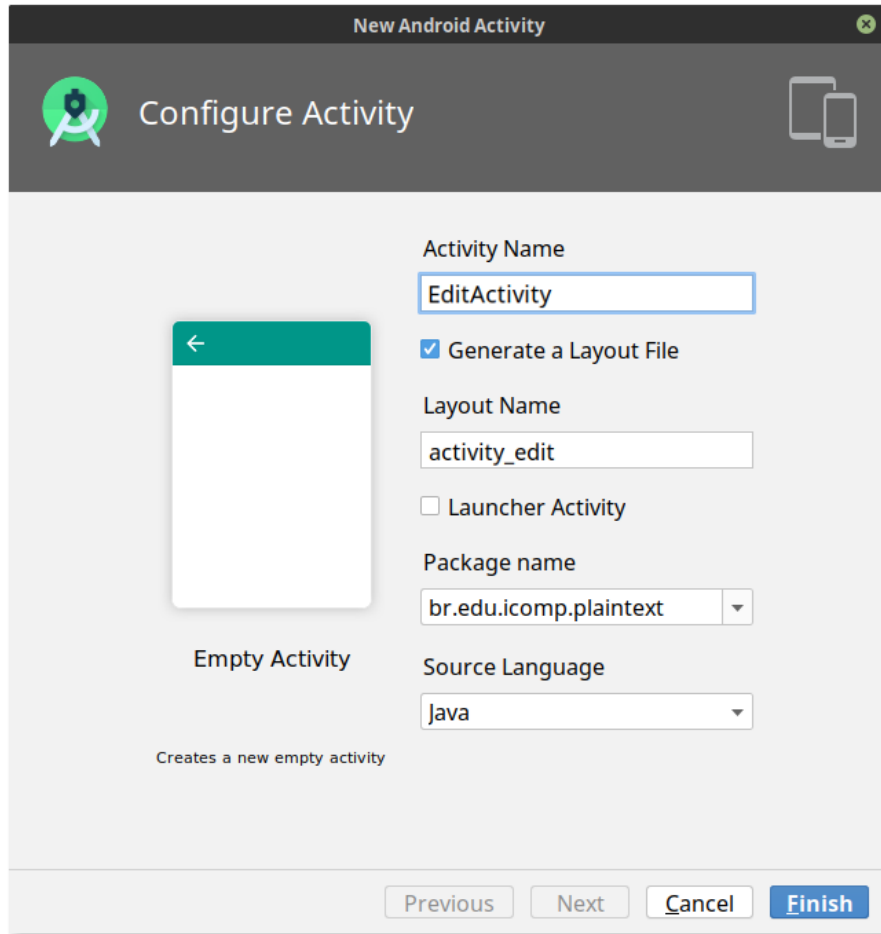
■ No Android Studio

□ File

→ New

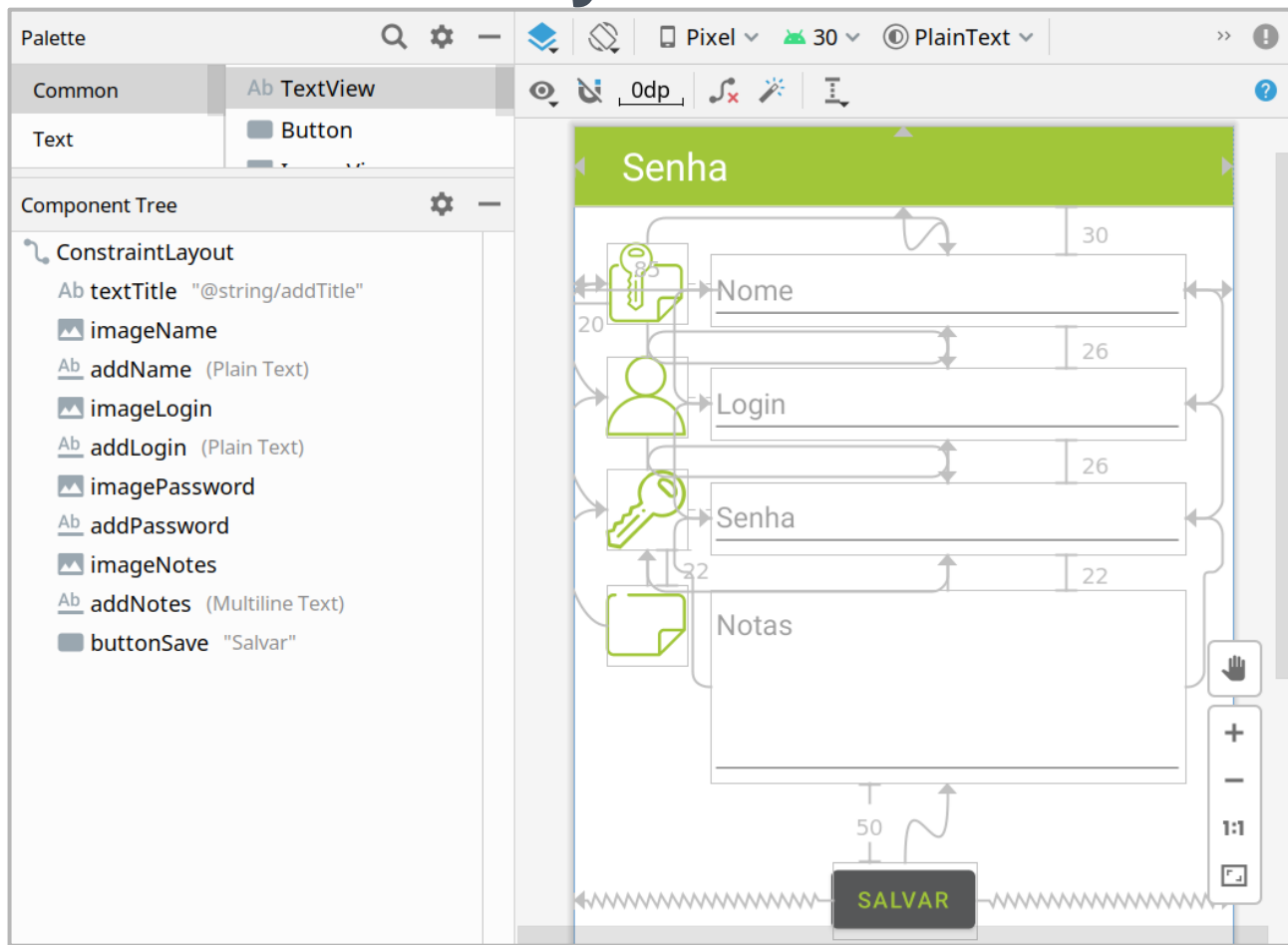
→ Activity

→ Empty Activity



Criando a Nova Activity

Layout



Criando a Nova Activity

XML do slide anterior

```
<?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/app"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".EditActivity">
    <TextView
        android:id="@+id/textTitle"
        android:paddingLeft="30dp"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:background="#a1c639"
        android:gravity="left|center_vertical"
        android:text="Senha"
        android:textAlignment="gravity"
        android:textColor="#ffffff" android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <ImageView
        android:id="@+id/imageName"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginStart="20dp"
        android:layout_marginBottom="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textTitle"
        app:srcCompat="@drawable/ic_add_name" />
```

<EditText

```
    android:id="@+id/addName"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="85dp"
    android:layout_marginTop="30dp"
    android:layout_marginEnd="30dp"
    android:ems="10"
    android:hint="Nome"
    android:inputType="textPersonName"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textTitle" />
```

<ImageView

```
    android:id="@+id/imageLogin"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginBottom="10dp"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/addLogin"
    app:srcCompat="@drawable/ic_add_user" />
```

<EditText

```
    android:id="@+id/addLogin"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="26dp"
    android:ems="10" android:hint="Login"
    android:inputType="textPersonName"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageLogin" />
```

```
<ImageView
    android:id="@+id/imagePassword"
    android:layout_width="50dp" android:layout_height="50dp"
    android:layout_marginBottom="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/addPassword"
    app:srcCompat="@drawable/ic_add_key" />
```

<EditText

```
    android:id="@+id/addPassword"
    android:layout_width="0dp"
    android:layout_marginTop="26dp"
    android:layout_height="wrap_content" android:ems="10"
    android:hint="Senha" android:inputType="text"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/addLogin" />
```

