Guião de Formação: Desenvolvimento de Aplicação Android com Kotlin

Introdução

Neste guião, vamos aprender a criar uma aplicação Android que funciona como um velocímetro. O objetivo não é apenas replicar o código, mas compreender como as diferentes partes se encaixam:

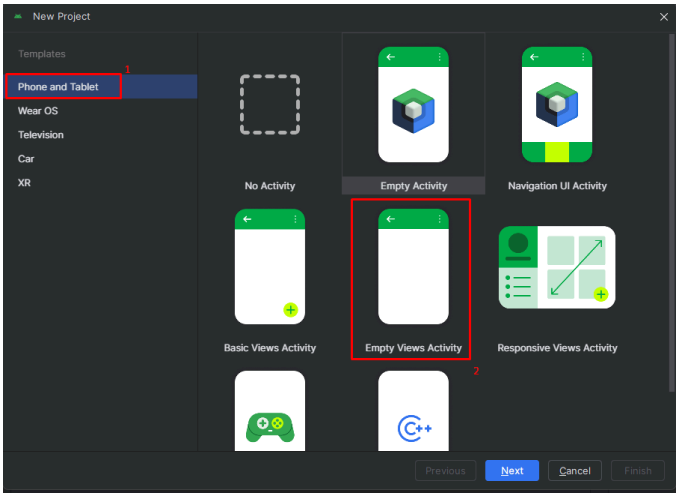
- **A Peça Pronta (SpeedometerView)**: O desenho do velocímetro (já fornecida)
- **A Montagem (XML):** A construção do ecrã com botões e elementos visuais
- **A Lógica (Kotlin):** O código que faz tudo funcionar

Fase 1: Preparação do Projeto

Passo 1: Escolher o Molde

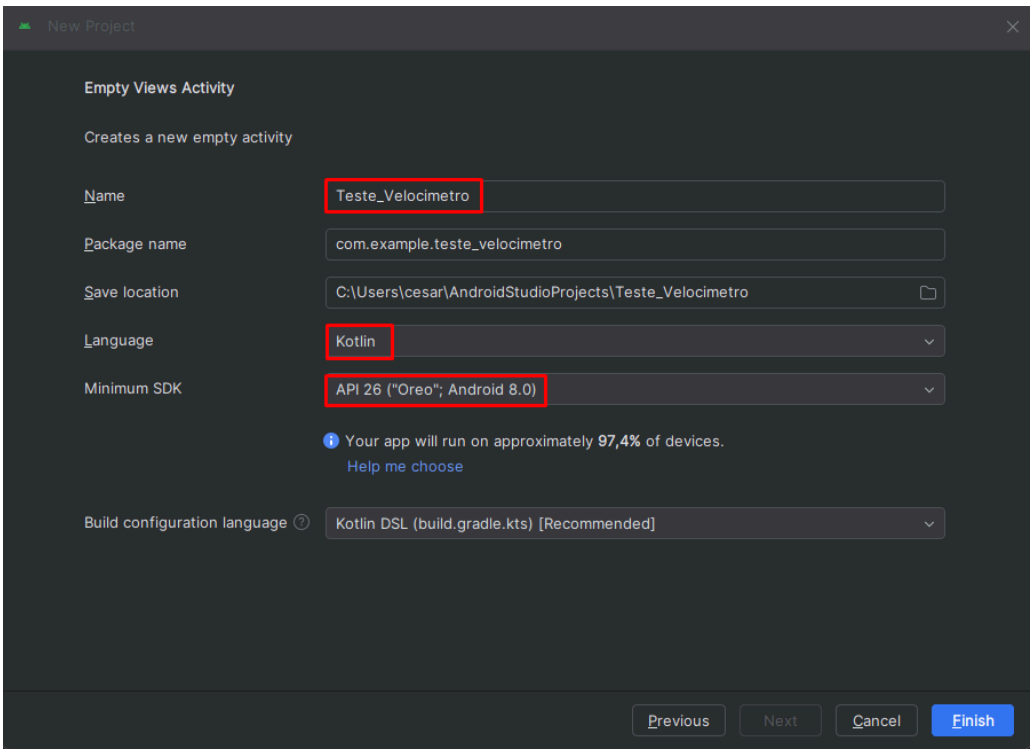
No ecrã "New Project", vamos garantir que escolhemos o molde certo.

1. Certifica-te que estás no separador **Phone and Tablet** (no menu da esquerda).
2. Procura a opção **Empty Views Activity**.



Passo 2: Nomear Projeto

- **Name:** `Teste_Velocimetro`
- **Package name:** `com.example.teste_velocimetro`
- **Language:** `Kotlin`
- **Minimum SDK:** `API 26 (Android 8.0)`



Aguarda que o projeto seja carregado.

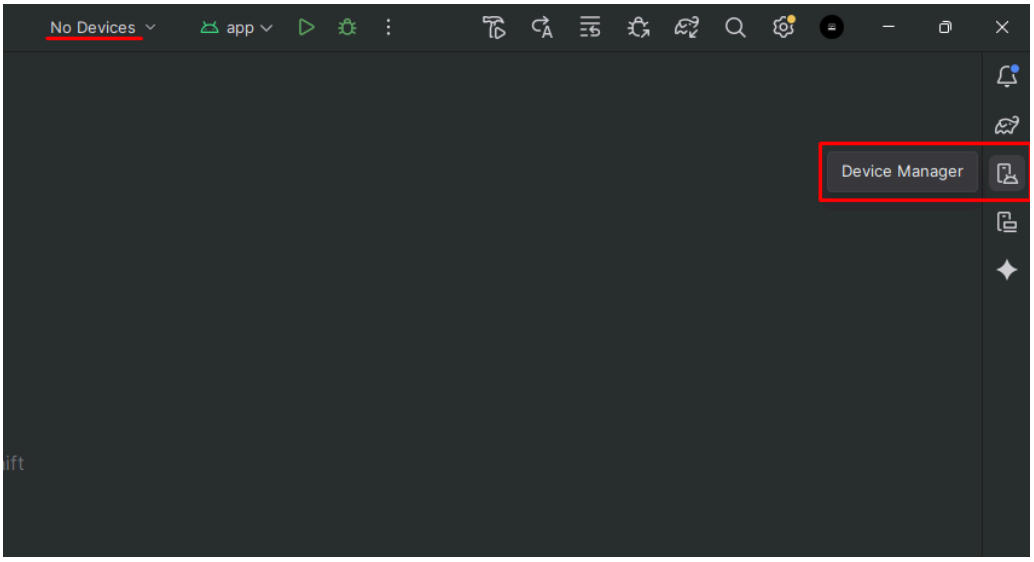
Assim que o projeto terminar de ser carregado, o botão Play (triângulo verde) deve estar verde e clicável. O projeto foi criado com sucesso!

No lado direito do Android Studio, deves ver uma barra lateral com ícones pequenos. Procura um ícone que parece um **telemóvel** e clica nele. (Se passares o rato por cima diz "Device Manager").

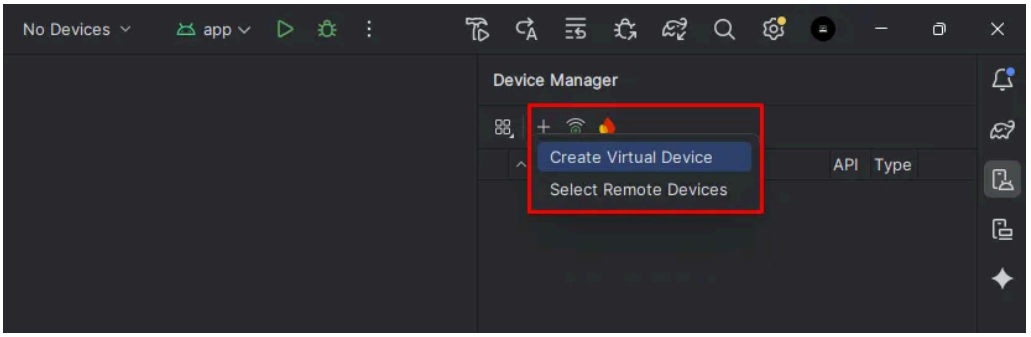
Passo 3: Criar um Dispositivo Virtual (Emulador)

Para testar a app, precisamos de um smartphone emulado.

1. No lado direito do Android Studio, procura um ícone que parece um telemóvel (diz
   "Device Manager" ao passar o rato).

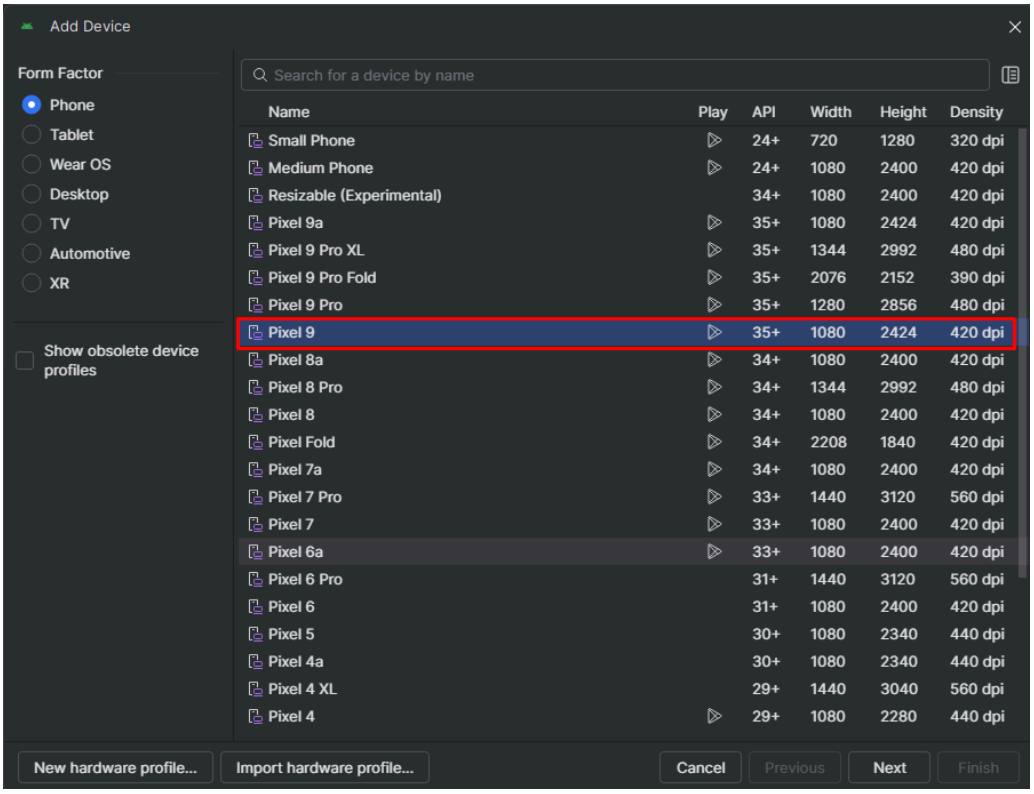


2. Clica no botão + (Create Virtual Device).

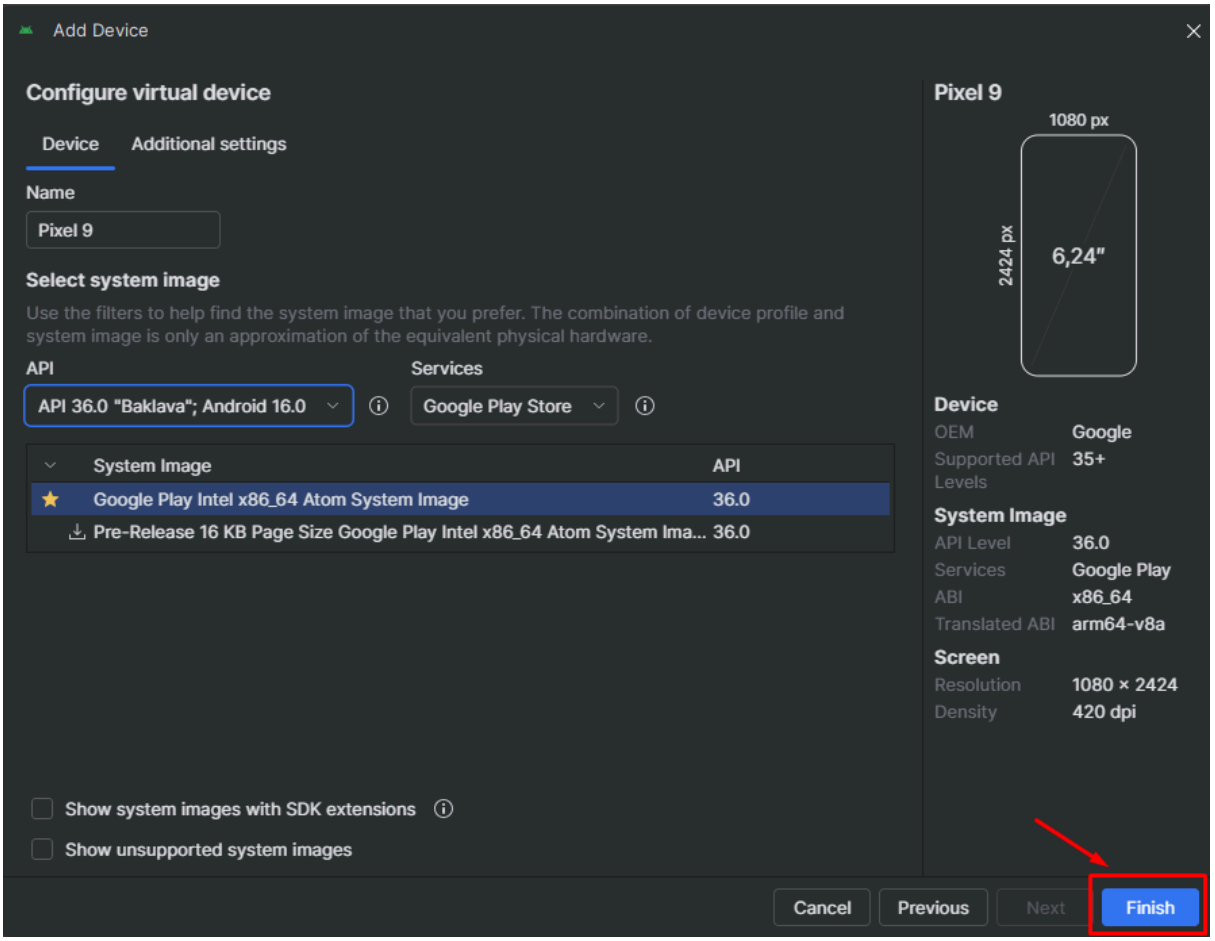


Escolher o Hardware:

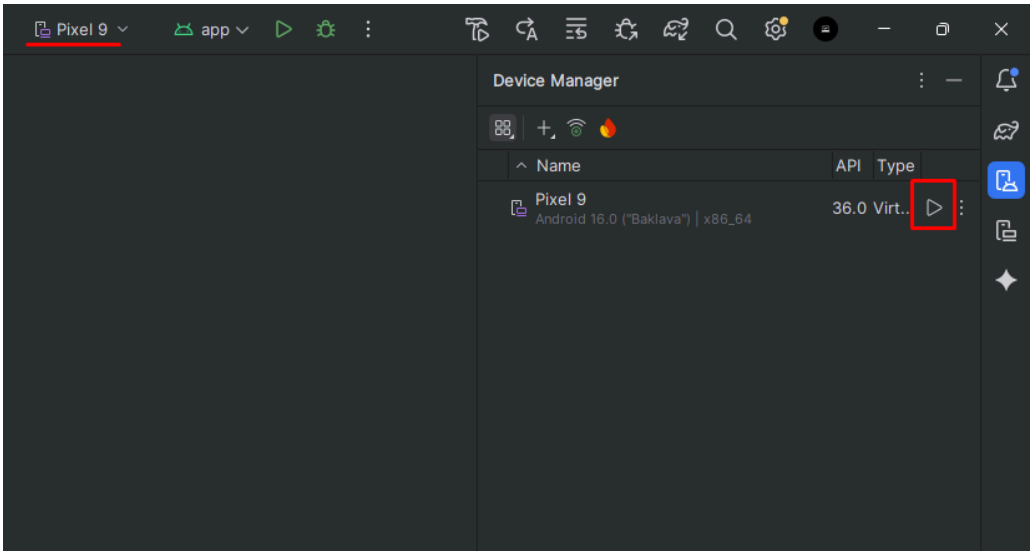
- Na lista que aparece ("Select Hardware"), escolhe **Phone** no lado esquerdo.
- No meio, escolhe **Pixel 9**
- Clica em **Next**.



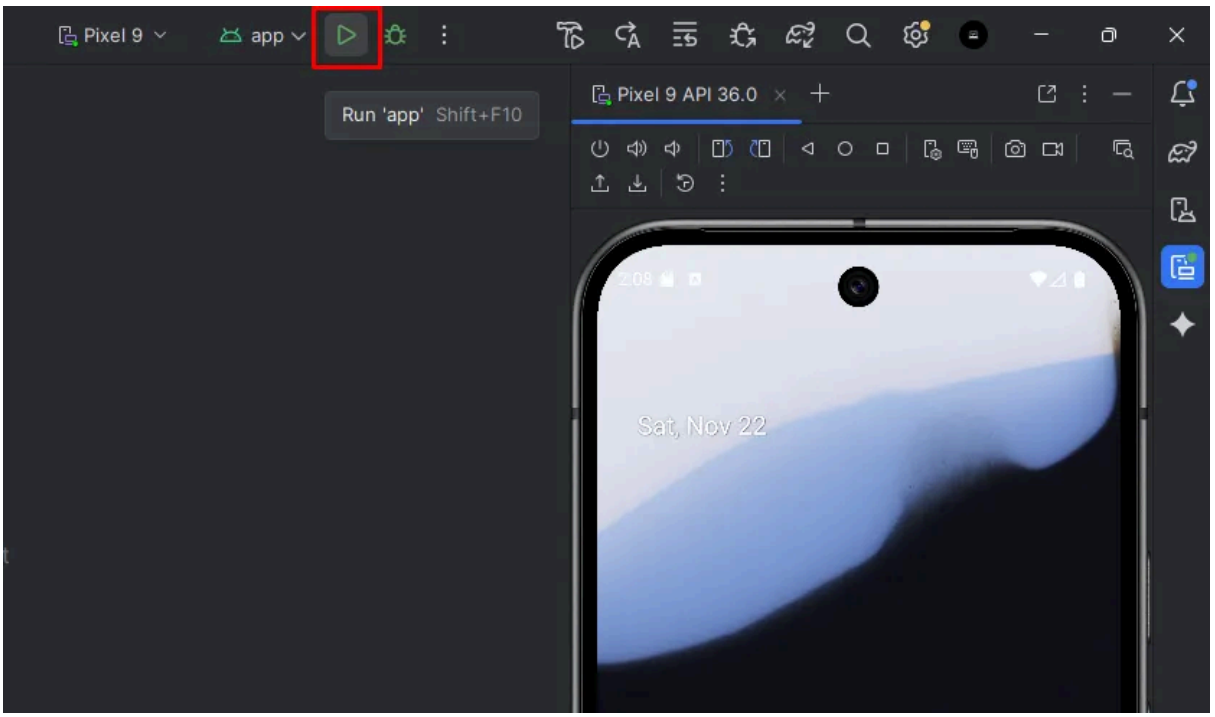
- Clica em **Finish**



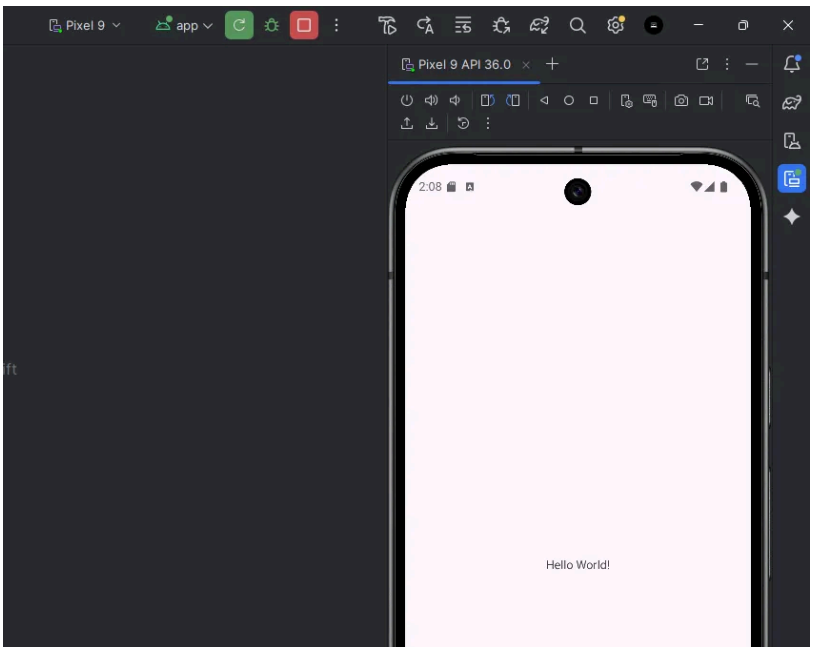
Ao clicar no botão “Play” marcado, temos acesso ao nosso telemóvel emulado



Agora vamos dar run ao nosso projeto no telemóvel emulado, clicando no **botão play verde,** que se encontra marcado na imagem.



Output esperado:

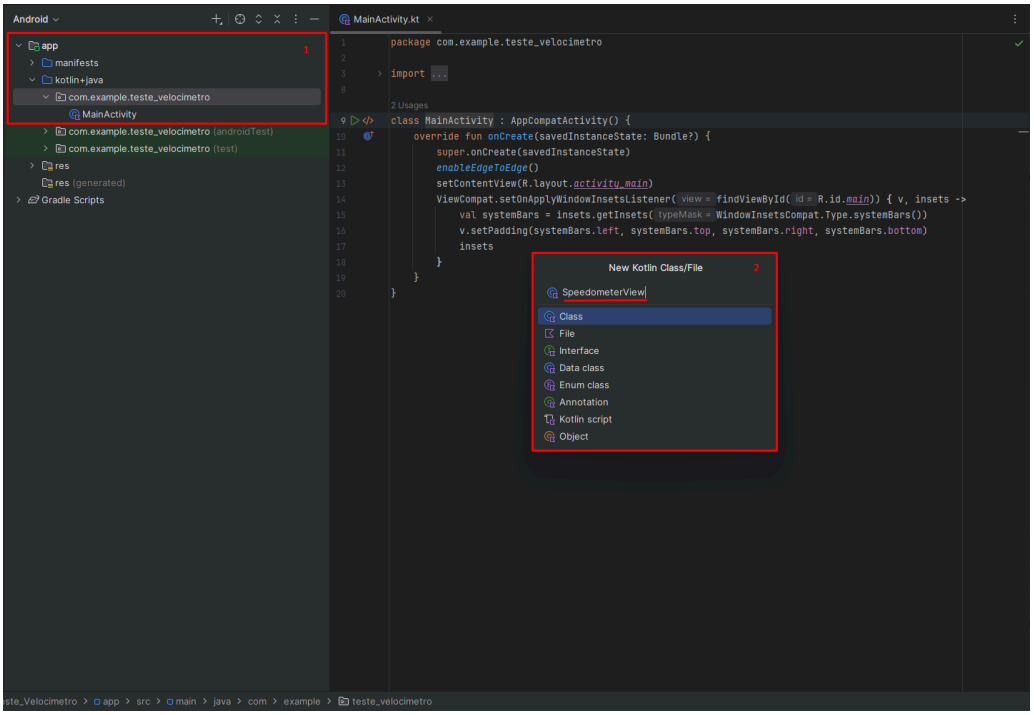


Fase 2: Criar o Velocímetro (SpeedometerView)

Agora vamos adicionar a peça pronta do velocímetro. Este é o componente visual que
mostra o velocímetro graficamente.

Passo 1: Criar o Ficheiro

1. No lado esquerdo do Android Studio, na aba **Project**, procura a pasta `app` > `kotlin + java` > `com.example.teste_velocimetro` (onde já está o `MainActivity`).
2. Clica com o botão direito nessa pasta (`com.example...`).
3. Vai a **New** > **Kotlin Class/File**.
4. Escreve o nome: `SpeedometerView`
5. Dá **Enter**



Passo 2: Colar o Código

O ficheiro vai abrir vazio. Apaga tudo e cola o seguinte código:

```
package com.example.teste_velocimetro

import android.content.Context
import android.graphics.*
import android.util.AttributeSet
import android.view.View
import kotlin.math.cos
import kotlin.math.min
import kotlin.math.sin

class SpeedometerView @JvmOverloads constructor(
    context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0
) : View(context, attrs, defStyleAttr) {

    // Configuração da velocidade máxima e atual
    var maxSpeed: Float = 180f
        set(value) { field = value; invalidate() }
    var speed: Float = 0f
```

```
    set(value) { field = value.coerceIn(0f, maxSpeed); invalidate() }

// Tintas para desenhar (Estilo)
private val basePaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
    color = Color.LTGRAY
    style = Paint.Style.STROKE
    strokeWidth = 24f
    strokeCap = Paint.Cap.ROUND
}

private val progressPaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
    color = Color.parseColor("#2962FF") // Azul
    style = Paint.Style.STROKE
    strokeWidth = 24f
    strokeCap = Paint.Cap.ROUND
}

private val needlePaint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
    color = Color.DKGRAY
    style = Paint.Style.STROKE
    strokeWidth = 6f
}

// O desenho propriamente dito
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    val w = width.toFloat()
    val h = height.toFloat()
    val radius = min(w, h) / 2f * 0.9f
    val cx = w / 2f
    val cy = h / 2f
    val rect = RectF(cx - radius, cy - radius, cx + radius, cy + radius)

    // 1. Desenhar o arco cinzento (fundo)
    canvas.drawArc(rect, 135f, 270f, false, basePaint)

    // 2. Desenhar o arco azul (progresso) baseado na velocidade
    val sweep = 270f * (speed / maxSpeed)
```

```
        canvas.drawArc(rect, 135f, sweep, false, progressPaint)

        // 3. Desenhar o ponteiro (Matemática para encontrar a ponta da agulh
a)
        val angleDeg = 135f + sweep
        val angleRad = Math.toRadians(angleDeg.toDouble())
        val needleLen = radius * 0.85f
        val nx = cx + (cos(angleRad) * needleLen).toFloat()
        val ny = cy + (sin(angleRad) * needleLen).toFloat()
        canvas.drawLine(cx, cy, nx, ny, needlePaint)
    }
}
```

Guarda o ficheiro (Ctrl+S).

Breve explicação do código:
maxSpeed e speed: Variáveis que guardam o valor máximo e atual da velocidade.
Paint: Define como desenhar (cores, estilos, espessura).
onDraw(): O método que desenha tudo no ecrã.

1. Desenha um arco cinzento (fundo)
2. Desenha um arco azul (que cresce com a velocidade)
3. Desenha um ponteiro (agulha) que roda

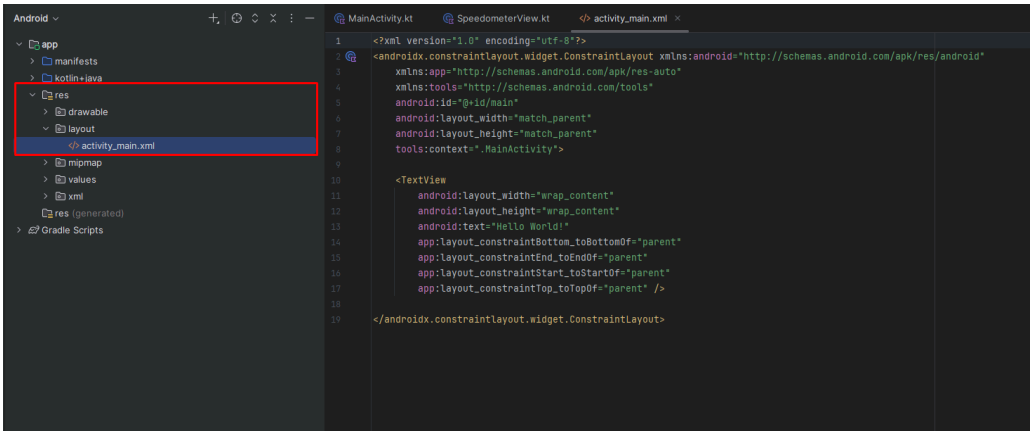
Fase 3: Montar o Ecrã (XML)

Agora vamos construir o ecrã da aplicação: velocímetro, texto da velocidade e botões.

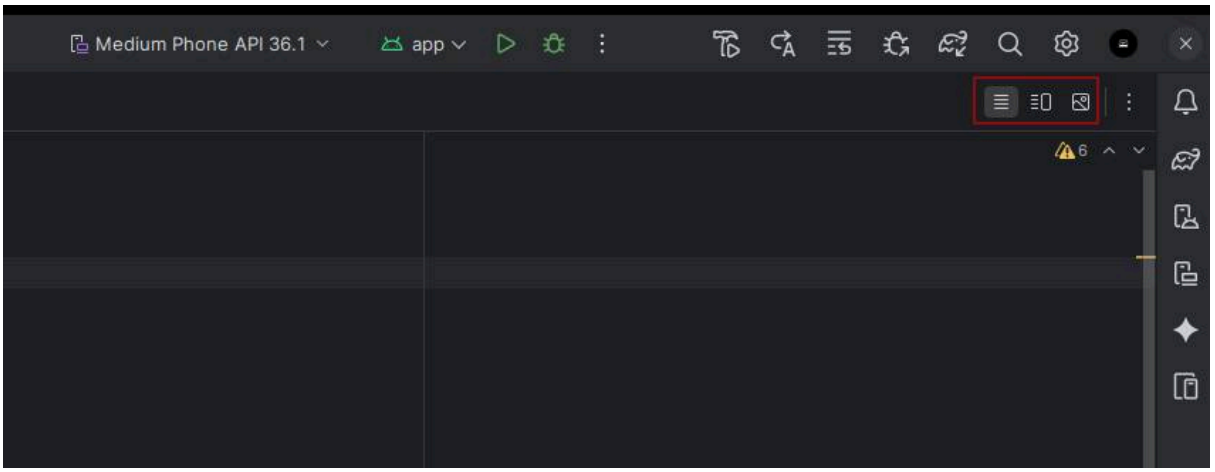
Passo 1: Abrir o ficheiro XML

Agora vamos construir o ecrã da aplicação: velocímetro, texto da velocidade e botões.

1. No lado esquerdo (Project), vai a: `app` > `res` > `layout` > `activity_main.xml`.



1. Quando o ficheiro abrir, olha para o canto superior direito do editor. Deves ver 3 botões: **Design**, **Split**, **Code**.
    - Clica em **Code** (para vermos só o código XML).



2. Apaga **tudo** o que lá estiver.
3. Cola este código:

```
<?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"
    android:padding="24dp">

    <TextView
```

```
    android:id="@+id/speedText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Velocidade: 0 km/h"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    />

<com.example.teste_velocimetro.SpeedometerView
    android:id="@+id/speedometerView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintTop_toBottomOf="@id/speedText"
    app:layout_constraintBottom_toTopOf="@id/btnAccel"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    />

<Button
    android:id="@+id/btnBrake"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Travar"
    android:backgroundTint="#D32F2F"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintWidth_percent="0.45"
    />

<Button
    android:id="@+id/btnAccel"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Acelerar"
    android:backgroundTint="#388E3C"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintWidth_percent="0.45"
        />

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

Breve Explicação

Elemento (ID)	Função
TextView (`speedText`)	Mostra o valor numérico da velocidade (ex: "0 km/h").
SpeedometerView (`speedometerView`)	O nosso velocímetro personalizado (o desenho dos arcos).
Button (`btnBrake`)	Botão **Travar** (Vermelho). Diminui a velocidade.
Button (`btnAccel`)	Botão **Acelerar** (Verde). Aumenta a velocidade.

Fase 4: Programar a Lógica (MainActivity)

Agora vamos programar o "cérebro" da app. É aqui que as coisas ganham vida!

Passo 1: Abrir a MainActivity

É aqui que vais gastar mais tempo a explicar aos teus alunos, porque é aqui que a "magia" acontece.

1. No lado esquerdo (Project), procura `MainActivity.kt`
2. Apaga todo o código.
3. Cola este código:

```
package com.example.teste_velocimetro

import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.widget.Button
import android.widget.TextView
```

```
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    // 1. Declarar as peças que vamos usar
    private lateinit var speedometerView: SpeedometerView
    private lateinit var speedText: TextView
    private lateinit var btnBrake: Button
    private lateinit var btnAccel: Button

    // Variáveis de estado (como está a mota agora?)
    private var speed = 0f
    private val maxSpeed = 180f

    // Coisas técnicas para a animação de desacelerar (Timer)
    private val handler = Handler(Looper.getMainLooper())
    private val decelRunnable = object : Runnable {
        override fun run() {
            if (speed > 0f) {
                speed = (speed - 0.5f).coerceAtLeast(0f)
                updateUI()
                handler.postDelayed(this, 50L) // Repetir daqui a 50 milisegundos
            }
        }
    }

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

        // 2. Ligar as variáveis ao XML (Encontrar as peças pelo ID)
        speedometerView = findViewById(R.id.speedometerView)
        speedText = findViewById(R.id.speedText)
        btnBrake = findViewById(R.id.btnBrake)
        btnAccel = findViewById(R.id.btnAccel)

        // Configurar o velocímetro
        speedometerView.maxSpeed = maxSpeed
```

```
        // 3. O que acontece quando clicamos em "Acelerar"?
        btnAccel.setOnClickListener {
            speed = (speed + 10f).coerceAtMost(maxSpeed) // Aumenta 10, ma
s não passa do máx
            updateUI()

