



UNIVERSIDADE

Estácio de Sá

Universidade	Estácio de Sá
Campus	Polo de Cobilândia / Vila – Velha/ES
Nome do Curso	Desenvolvimento Full Stack
Nome da Disciplina	RPG0025 - Lidando com sensores em dispositivos móveis
Turma	9001
Semestre	Segundo Semestre de 2024
Integrantes do Grupo	Tiago de Jesus Pereira Furtado
Matrícula	202306189045

VILA VELHA
2024

Micro atividade 1:

1º Implementar a visão geral e melhores práticas para acesso a sensores

1-Objetivo da Prática:

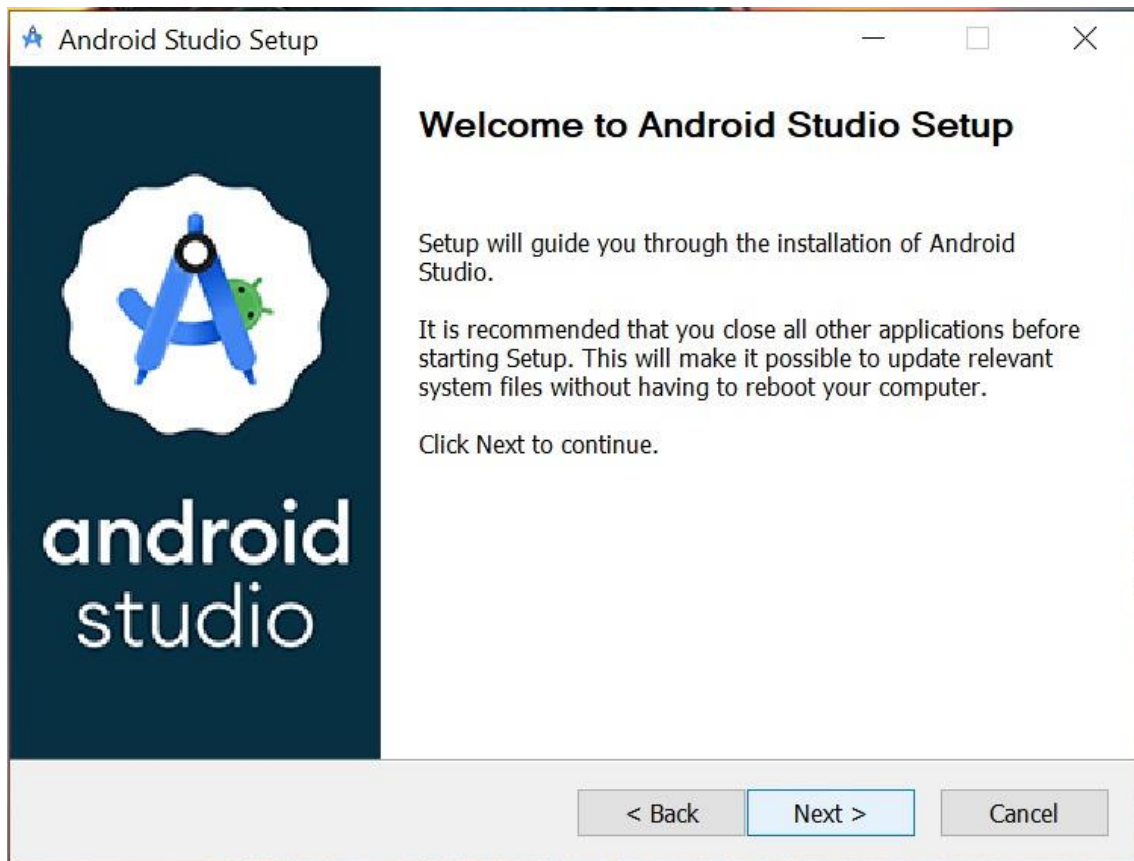
- Instalação do Android Studio e do emulador;
- Criar um app para Wear OS;
- Executar um app no emulador;
- Fazer capturas de telas no Android Studio;
- Fazer capturas de telas com app complementar.

Material necessário para a prática

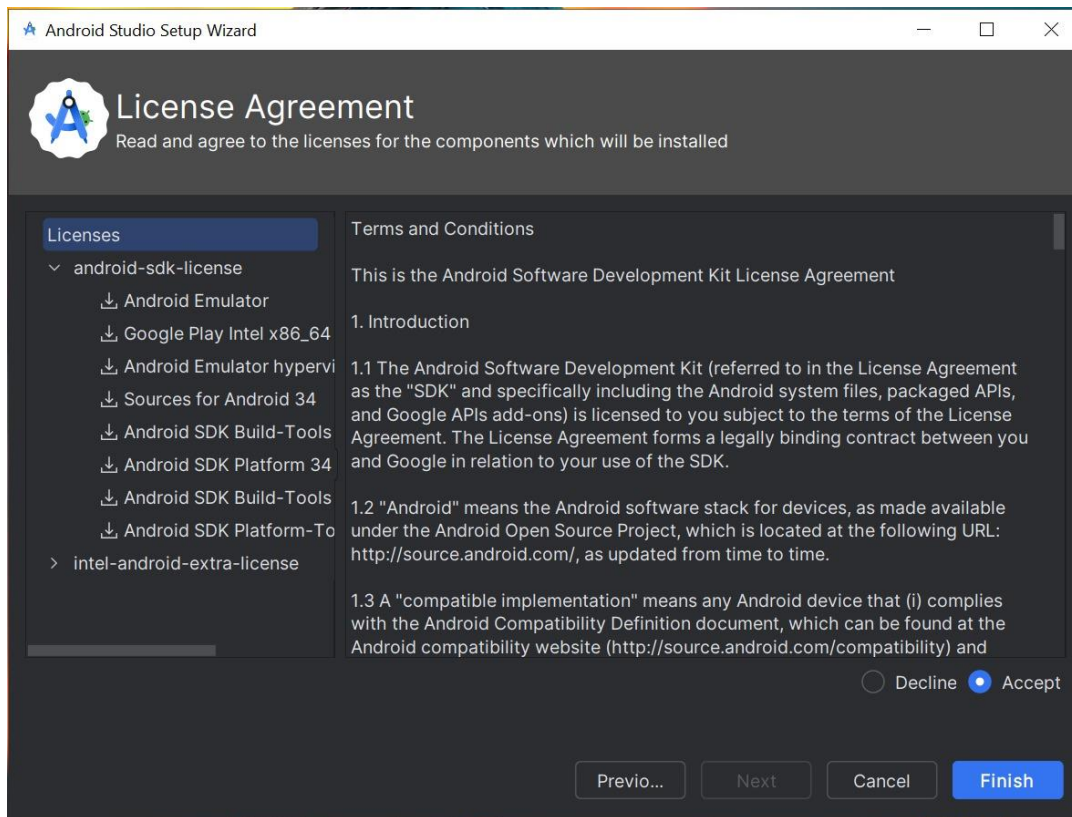
- **Android Studio:** Para o desenvolvimento de aplicativos Android.
- **Simulador Android ou iOS:** Para testar aplicativos no ambiente simulado.
- **Navegador Web:** Google Chrome, Firefox, MS Edge, Safari ou Opera.

- **Windows:**

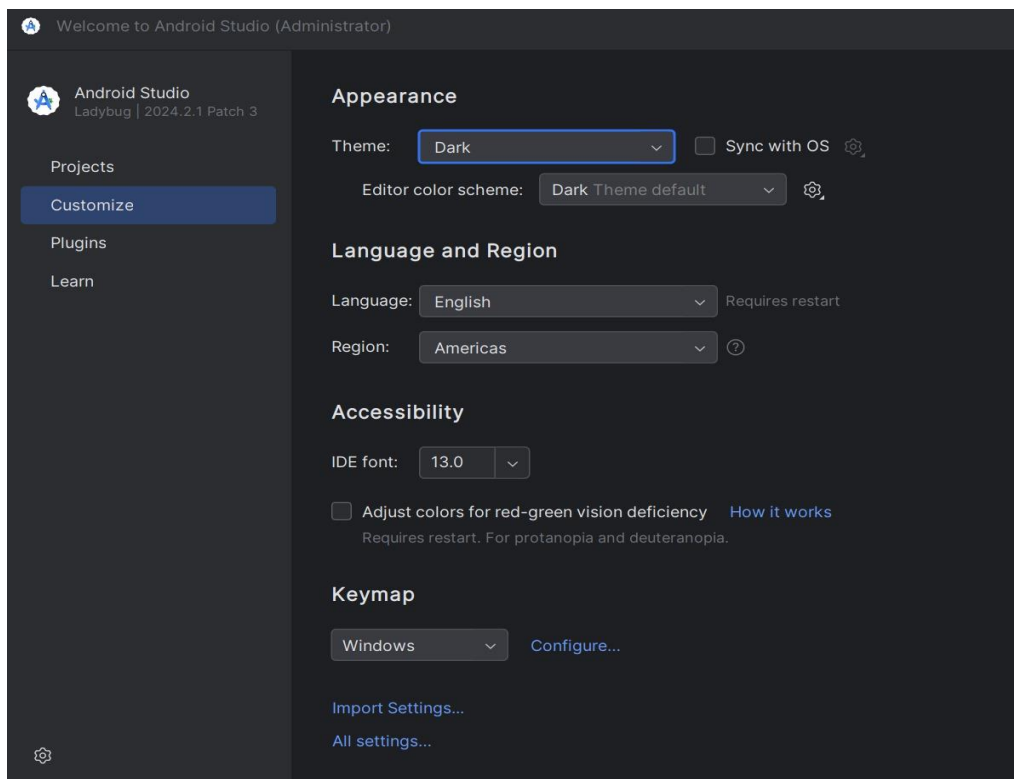
1. Abra a pasta onde você salvou o arquivo de instalação do Android Studio.
2. Clique duas vezes no arquivo.
3. Se a caixa de diálogo Controle da conta do usuário aparecer pedindo para permitir que a instalação faça mudanças no computador, clique em Sim para confirmar a instalação.
4. A caixa de diálogo Welcome to Android Studio Setup vai aparecer.



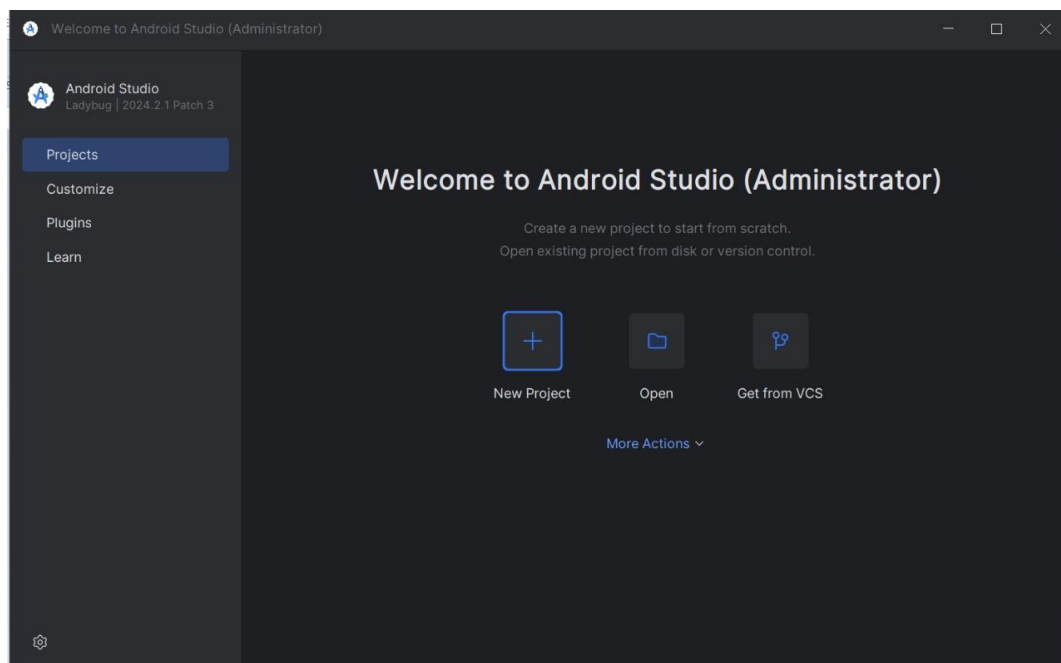
5. Clique em Next para iniciar a instalação.
6. Aceite as configurações de instalação padrão para todas as etapas.
7. Clique em Finish quando a instalação terminar para iniciar o Android Studio.



8. Escolha se prefere o tema claro ou escuro quando o Android Studio for iniciado pela primeira vez. As capturas de tela deste curso usam o tema claro, mas escolha o que você preferir.



9. Quando o download e a instalação estiverem concluídos, clique em Finish. A janela **Welcome to Android Studio** vai aparecer e você poderá começar a criar apps.



Após a instalação, configure o emulador no Android Studio.

Micro atividade 2:

2º Criando um novo projeto no Android Studio

1- Objetivo da Prática:

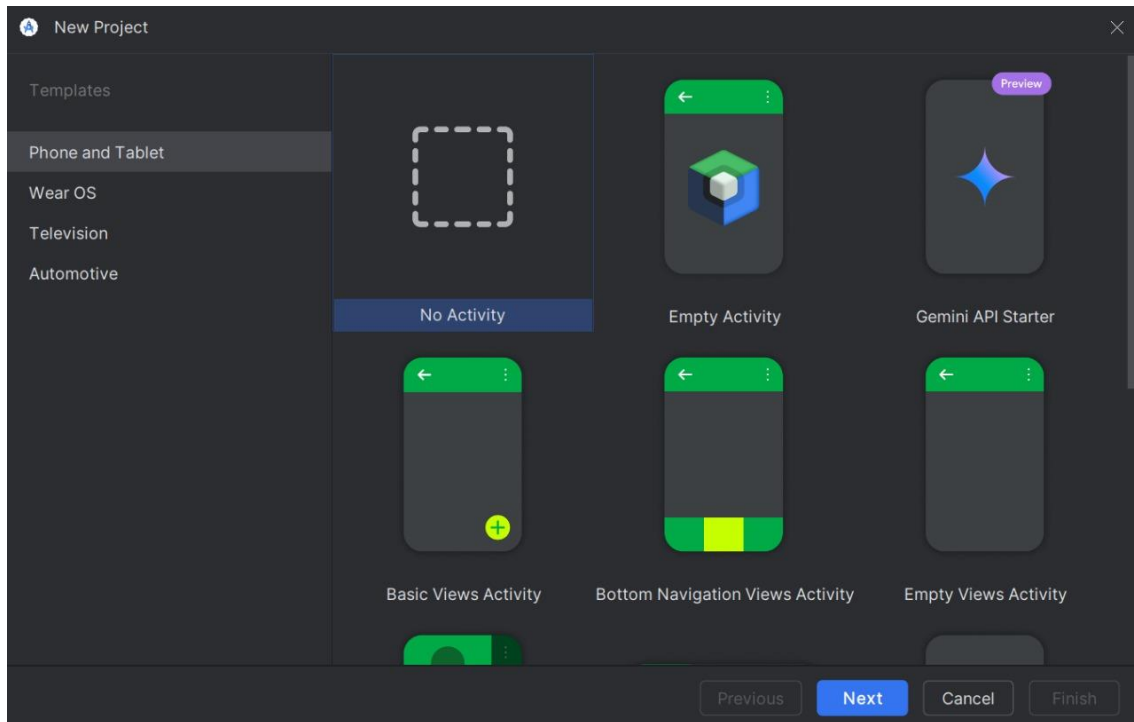
- Instalação do Android Studio e do emulador;
- Criar um app para Wear OS;
- Executar um app no emulador;
- Fazer capturas de telas no Android Studio;
- Fazer capturas de telas com app complementar.

Material necessário para a prática

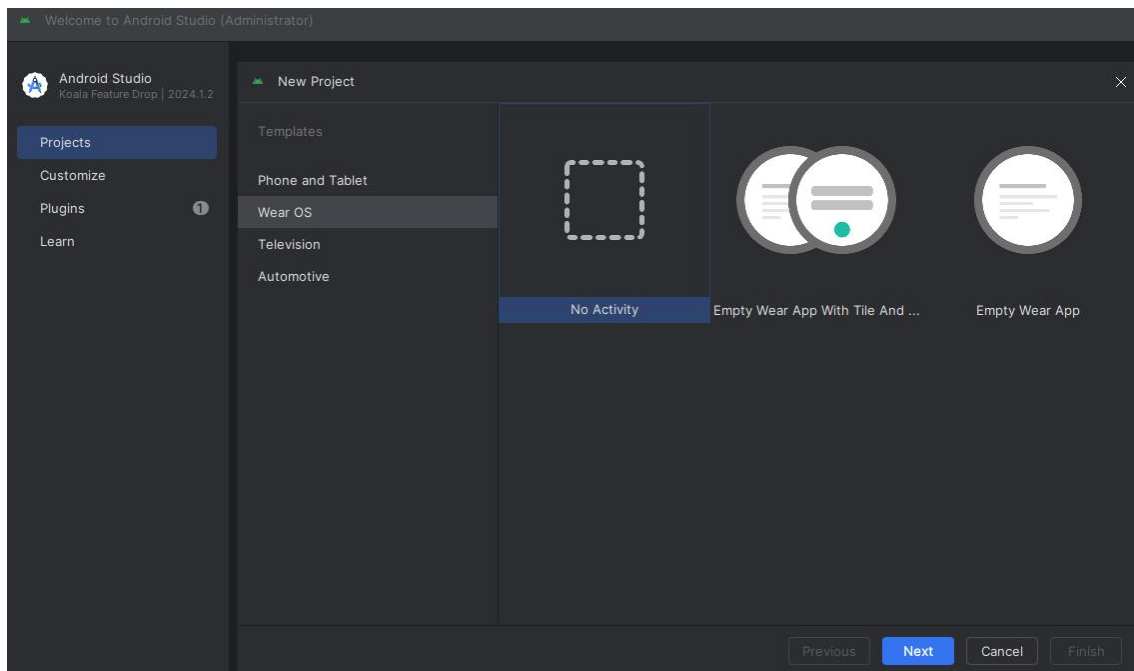
- **Android Studio:** Para o desenvolvimento de aplicativos Android.
- **Simulador Android ou iOS:** Para testar aplicativos no ambiente simulado.
- **Navegador Web:** Google Chrome, Firefox, MS Edge, Safari ou Opera.

Procedimentos

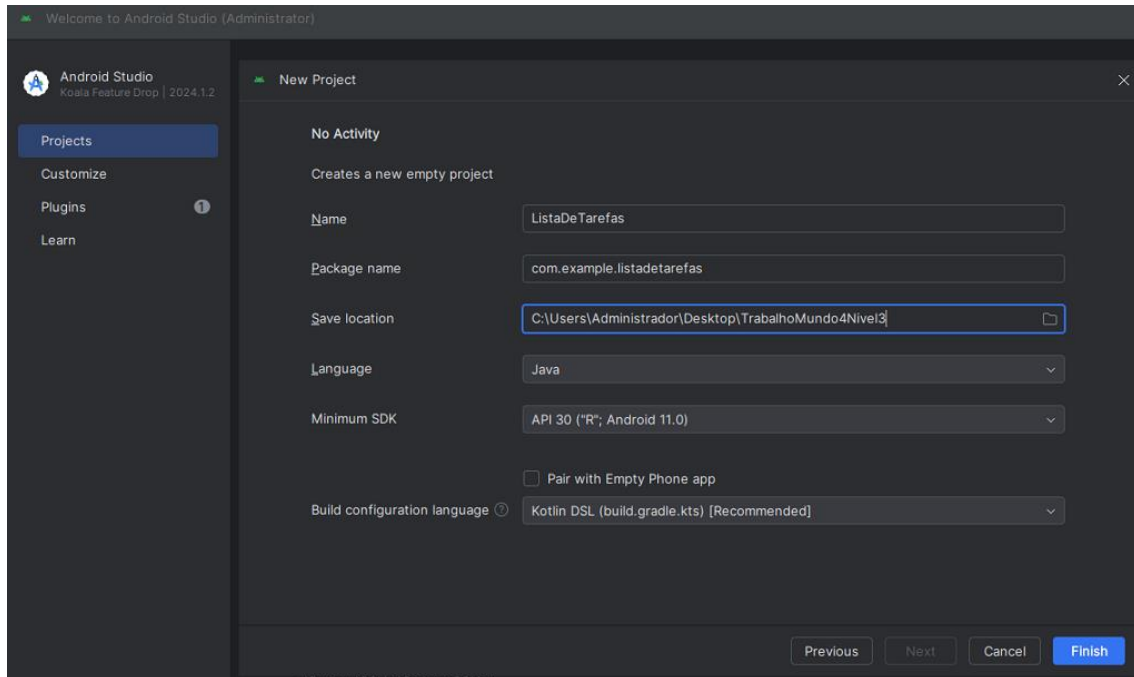
1. Abra o Android Studio e acesse file > New > New Project. A janela New Project vai aparecer.



2. No painel Templates, selecione Wear OS. Em seguida, no painel principal, selecione o modelo "No Activity" e clique em "Next".



3. Em Name, nós vamos utilizar "ListaDeTarefas" para esse exemplo. No campo "Package name", o próprio Android Studio irá sugerir algo baseado no nome do projeto, como "com.example.listadetarefas". Em "Minimum SDK", utilizaremos a API 30: Android 11.0 (R), por ser a mais recente, depois basta clicar em "Finish" e o Android Studio criará o projeto para você.



- Resultados esperados ✨

Esta microatividade permitirá que o aluno execute os passos iniciais para criar seu primeiro aplicativo para Wear OS. Ao seguir esses procedimentos, o aluno terá configurado um projeto usando um modelo do Android Studio e estará pronto para iniciar o desenvolvimento do aplicativo.

Microatividade 3:

3º Arquivos de Lógica e Configurações

1- Objetivo da Prática:

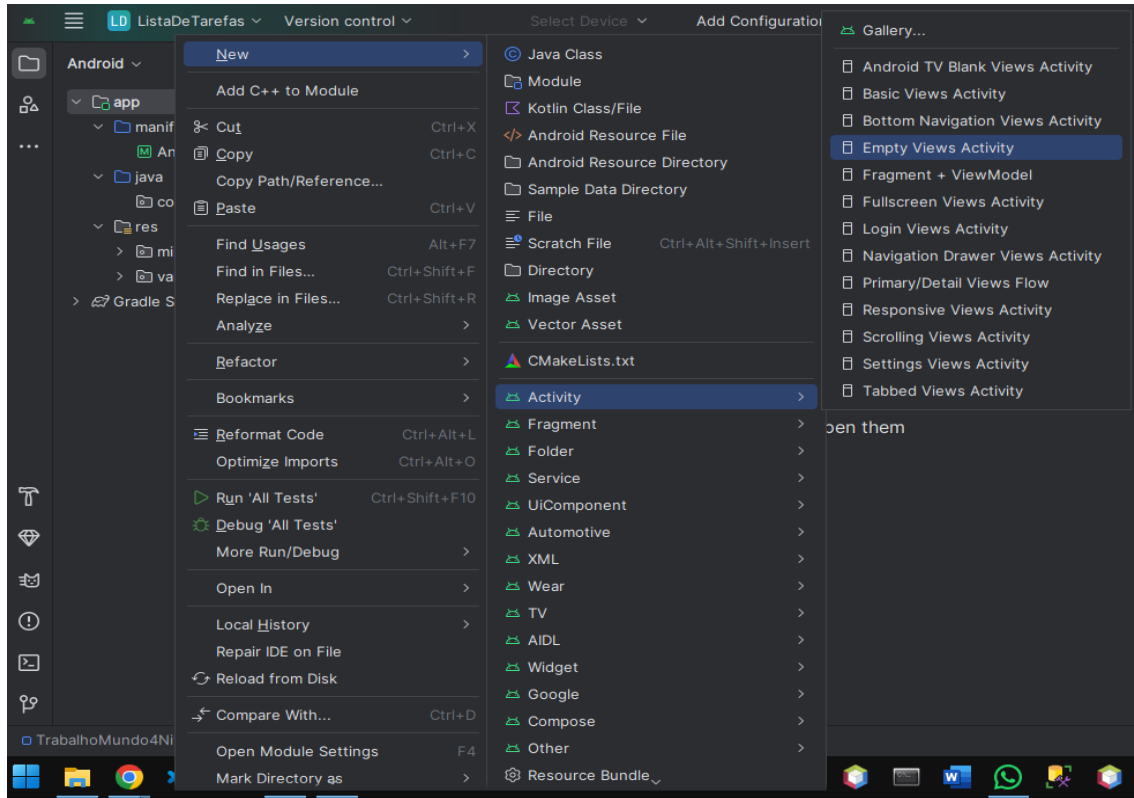
- Instalação do Android Studio e do emulador;
- Criar um app para Wear OS;
- Executar um app no emulador;
- Fazer capturas de telas no Android Studio;
- Fazer capturas de telas com app complementar.