<ImageView

```
    android:id="@+id/imageNotes" android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginTop="22dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imagePassword"
    app:srcCompat="@drawable/ic_add_notes" />
```

<EditText

```
    android:id="@+id/addNotes" android:layout_width="0dp"
    android:layout_height="120dp"
    android:layout_marginTop="22dp" android:ems="10"
    android:gravity="start|top" android:hint="Notas"
    android:inputType="textMultiLine"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/addPassword" />
```

<Button

```
    android:id="@+id/buttonSave"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:onClick="salvarClicado" android:text="Salvar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/addNotes" />
```


Criando a Nova Activity

■ EditActivity

Acessa os componentes da tela (e atribui para os atributos da classe)

DAO para gerenciar as senhas

Tenta acessar o passwordId, enviado ao abrir a activity

Se o passwordId foi enviado, pega os dados da senha e preenche os campos de texto da tela

```
public class EditActivity extends AppCompatActivity {  
    private PasswordDAO passwordDAO;  
    private int passwordId;  
    private TextView editName, editLogin, editPassword, editNotes;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_edit);  
  
        editName = findViewById(R.id.addName);  
        editLogin = findViewById(R.id.addLogin);  
        editPassword = findViewById(R.id.addPassword);  
        editNotes = findViewById(R.id.addNotes);  
        passwordDAO = new PasswordDAO(this);  
  
        Intent intent = getIntent();  
        passwordId = intent.getIntExtra("passwordId", -1);  
  
        // Verifica se uma senha foi passada como parâmetro  
        if (passwordId != -1) {  
            Password password = passwordDAO.get(passwordId);  
            editName.setText(password.getName());  
            editLogin.setText(password.getLogin());  
            editPassword.setText(password.getPassword());  
            editNotes.setText(password.getNotes());  
        }  
    }  
}  
  
// Método saveClicked (próximo slide)
```

Clicando no Botão de Salvar

- Ao clicar no botão de salvar, o seguinte método será executado
 - Arquivo `EditActivity.java`, dentro da classe `EditActivity`

```
public void salvarClicado(View view) {  
    Password password = new Password(passwordId,  
editName.getText().toString(),  
        editLogin.getText().toString(), editPassword.getText().toString(),  
        editNotes.getText().toString());  
  
    boolean result;  
    if (passwordId == -1) result = passwordDAO.add(password);  
    else result = passwordDAO.update(password);  
  
    if (result) finish();  
}
```

Cria um novo Password (modelo)

É uma nova senha, adiciona

É uma senha já existente, atualiza

Se deu certo, volta
para a activity
anterior (lista)

Resultado








- ▣ Testando a nova activity

Testando a Nova Activity

□ Resultado





PlainText


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








PlainText


Senha

-  Coursera _____
-  sovngarde _____
-  HunKaalZoor _____
-  Notas



PlainText

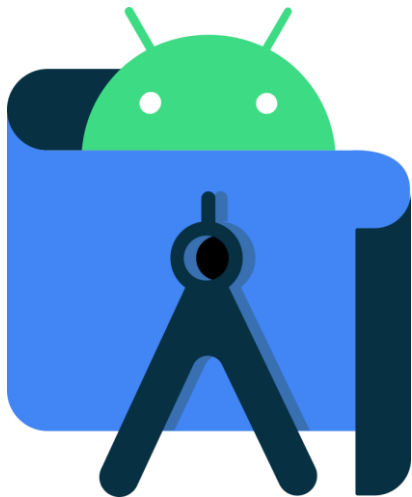
-  Facebook
dovahkiin@gmail.com >
-  GMail
dovahkiin >
-  IComp
dfrd@icomp.ufam.edu.br >
-  Steam
dovahkiin >
-  Coursera
sovngarde >



Conclusão



- Ao reiniciar o aplicativo, os dados inseridos são perdidos, pois eles não foram armazenados, estavam apenas na memória
- A seguir, iremos modificar o `PasswordDAO` para salvar os dados no banco de dados (SQLite), de forma que os dados fiquem armazenados permanentemente



Challenging

Prof. Diogo Soares



Sunflower

- Usando apenas recyclerView com gridLayout, tente criar uma lista similar ao do código Sunflower, da android
- Link: [android/sunflower at compose_recyclerview \(github.com\)](https://github.com/android/sunflower)
- Ex:

