ANDROID PARA TODES

Guia Inclusivo para a Criação do Seu Primeiro Aplicativo

Paula Justino

ANDROID PARA TODES

Guia Inclusivo para a Criação do Seu Primeiro Aplicativo

Autora
Paula Justino

maio, 2024

SUMÁRIO

INTRODUÇÃO

Seja bem-vinde ao mundo do desenvolvimento de aplicativos Android! Este guia foi cuidadosamente elaborado para proporcionar uma experiência de aprendizado acessível e inclusiva, especialmente voltada para você que deseja dar os primeiros passos na criação de aplicativos.

Vamos criar juntes um aplicativo simples de lista de tarefas, onde você poderá ver, adicionar e remover tarefas. Este projeto será uma oportunidade para explorarmos os fundamentos do desenvolvimento Android de maneira prática.

Ao longo deste guia, garanto explicações claras e compreensíveis. Não importa se você é iniciante na programação ou busca uma introdução amigável ao desenvolvimento Android, este guia é para você!

Vamos começar esta jornada? Estou animada para compartilhar este conhecimento com você!

01

CONFIGURANDO O AMBIENTE DE DESENVOLVIMENTO

CONFIGURANDO O AMBIENTE DE DESENVOLVIMENTO

INSTALANDO O ANDROID STUDIO

Antes de começar, é necessário preparar o ambiente de desenvolvimento do seu aplicativo.

O que é o Android Studio?

Android Studio é um programa gratuito que possui todas as ferramentas necessárias para criar aplicativos Android. Nele, você é capaz de desenhar o visual, implementar suas funcionalidades e testar o funcionamento do seu aplicativo.

Passos para instalar o Android Studio:

1. Acesse o site oficial: developer.android.com/studio.
2. Baixe o instalador para seu sistema operacional (Windows, macOS ou Linux).
3. Execute o instalador e siga as instruções dele para concluir a instalação.

02

CRIANDO O PROJETO

CRIANDO O PROJETO

CRIANDO O PROJETO NO ANDROID STUDIO

Passos para criar um projeto:

1. Abra o Android Studio e clique no botão **New Project**.

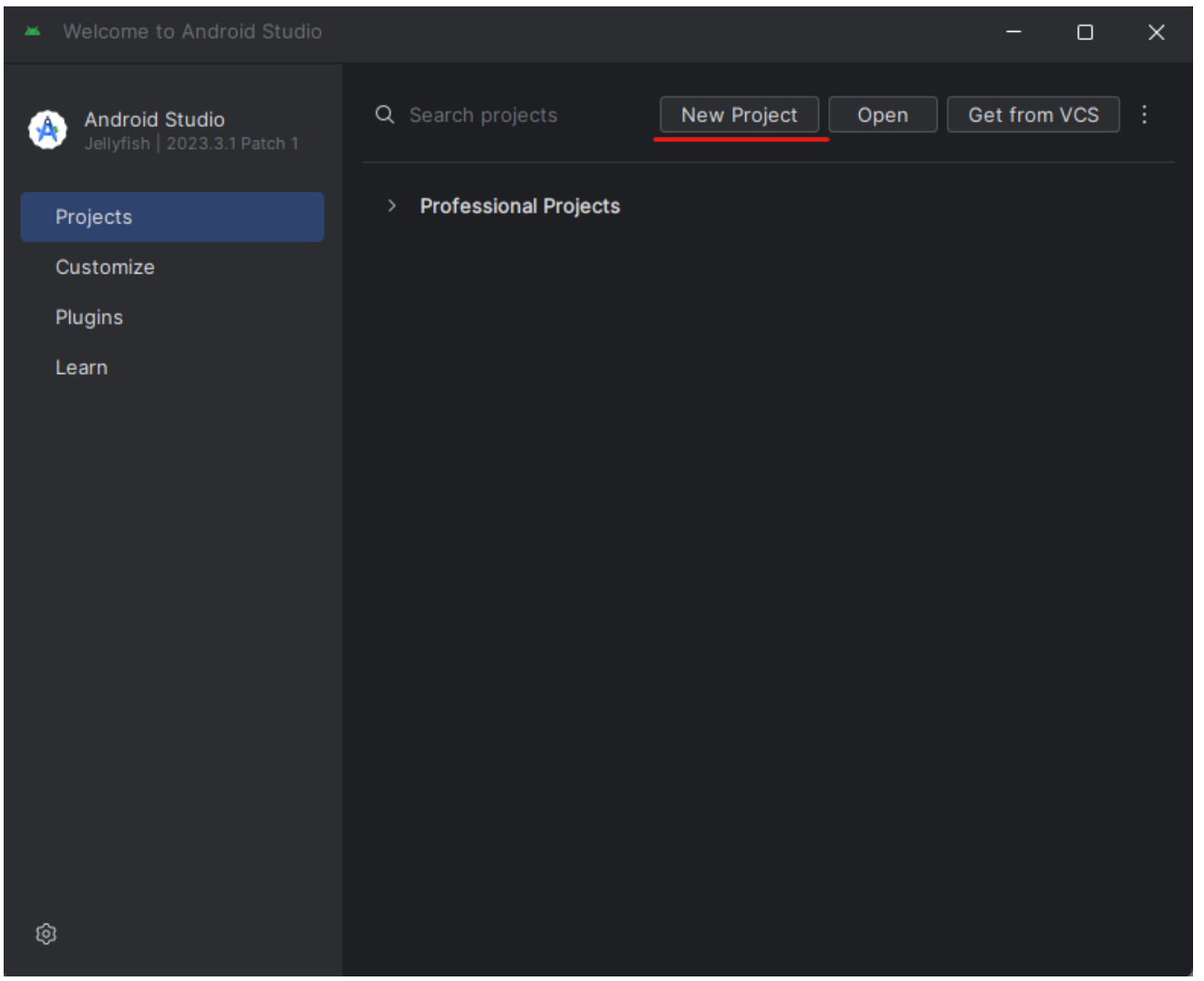

Figura 1: Interface inicial do Android Studio, com o botão “New Project” sublinhado em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO O PROJETO

2. Clique em **Empty Views Activity**, e depois em **Next**.

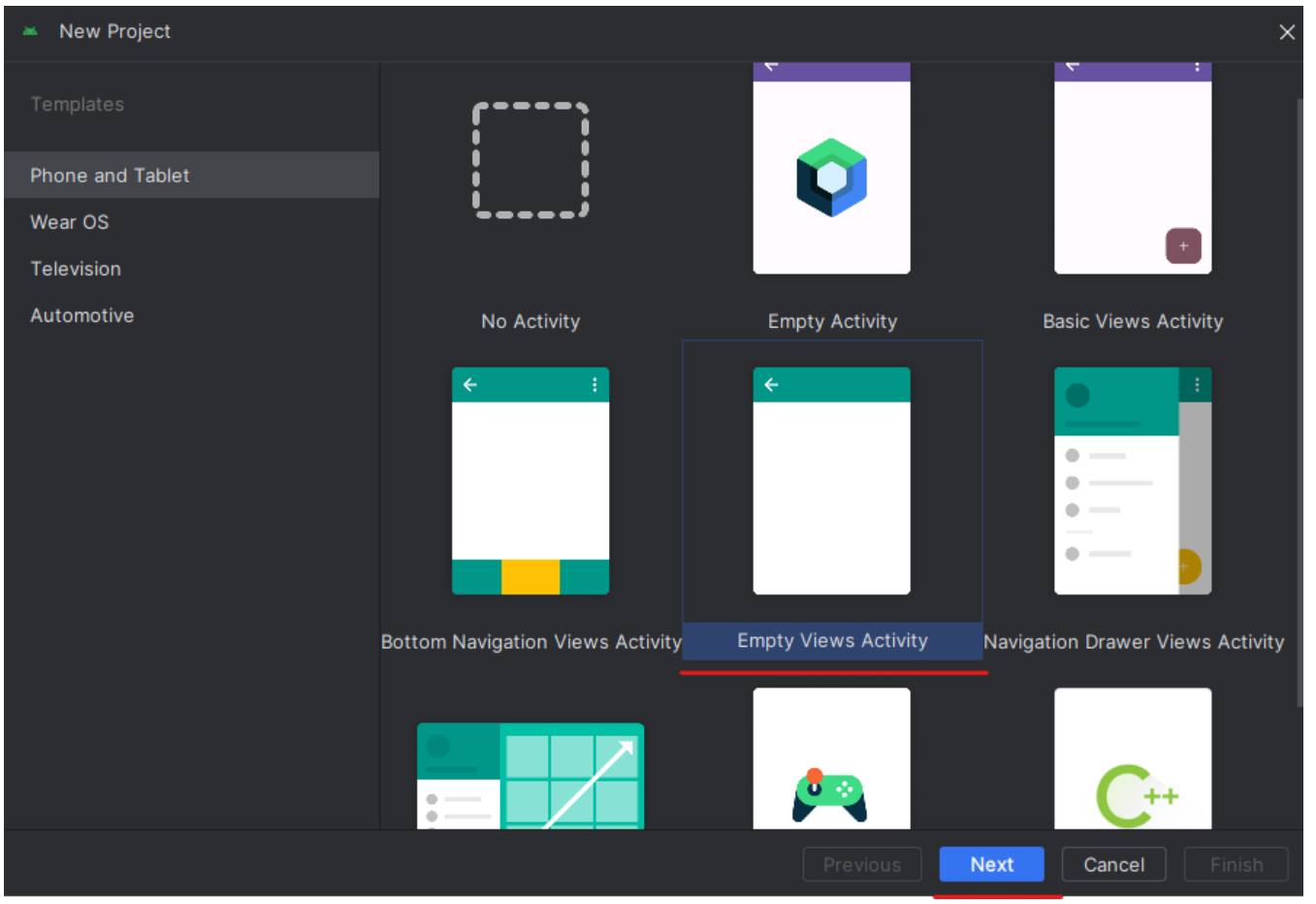

Figura 2: Interface de New Project do Android Studio, com a opção “Empty Views Activity” e o botão “Next” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO O PROJETO

3. Dê um nome ao projeto (por exemplo: ToDoList); selecione a linguagem **Kotlin**, e depois clique em **Finish**.

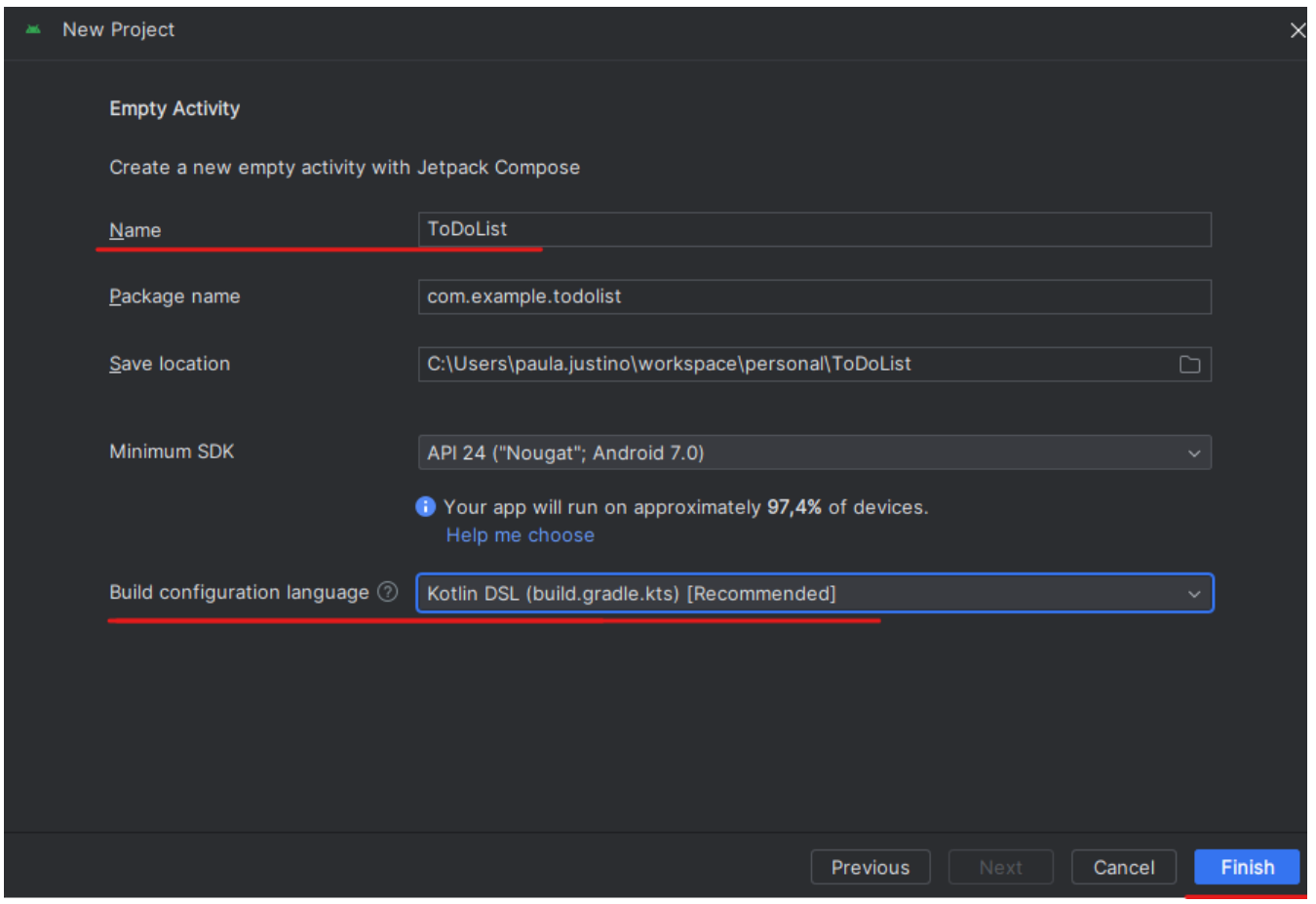

Figura 3: Interface de New Project do Android Studio, com os campos “Name”, “Build configuration language” e o botão “Finish” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

03

CRIANDO A INTERFACE DO USUÁRIO

CRIANDO A INTERFACE DO USUÁRIO

ADICIONANDO UM LAYOUT SIMPLES

O que é uma interface? E um layout?

Uma **interface** representa uma tela do aplicativo. Onde as interações entre humano e aparelho acontecem. Ela é desenvolvida através de um **layout**, que define sua apresentação. Pense no layout como um esqueleto que organiza os elementos visuais da interface, como textos, botões, imagens, entre outros e outros.

CRIANDO A INTERFACE DO USUÁRIO

Passos para criar um layout:

1. No seu projeto, abra as pastas na seguinte ordem: **app**, **res** e **layout**. Depois, abra o arquivo **activity_main.xml**. Nele criaremos o layout.

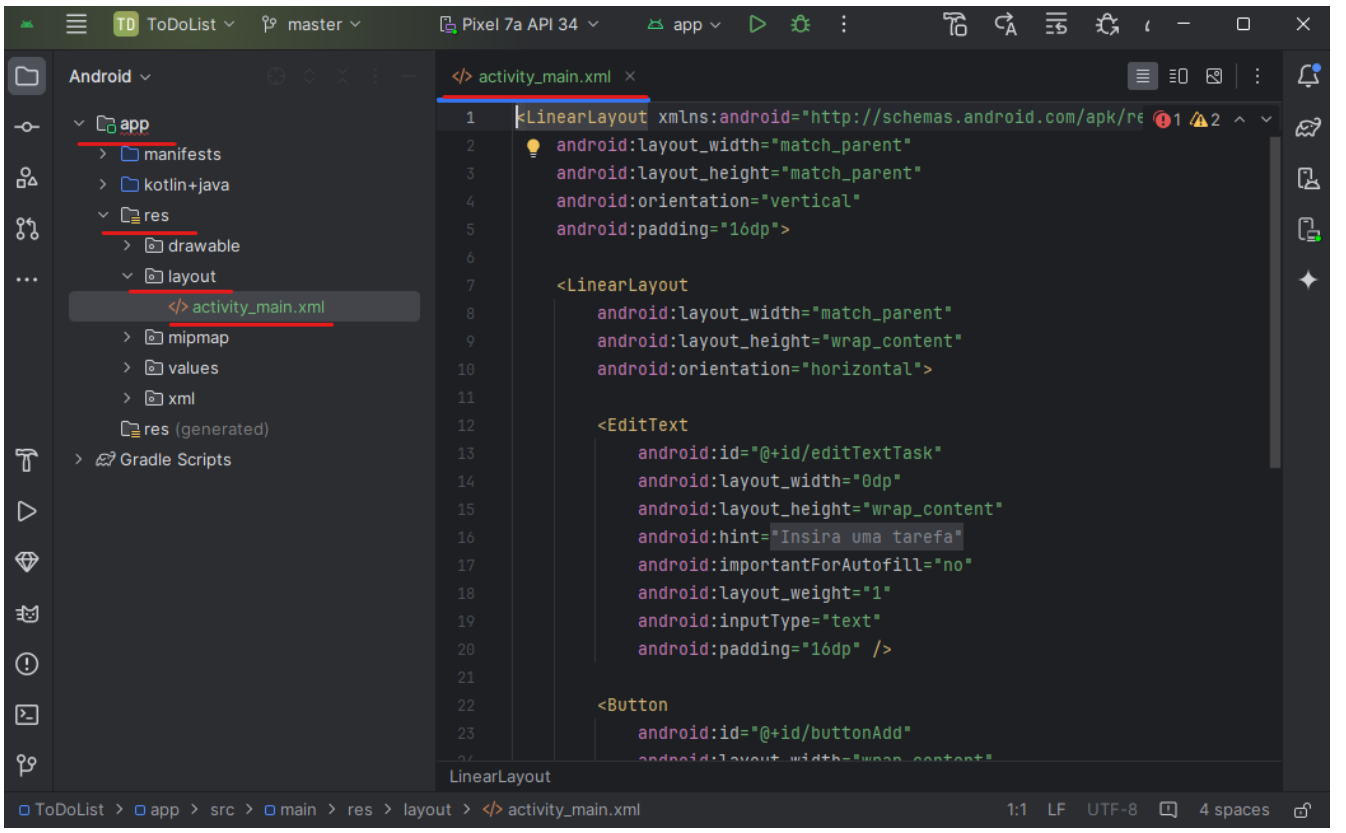

Figura 4: Interface do Android Studio, exibindo o projeto ToDoList e a hierarquia de pastas dele. As pastas “app”, “res”, “layout” e o arquivo “activity_main.xml” estão abertos e sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO A INTERFACE DO USUÁRIO

2. Apague todo o conteúdo pré-existente no arquivo e adicione o código abaixo. Não se apegue a tentar compreender cada linha desse código agora.

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/editTextTask"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:hint="Insira uma tarefa"
            android:importantForAutofill="no"
            android:layout_weight="1"
            android:inputType="text"
            android:padding="16dp" />

        <Button
            android:id="@+id/buttonAdd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:background="@color/white"
            android:padding="0dp"
            android:text="Adicionar"
            android:textSize="18sp" />
    </LinearLayout>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerViewTasks"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:padding="16dp" />
</LinearLayout>
```

Figura 5: Código em formato xml, com a implementação do layout activity_main.xml do app ToDoList. Os componentes usados são: LinearLayout, EditText, Button e RecyclerView. Fonte: Captura de tela do código, 2024.

CRIANDO A INTERFACE DO USUÁRIO

3. Vamos criar um layout auxiliar. Clique com o botão direito na pasta **layout**; clique em **New** e depois em **Layout Resource File**.

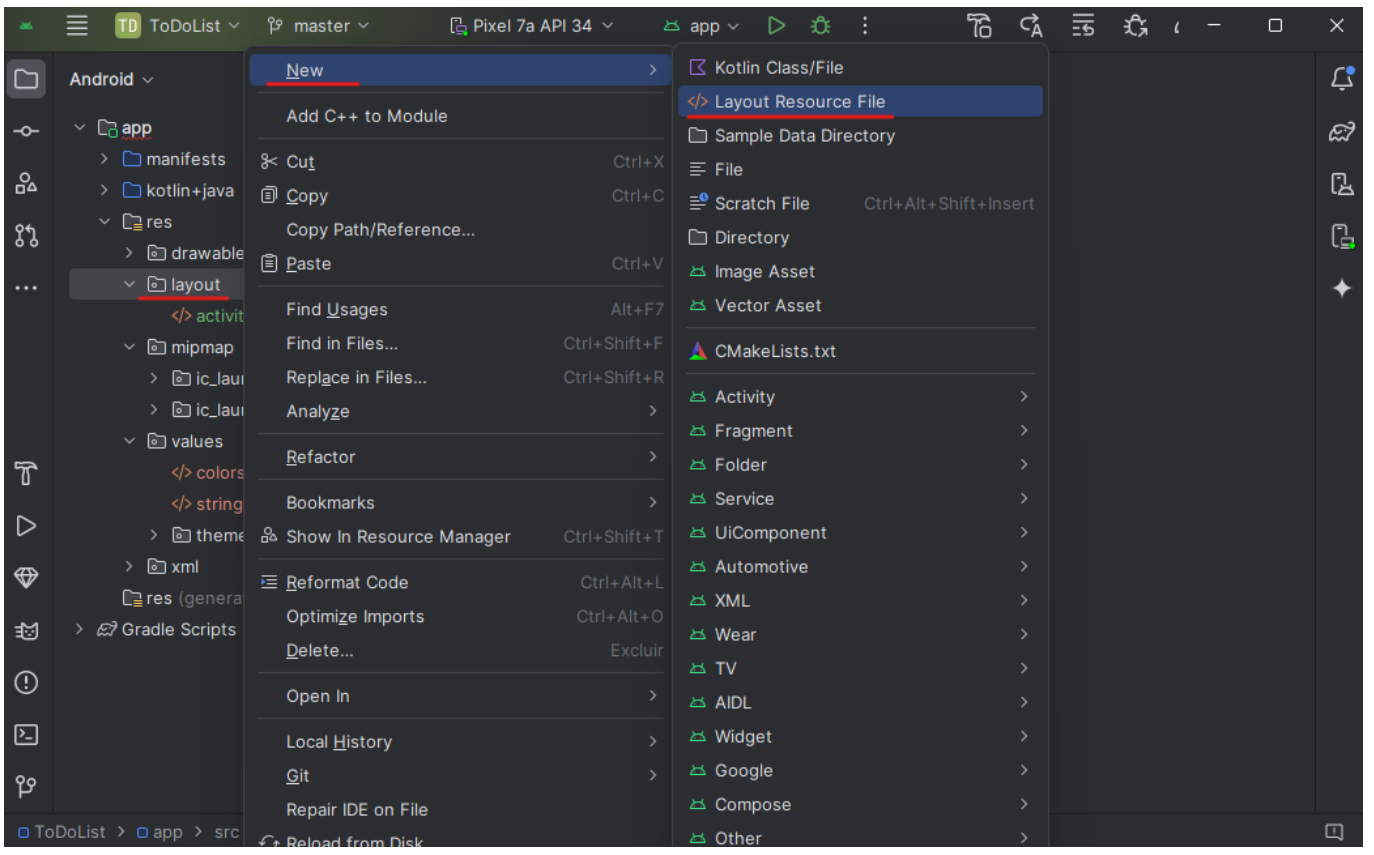

Figura 6: Interface do Android Studio, exibindo o projeto ToDoList e a hierarquia de pastas dele. Uma lista de opções é exibida, decorrente de um clique com o botão direito na pasta “layout”. A pasta “layout” e as opções “New” e “Layout Resource File” estão sublinhadas em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO A INTERFACE DO USUÁRIO

4. A tela **New Resource File** abrirá. Dê um nome para o arquivo (por exemplo: item_list) e clique em **OK**.

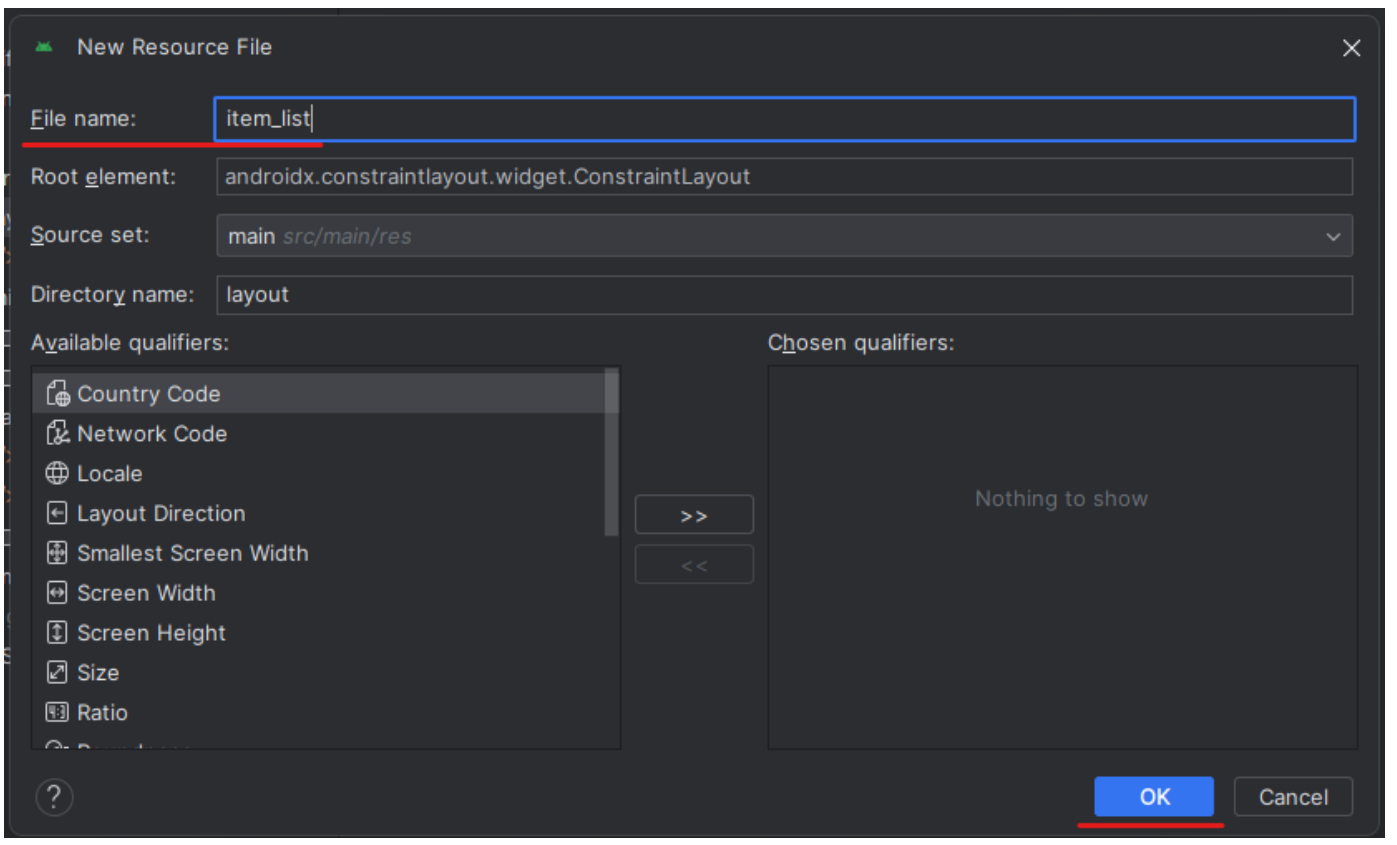

Figura 7: Interface de New Resource File do Android Studio, com o campo “File name” e o botão “OK” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO A INTERFACE DO USUÁRIO

5. Abra o novo arquivo **item_list.xml**, apague todo o conteúdo pré-existente nele e adicione o código abaixo. Não se apegue a tentar compreender cada linha desse código agora.

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

    <TextView
        android:id="@+id/textViewTask"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        tools:text="Sample Task" />

    <Button
        android:id="@+id/buttonRemove"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Remover"
        android:textSize="18sp"
        android:background="@color/white"
        android:layout_marginStart="8dp"
        android:padding="0dp" />
</LinearLayout>
```

Figura 8: Código em formato xml, com a implementação do layout item_list.xml do app ToDoList. Os componentes usados são: LinearLayout, TextView e Button. Fonte: Captura de tela do código, 2024.

Após seguir os passos anteriores, o layout do aplicativo estará pronto! Vamos adicionar funcionalidades para ele funcionar?

04

CRIANDO A LÓGICA DO APLICATIVO

CRIANDO A LÓGICA DO APLICATIVO

ADICIONANDO FUNCIONALIDADES

Lógica? Funcionalidades? O que é tudo isso?

Ambas palavras descrevem a mesma coisa. Basicamente, vamos escrever instruções para que o aplicativo saiba o que, como e quando fazer. Por exemplo, o que o aplicativo deve fazer quando você clicar em um botão? Como fazer com que ele saiba se é pra apagar ou incluir uma tarefa?

CRIANDO A LÓGICA DO APLICATIVO

Passos para a lógica de Adição de Tarefas:

1. No seu projeto, abra as pastas na seguinte ordem: **app**, **kotlin+java** e **com.example.todolist** (pode estar diferente se tiver colocado outro nome no seu projeto). Depois abra o arquivo **MainActivity.kt**. Nele desenvolveremos toda a lógica do aplicativo.

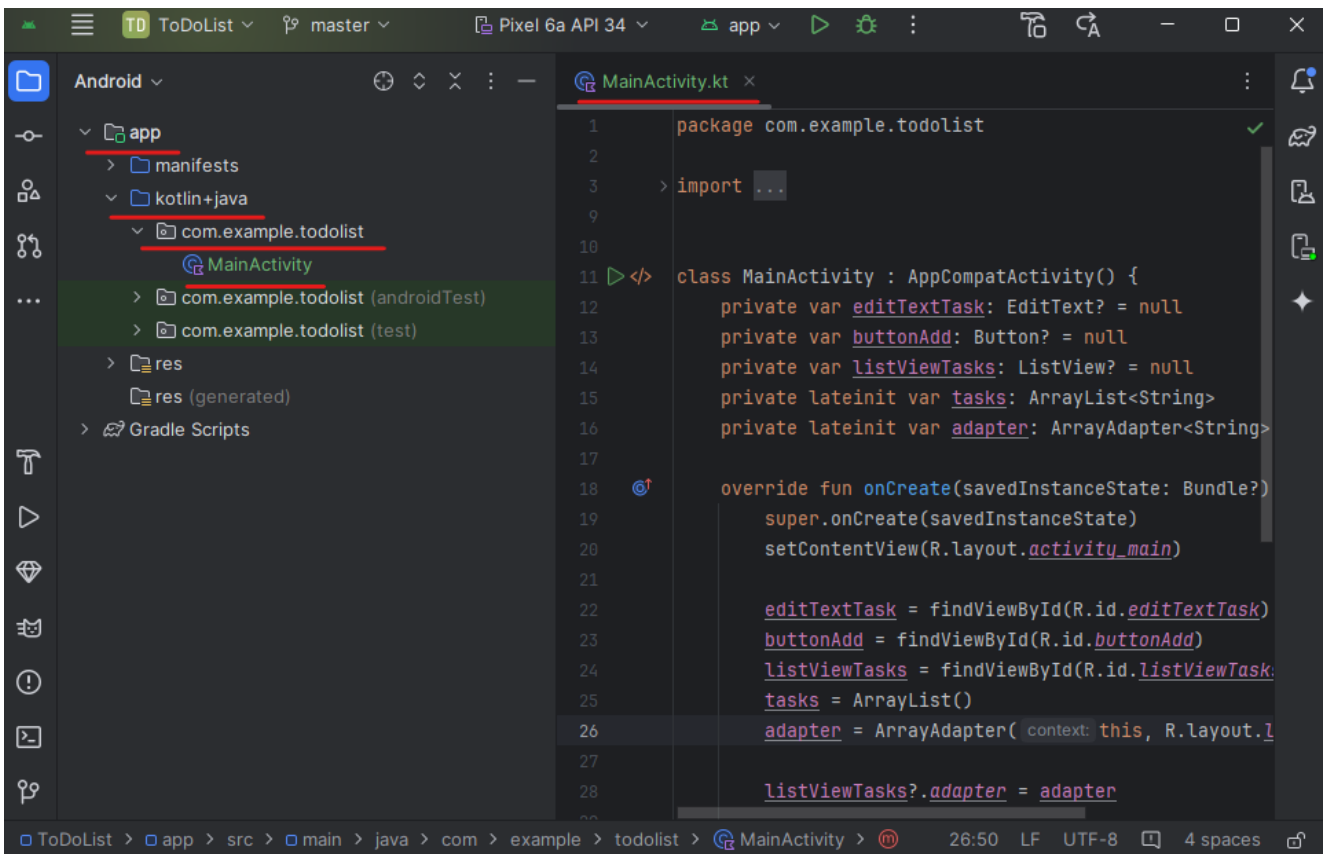

Figura 9: Interface do Android Studio, exibindo o projeto ToDoList e a hierarquia de pastas dele. As pastas “app”, “Kotlin+java”, “com.example.todolist” e o arquivo “MainActivity” estão abertos e sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

CRIANDO A LÓGICA DO APLICATIVO

2. Apague todo o conteúdo pré-existente no arquivo e adicione o código abaixo. Certifique-se que a linha referente ao **package**, contenha o mesmo nome que colocou em seu projeto.

```kotlin
package com.example.todolist

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView


class MainActivity : AppCompatActivity() {

    private lateinit var taskAdapter: TaskAdapter
    private val tasks = mutableListOf<String>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Obtendo referências para os elementos do layout
        val recyclerView: RecyclerView = findViewById(R.id.recyclerViewTasks)
        val editTextTask: EditText = findViewById(R.id.editTextTask)
        val buttonAdd: Button = findViewById(R.id.buttonAdd)

        // Inicializando o adapter com a lista de tarefas
        taskAdapter = TaskAdapter(tasks)
        recyclerView.adapter = taskAdapter
        recyclerView.layoutManager = LinearLayoutManager(this)

        // Configurando o clique do botão de adicionar tarefa
        buttonAdd.setOnClickListener {
            val task = editTextTask.text.toString()
            if (task.isNotEmpty()) {
                tasks.add(task)  // Adicionando a tarefa à lista
                // Notificando o adapter que um item foi inserido
                taskAdapter.notifyItemInserted(tasks.size - 1)
                editTextTask.text.clear()  // Limpando o campo de entrada de texto
            }
        }
    }
}
```

Figura 10: Código em formato kotlin, com a implementação da classe MainActivity do app ToDoList. Fonte: Captura de tela do código, 2024.

CRIANDO A LÓGICA DO APLICATIVO

Passo para a lógica de Exclusão de Tarefas:

1. Ainda no **MainActivity.kt**, no final do arquivo e antes da última chave (chave: }), adicione o código abaixo. Certifique-se de importar as bibliotecas que o Android Studio requisitar. Não se preocupe em tentar compreender cada linha agora.

```
    // Classe interna TaskAdapter para gerenciar a RecyclerView (Lista de Tarefas)
    inner class TaskAdapter(private val tasks: MutableList<String>) :
RecyclerView.Adapter<TaskAdapter.TaskViewHolder>() {

        // ViewHolder que contém as referências para os elementos de cada item da lista
        inner class TaskViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
            val taskTextView: TextView = itemView.findViewById(R.id.textViewTask)
            val removeButton: Button = itemView.findViewById(R.id.buttonRemove)
        }

        // Inflando o layout do item da lista e criando um ViewHolder
        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder {
            val view = LayoutInflater.from(parent.context).inflate(R.layout.item_list, parent, false)
            return TaskViewHolder(view)
        }

        // Vinculando os dados do item à posição específica do RecyclerView
        override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {
            val task = tasks[position]
            holder.taskTextView.text = task

            // Configurando o clique do botão de remoção
            holder.removeButton.setOnClickListener {
                tasks.removeAt(position)  // Removendo a tarefa da lista
                notifyItemRemoved(position)  // Notificando o adapter que um item foi removido
                notifyItemRangeChanged(position, tasks.size)  // Atualizando a faixa de itens afetados
            }
        }

        // Retornando o número total de itens na lista de tarefas
        override fun getItemCount() = tasks.size
    }
```

Figura 11: Código em formato kotlin, com a implementação da classe TaskAdapter, necessária para uso na MainActivity. Fonte: Captura de tela do código, 2024.

Agora o aplicativo está funcional! Vamos testá-lo?

05

TESTANDO O APLICATIVO

TESTANDO O APLICATIVO

Chegou um momento muito divertido e importante:
Ver o aplicativo que você criou, funcionando!!!

Para você rodar um aplicativo Android, é preciso um aparelho que use sistema operacional Android.
Neste ambiente de desenvolvimento, você pode testar seu aplicativo de duas formas: 1) em um dispositivo Android físico ou 2) em um dispositivo Android virtual.

Caso tenha curiosidade em ver o seu aplicativo rodando no seu próprio celular Android, você pode seguir aqui: https://developer.android.com/studio/run/device

Pensando que nem todes podem ter um aparelho em mãos, o passo a passo a seguir será com base num dispositivo Android virtual.

Vamos lá!

TESTANDO O APLICATIVO

Passos para Testar o Aplicativo:

1. No Android Studio, vá ao menu lateral direito, e clique no ícone **Device Manager**.

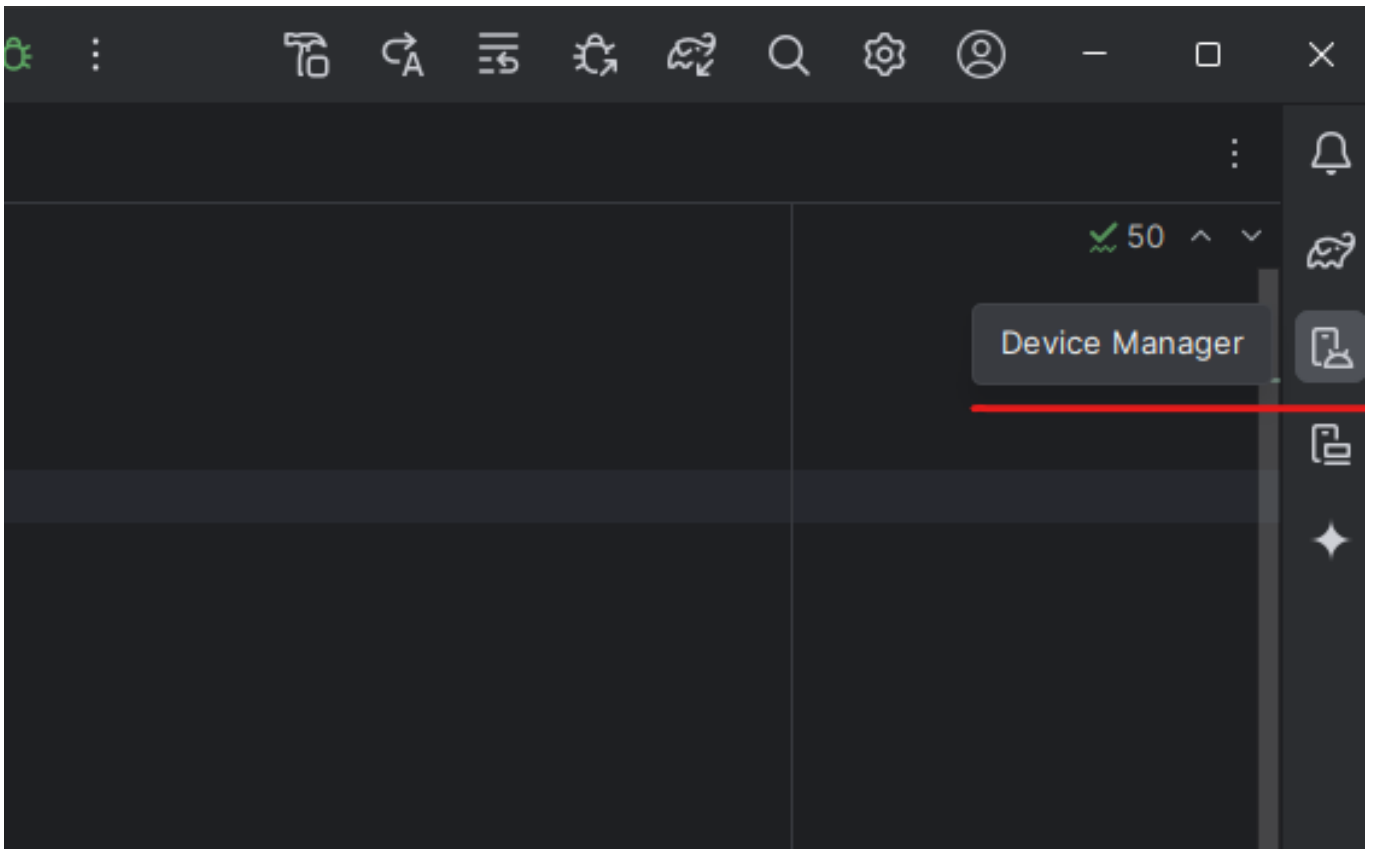

Figura 12: Parte da interface do Android Studio, exibindo o menu lateral direito, com o ícone “Device Manager”, este representado por um aparelho celular em pé atrás e a cabeça do robô mascote do Android na base à frente dele, sublinhado em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

2. Clique no ícone **Add a new Device**.

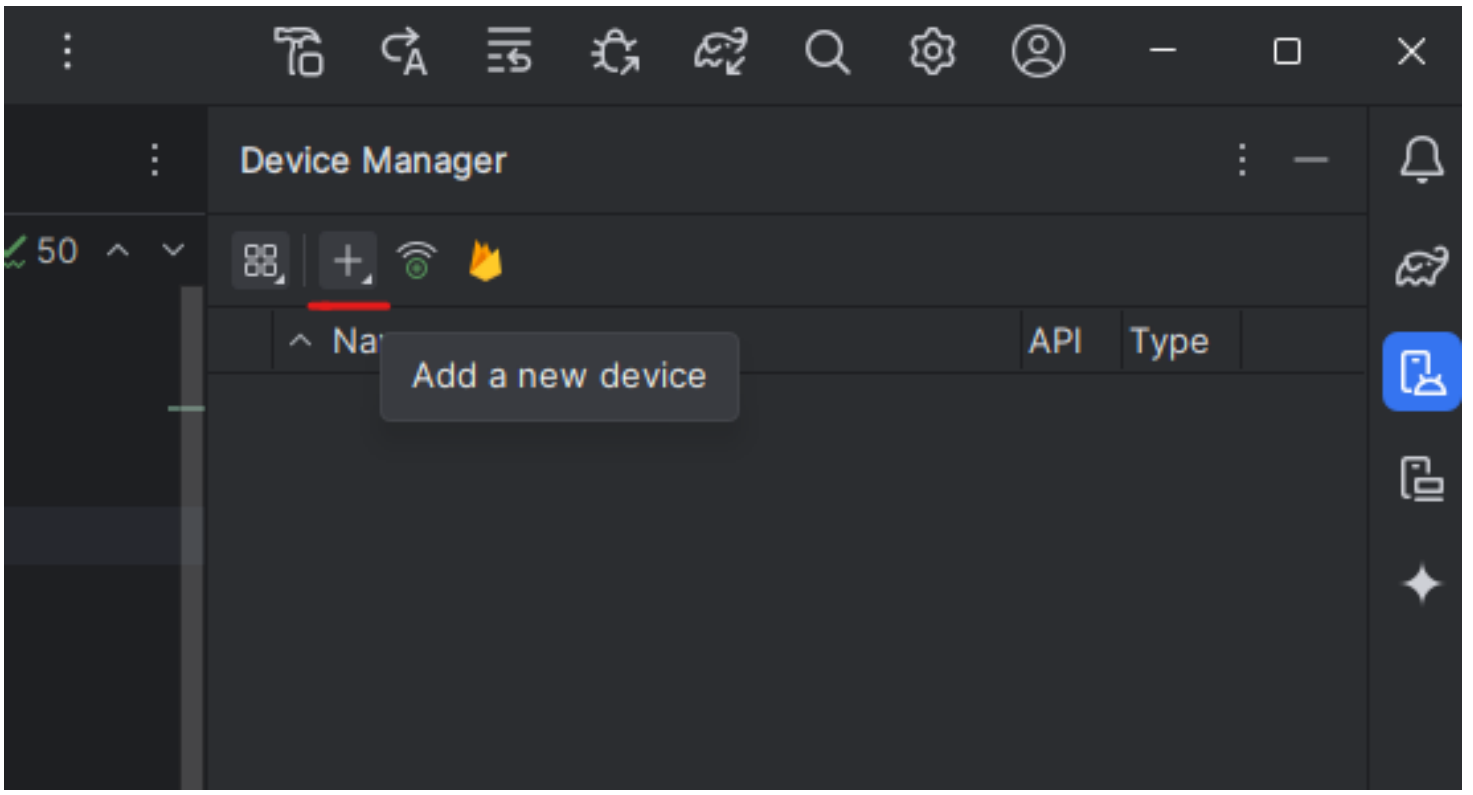

Figura 13: Parte da interface do Android Studio, exibindo a área de Device Manager, com o ícone “Add a new device”, este representado pelo símbolo de soma, sublinhado em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

3. Clique na opção **Create Virtual Device**.

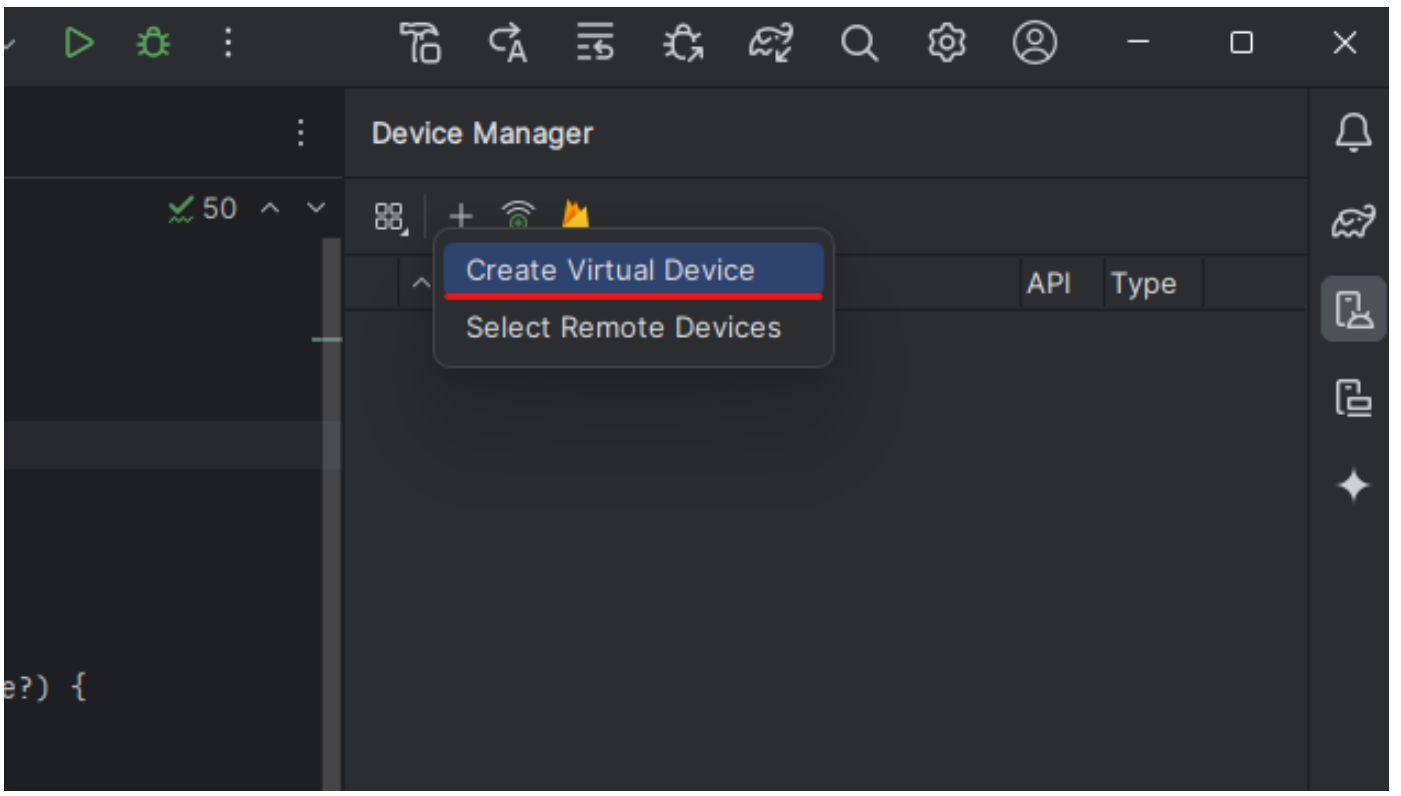

Figura 14: Parte da interface do Android Studio, exibindo a área de Device Manager, e as opções “Create Virtual Device” e “Select Remote Devices”, após clique no ícone “Add a new device”. A opção “Create Virtual Device” está sublinhada em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

4. A tela **Virtual Device Configuration** abrirá. Escolha o device que preferir, desde que esteja na categoria **Phone**. Quando escolher, clique em **Next**.

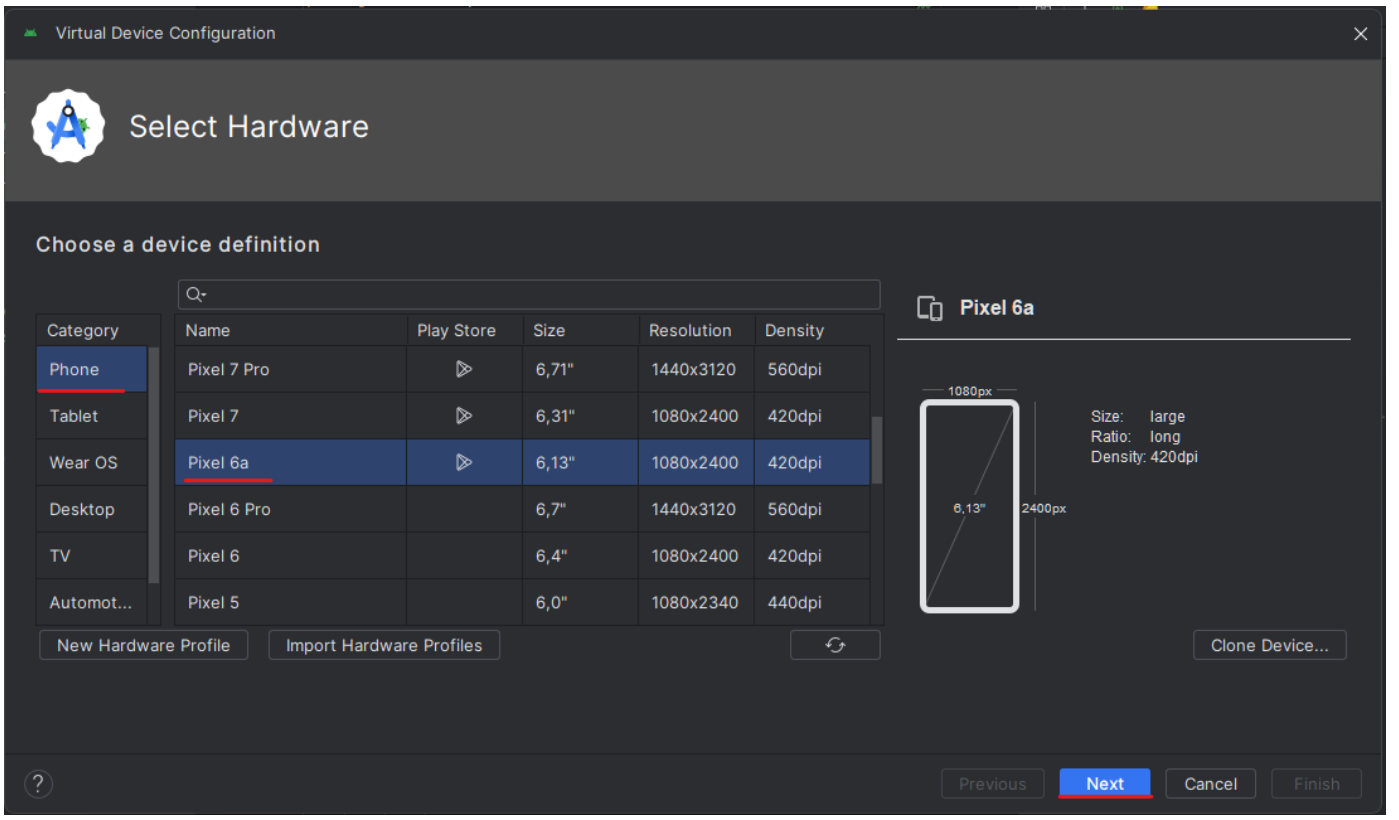

Figura 15: Interface de Virtual Device Configuration do Android Studio, com a categoria “Phone”, o device “Pixel 6a” e o botão “Next” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

5. Escolha a versão de Android para instalar no device (quanto mais recente, melhor), e clique em **Next**.

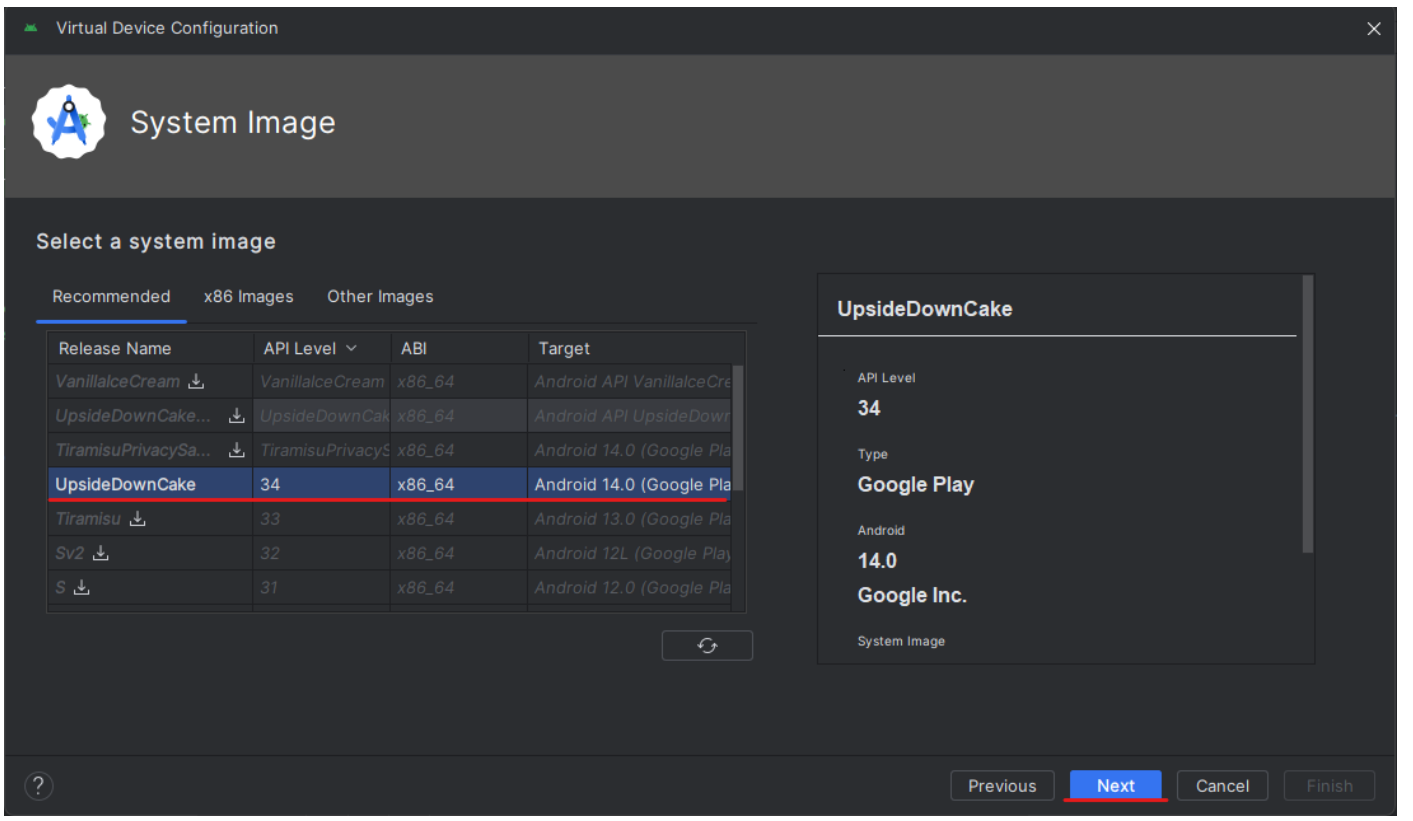

Figura 16: Interface de Virtual Device Configuration do Android Studio, com a versão de Android “UpsideDownCake” e o botão “Next” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

6. Na etapa **Verify Configuration** é exibido um resumo das configurações do device, incluindo o nome dele. Clique em **Finish**.

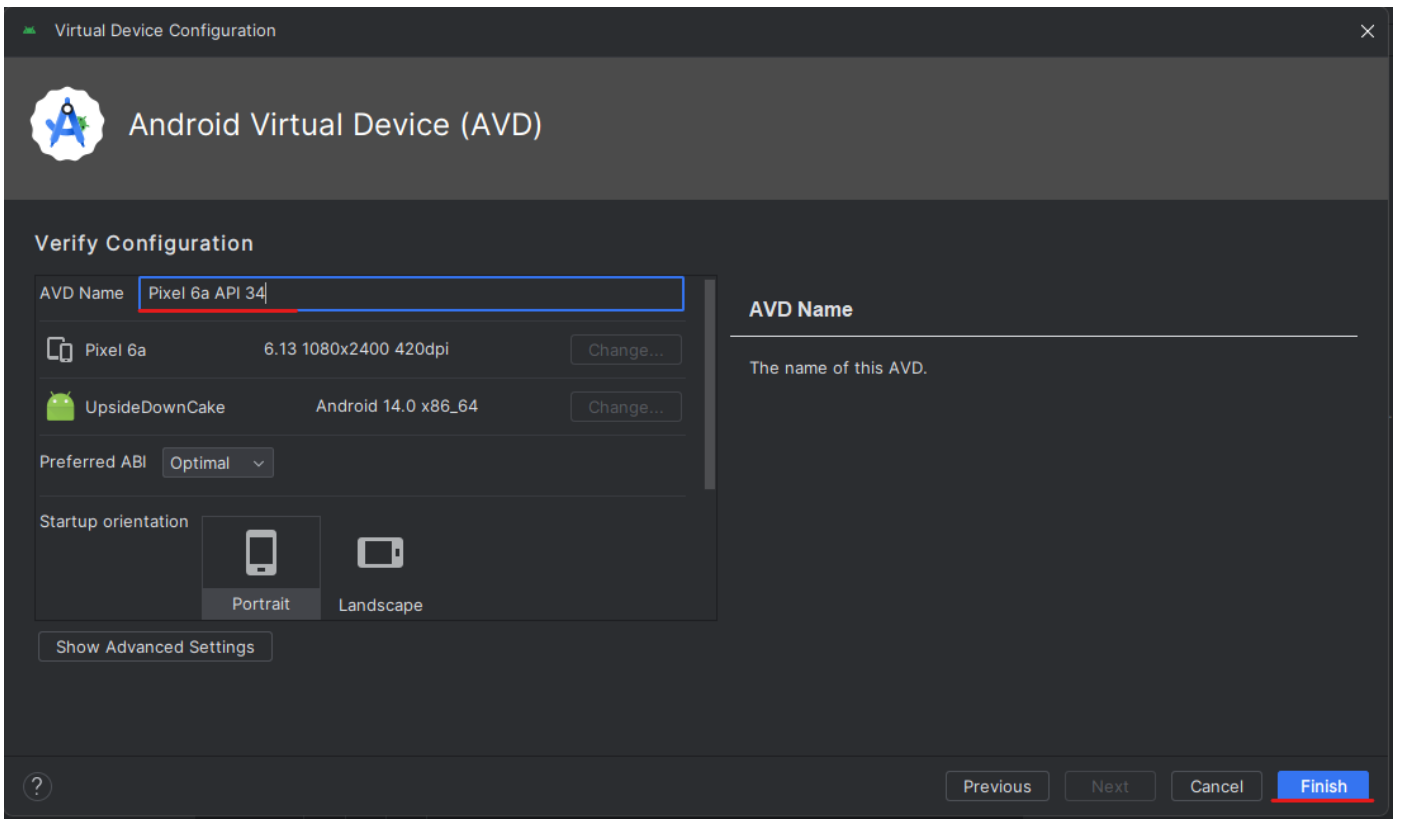

Figura 17: Interface de Virtual Device Configuration do Android Studio, com o campo “AVD Name” e o botão “Finish” sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

7. De volta à tela principal do Android Studio, e agora é possível ver o device virtual criado no **Device Manager**. No menu superior, clique no ícone de **Run ‘app’**.

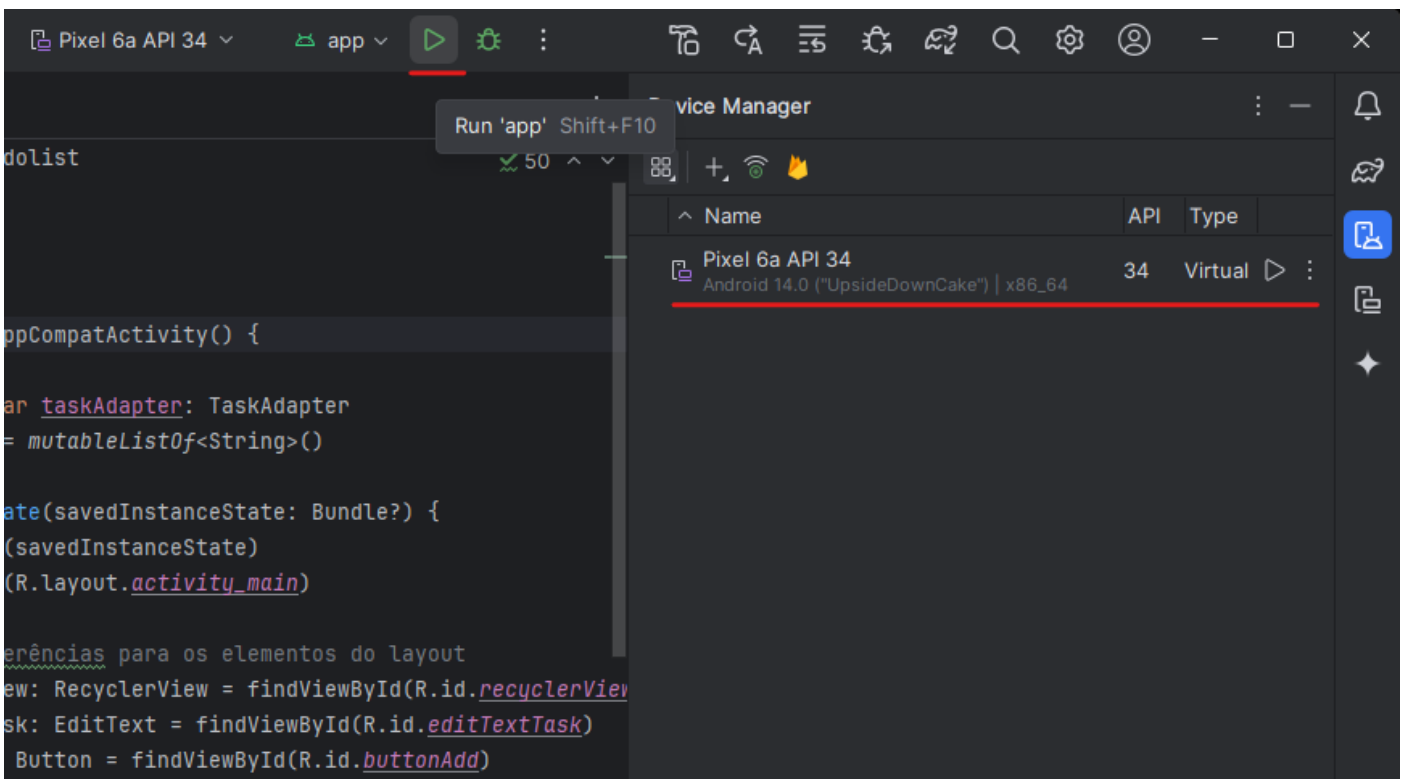

Figura 18: Parte da interface do Android Studio, mostrando o device virtual criado na área de Device Manager. O nome do device e o ícone “Run ‘app’”, este representado por uma seta com a ponta pra direita (comumente usado em botões de play), estão sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

TESTANDO O APLICATIVO

8. Quando tudo estiver pronto da parte do Android Studio, o device virtual aparecerá rodando o aplicativo! Caso não apareça, vá ao mesmo menu vertical na lateral direita, usado em passos anteriores, e clique no ícone **Running Devices**.

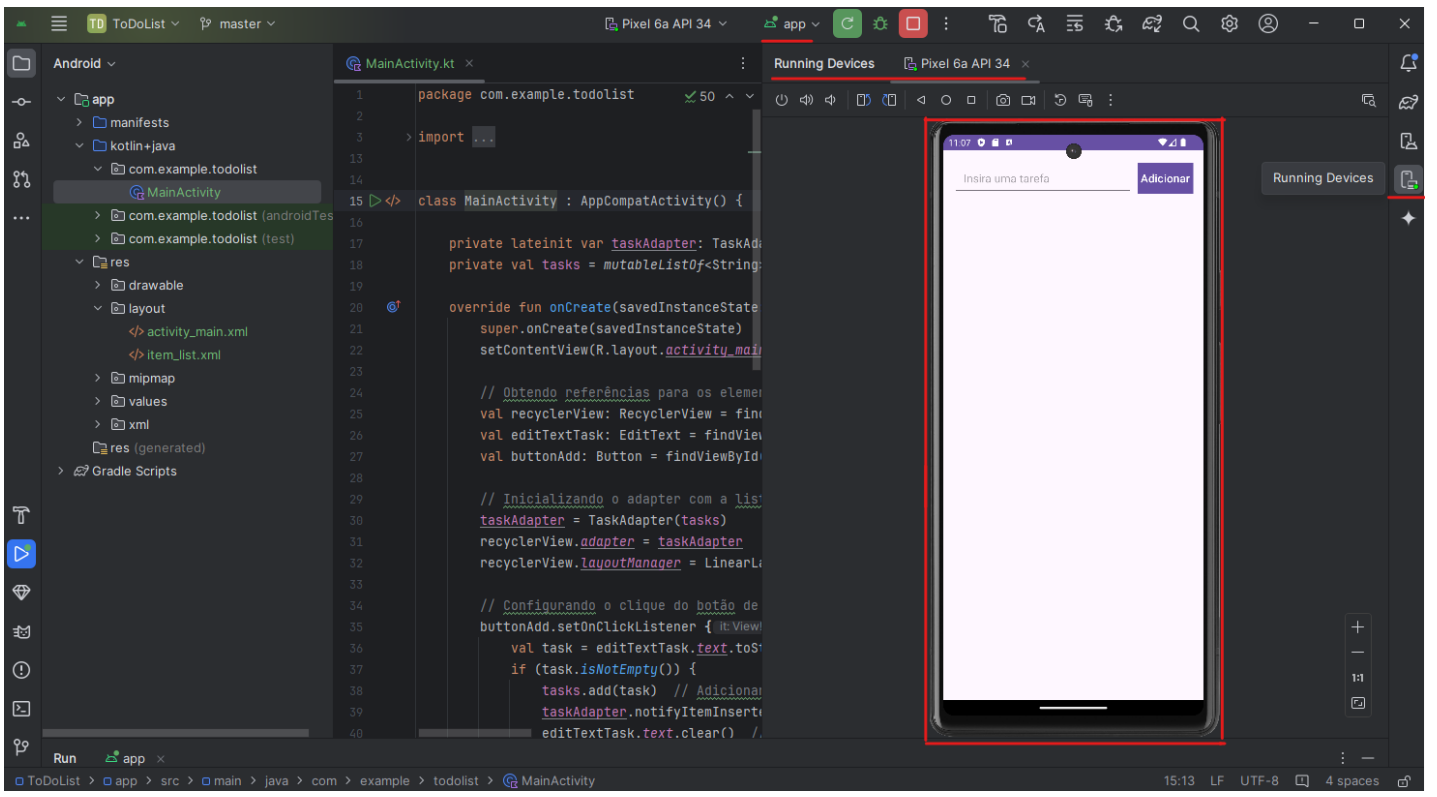

Figura 19: Interface do Android Studio, com destaque à área de Running Devices. O botão “Run ‘app’”, o título “Running Devices”, o nome do device virtual e a tela do device, que exibe a tela do app ToDoList, estão sublinhados em vermelho. Fonte: Captura de tela do Android Studio, 2024.

Muito legal né? Agora que o aplicativo está rodando, é possível testar as funcionalidades que desenvolvemos!

TESTANDO O APLICATIVO

9. Agora é a diversão! Adicione e remova tarefas, veja como sua lista se comporta na tela. Identifique pontos de melhoria, seja corrigindo ou adicionando novas funcionalidades. Isso pode ser um grande motivador para seguir nos estudos de Android!

Figura 20: Interface do app ToDoList, exibindo o campo para inserir uma tarefa, e o botão “Adicionar”. Não há nenhuma tarefa na lista. Fonte: Captura de tela do emulador do Android Studio, 2024.

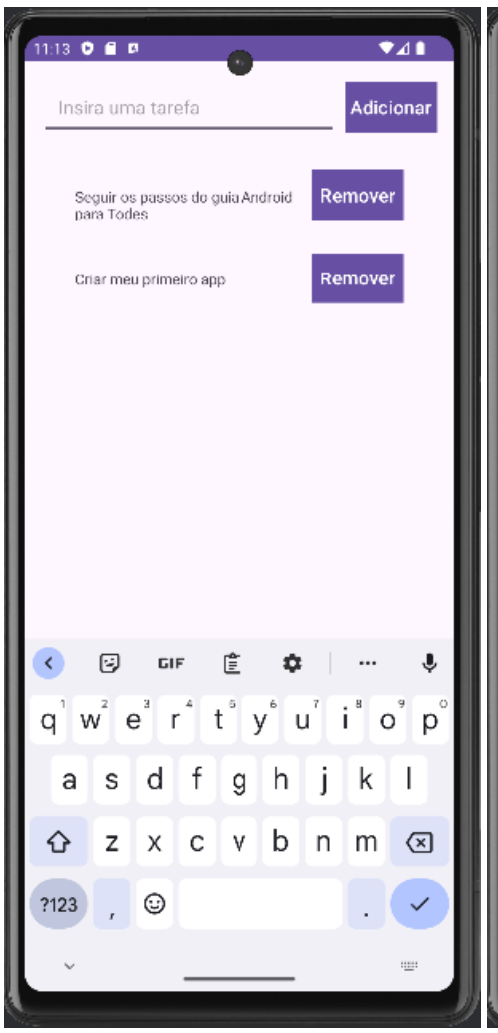

Figura 21: Interface do app ToDoList, exibindo duas tarefas na lista, e cada uma atrelada a um botão “Remover”. Fonte: Captura de tela do emulador do Android Studio, 2024.

Figura 22: Interface do app ToDoList, exibindo uma tarefa na lista, atrelada a um botão “Remover”, e dando a entender que a outra tarefa foi removida. Fonte: Captura de tela do emulador do Android Studio, 2024.

AGRADECIMENTOS

Obrigade por ler este guia para a criação de seu primeiro aplicativo! Espero que tenha conseguido acompanhar e que te dê motivação para seguir nessa jornada do Desenvolvimento Android.

Figura 23: Ícone verde do mascote do Android. Fonte: Google, 2024.

Esse ebook foi construído com auxílio de IAs. Os passos para sua criação encontram-se no meu GitHub.
Houve intercepção humana para melhoria e correção do conteúdo gerado, além da criação de códigos, prints dos passos e diagramação.

Esse conteúdo foi gerado para fins didáticos, então a validação humana não foi muito rigorosa e pode conter erros despercebidos.

Para sugestões/dúvidas, contate-me nas redes:
@github @linkedin