            // Se começámos a acelerar, garantimos que a desaceleração está a
tiva
            handler.removeCallbacks(decelRunnable)
            handler.post(decelRunnable)
        }

        // 4. O que acontece quando clicamos em "Travar"?
        btnBrake.setOnClickListener {
            speed = (speed - 15f).coerceAtLeast(0f) // Diminui 15, mas não baix
a de 0
            updateUI()
        }
    }

    // Função auxiliar para atualizar o texto e o desenho
    private fun updateUI() {
        speedometerView.speed = speed
        speedText.text = "Velocidade: ${speed.toInt()} km/h"
    }
}
```

Fase 5: Testar a Aplicação

Executar a App

1. Clica no botão **Play** (triângulo verde).
2. A app deve abrir no emulador.

Testar Funcionalidades

- **Botão Acelerar (verde):** Clica várias vezes. A velocidade deve aumentar, o velocímetro deve evoluir e o texto deve atualizar.
- **Botão Travar (vermelho):** Clica para diminuir a velocidade rapidamente.
- **Desaceleração Automática:** Sem clicares em nada, a mota deve abrandar lentamente (cada 50 ms diminui 0.5 km/h).

Fase 6: Desafios de Extensão (Avaliação)

Desafio 1: O Limitador de Velocidade 🚦
Objetivo: A velocidade nunca pode passar dos 120 km/h. Se passar, o texto fica vermelho.
Dica: Usar estruturas if/else e setTextColor():

```
if (speed > 120) {
speedText.setTextColor(android.graphics.Color.RED)
} else {
speedText.setTextColor(android.graphics.Color.BLACK)
}
```

Onde adicionar: Dentro da função updateUI().

Desafio 2: A Bateria 🔋
Objetivo: Adicionar uma barra de bateria que desce quando acelera.
Passos:

1. No XML, adiciona um ProgressBar horizontal.
2. No Kotlin, cria var fuel = 100.
3. Quando acelera, diminui o combustível: fuel -= 1.
4. Se fuel <= 0, impede acelerar.

Desafio 3: O Indicador de Mudanças ⚙️
Objetivo: Adicionar seletor de mudanças (N, 1, 2, 3, 4, 5, 6).
Passos:

1. Cria dois botões: "+" e "-".
2. Cria uma variável: var gear = 0.
3. Buttons incrementam/decrementam dentro dos limites.
4. Mostra a mudança atual num TextView.

Desafio 4: O Conta-Km 🛣️
Objetivo: Mostrar quilómetros totais percorridos.
Passos:

1. Cria var distanciaTotal = 0f.
2. No decelRunnable (que corre a cada 50ms), incrementa: distanciaTotal += speed *
   0.001.
3. Mostra num novo TextView.

Desafio 5: O Modo Piscas ↔️
Objetivo: Dois botões que ligam/desligam setas animadas.
Passos:

1. Adiciona duas ImageView (setas) no XML com visibility="invisible".
2. Dois botões (Esquerda/Direita) que mudam a visibilidade.
3. Usa View.VISIBLE para mostrar e View.INVISIBLE para esconder.

Notas Importantes

Salva regularmente: Usa Ctrl+S frequentemente.
Em caso de erro: Verifica a consola (abaixo) para mensagens de erro.
Copia exatamente: Mesmo um caráter errado pode quebrar o código.
Testa passo a passo: Não deixes tudo para o fim.

Glossário

Termo	O que faz (Explicação Simples)
var	Cria uma variável que **pode mudar** de valor. *(Ex: a velocidade atual da mota)*
val	Cria um valor **fixo** que nunca muda. *(Ex: a velocidade máxima da mota)*
onCreate()	É o **ponto de partida**. É o primeiro código a correr quando a aplicação abre.
findViewById()	**Encontra uma peça** visual (botão, texto) no ecrã para a podermos controlar no código.

setOnClickListener()	É o **"ouvido"** do botão. Define o código que vai correr quando o utilizador clica nele.
coerceAtMost()	Funciona como um **Teto**. Garante que um número nunca ultrapassa um limite máximo. *(Impede a mota de ir aos 300 km/h)*
coerceAtLeast()	Funciona como um **Chão**. Garante que um número nunca desce abaixo de um limite mínimo. *(Impede a velocidade de ser negativa)*
Handler	É um **Temporizador**. Permite agendar tarefas para acontecerem no futuro (ex: "corre isto daqui a 50 milissegundos").
invalidate()	**Força um desenho novo**. Avisa o Android que os dados mudaram e que ele precisa de repintar o velocímetro no ecrã.

Recursos Úteis/Documentação

Instalação Android Studio: https://developer.android.com/studio?hl=pt-br
Documentação Android: https://developer.android.com/
Kotlin Reference: https://kotlinlang.org/docs/
Android Studio Help: Menu Help > Android Studio Help