Material necessário para a prática

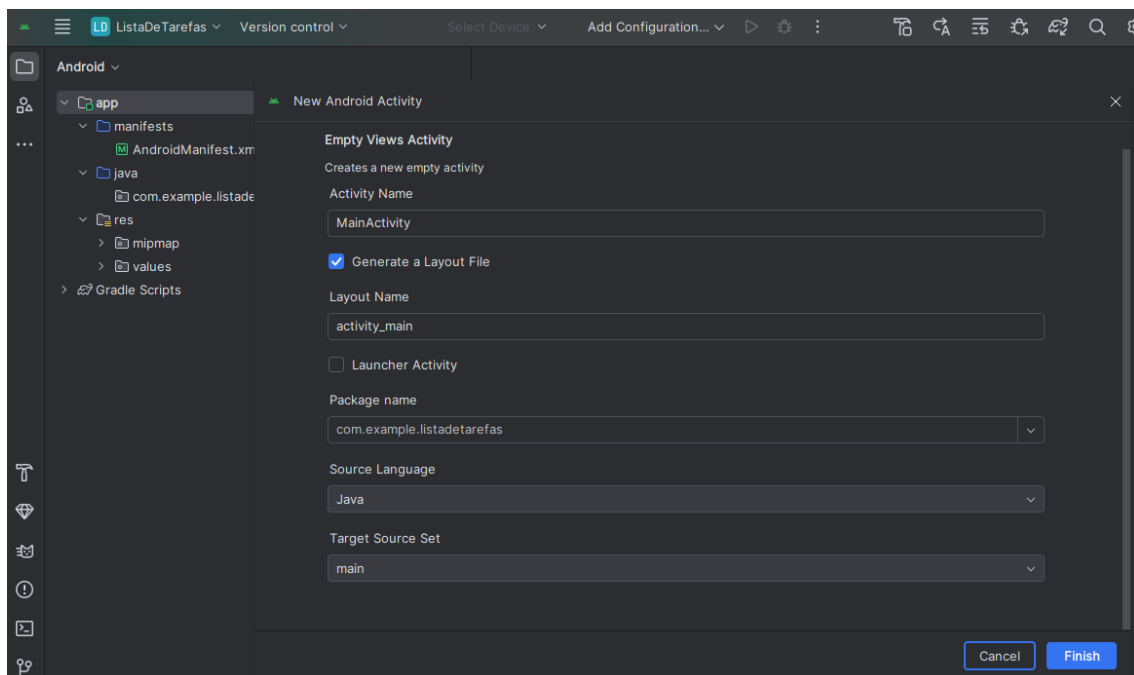
- **Android Studio:** Para o desenvolvimento de aplicativos Android.
- **Simulador Android ou iOS:** Para testar aplicativos no ambiente simulado.
- **Navegador Web:** Google Chrome, Firefox, MS Edge, Safari ou Opera.

- Procedimentos

1. Criação da MainActivity: Crie a `MainActivity.java` clicando com o botão direito em "app" e selecione New > Activity > Empty Views Activity.



2. Nome e Layout da Atividade: Na janela, mantenha o nome da atividade como `MainActivity` e o "Layout Name" como `activity_main`.



3.Interface de Usuário: Desenvolva a interface da primeira tela do aplicativo com uma `ListView` e um `Button`.

4.Permissões no AndroidManifest.xml: Localize o `AndroidManifest.xml` na pasta manifests e adicione as permissões:

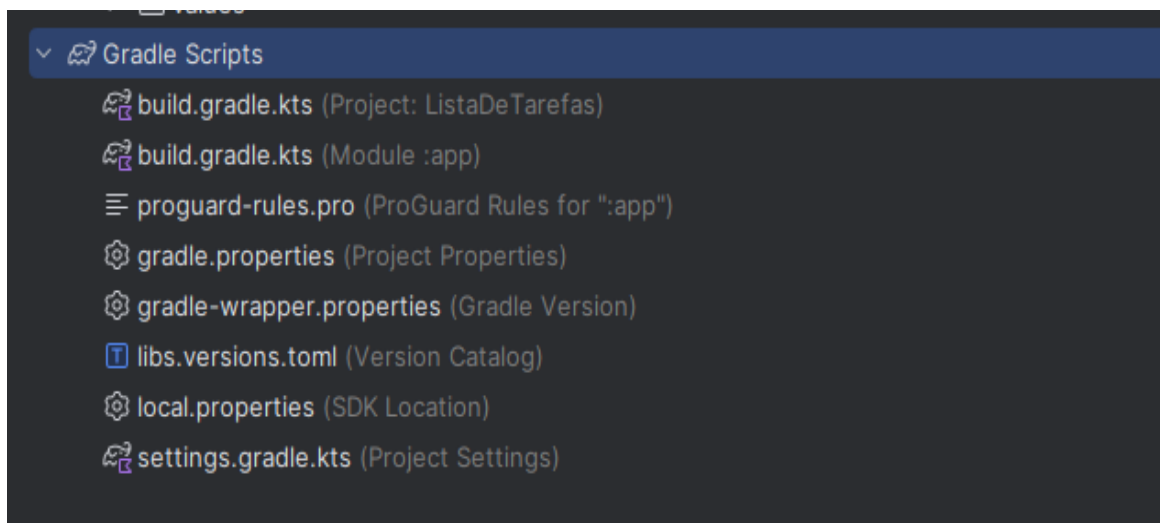
```
<uses-permission android:name="android.permission.BODY_SENSORS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

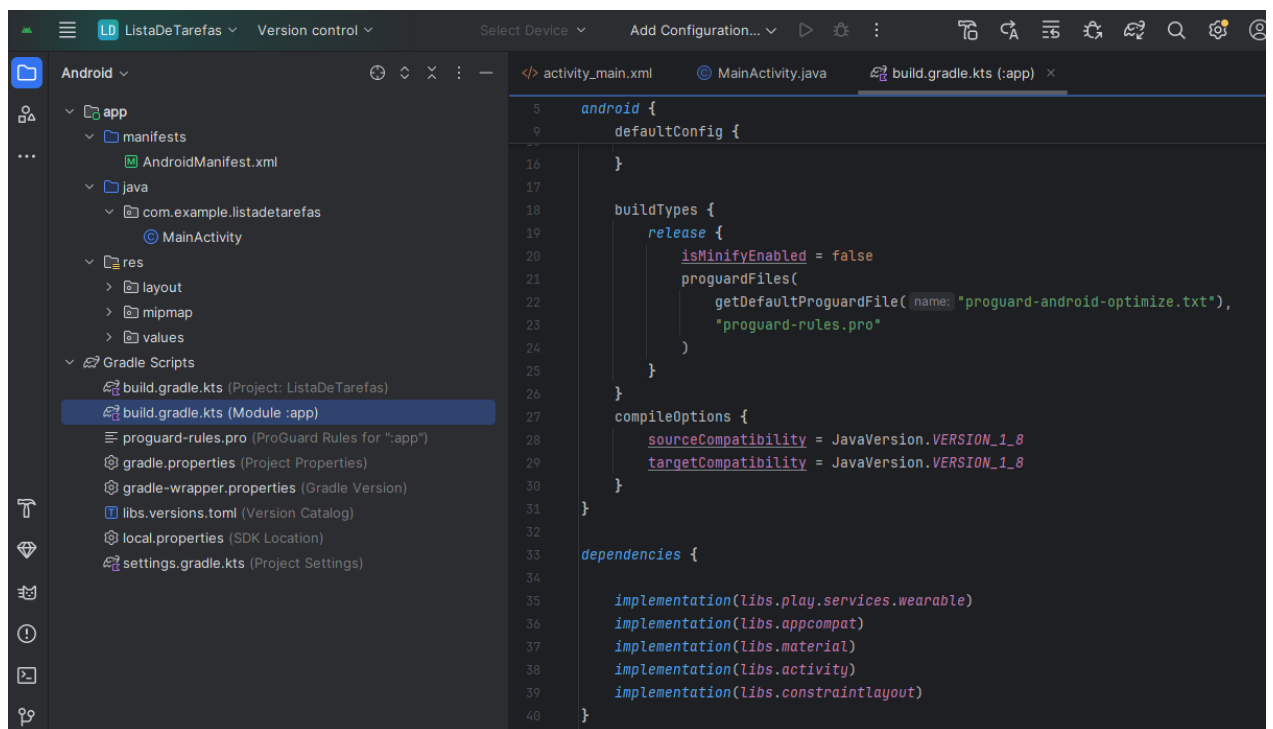
Isso permite a interação com partes do sistema.

5.Intent-filter para MainActivity: No arquivo maAdicione o elemento `intent-filter` para especificar as intents que a atividade pode responder, respondendo a intents com a ação MAIN e a categoria LAUNCHER. Exemplo:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

6. Dependências no build.gradle: Na área Gradle Scripts, temos o build.gradle, e lá encontraremos as dependências do projeto.

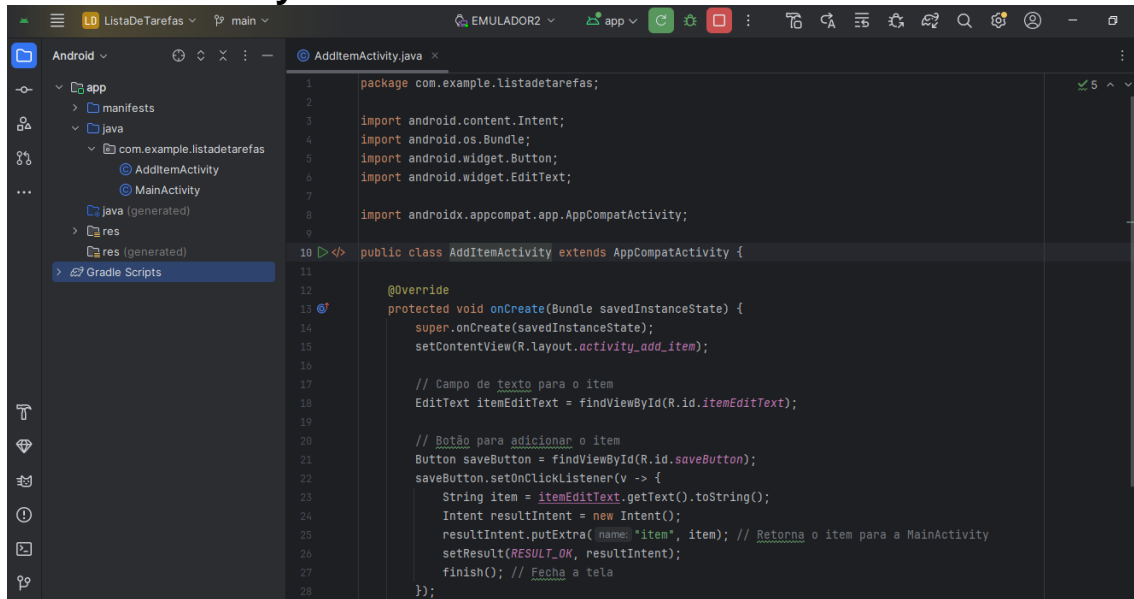




Lembre-se de sincronizar o projeto após realizar essas alterações para garantir que as dependências sejam baixadas corretamente.

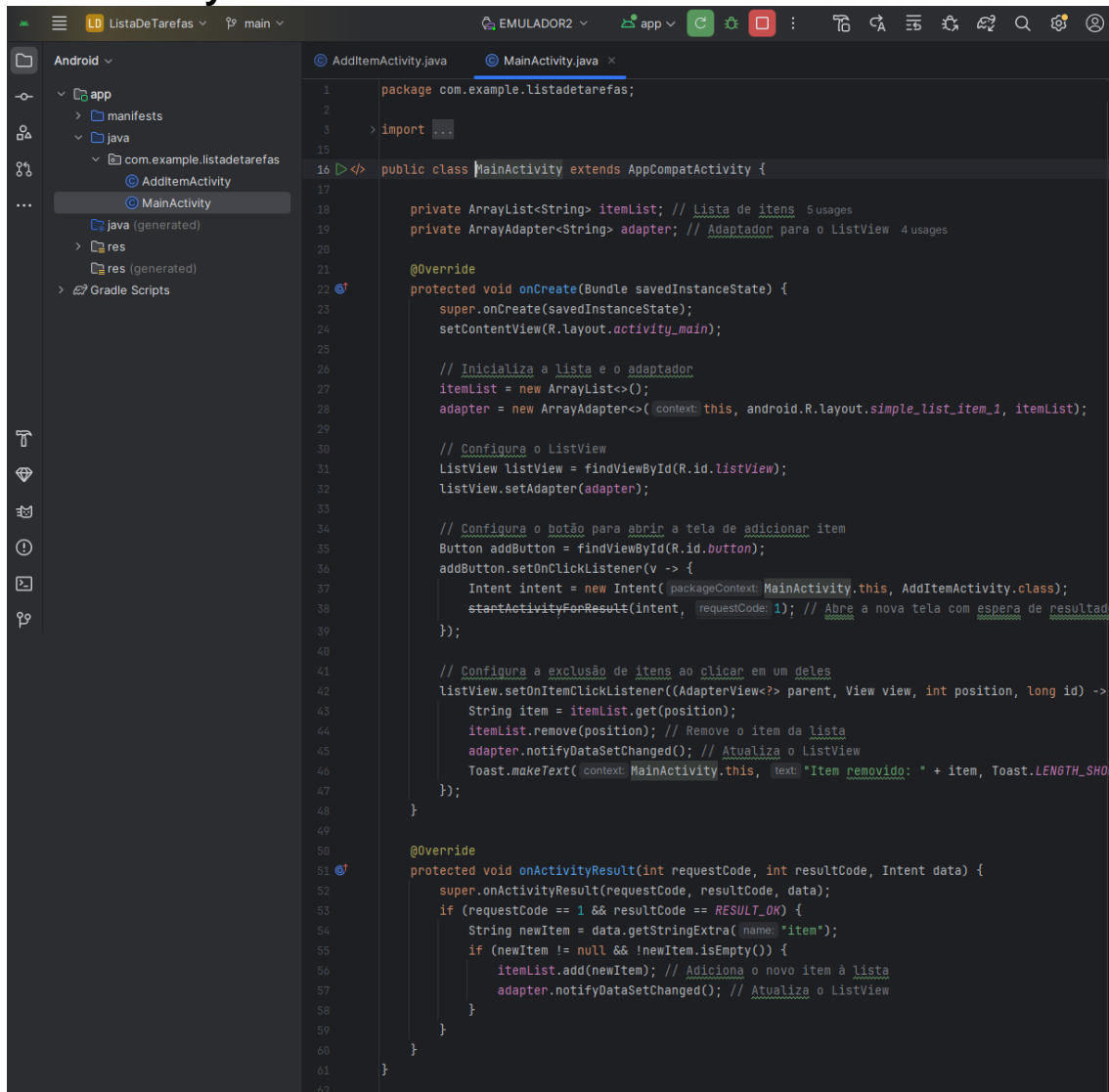
Codigos Usados Na MicroAtividade 3

AdditemActivity:



```
1 package com.example.listadetarefas;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.widget.Button;
6 import android.widget.EditText;
7
8 import androidx.appcompat.app.AppCompatActivity;
9
10 public class AddItemActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_add_item);
16
17         // Campo de texto para o item
18         EditText itemEditText = findViewById(R.id.itemEditText);
19
20         // Botão para adicionar o item
21         Button saveButton = findViewById(R.id.saveButton);
22         saveButton.setOnClickListener(v -> {
23             String item = itemEditText.getText().toString();
24             Intent resultIntent = new Intent();
25             resultIntent.putExtra(name="item", item); // Retorna o item para a MainActivity
26             setResult(RESULT_OK, resultIntent);
27             finish(); // Fecha a tela
28         });
29     }
30 }
```

MainActivity:

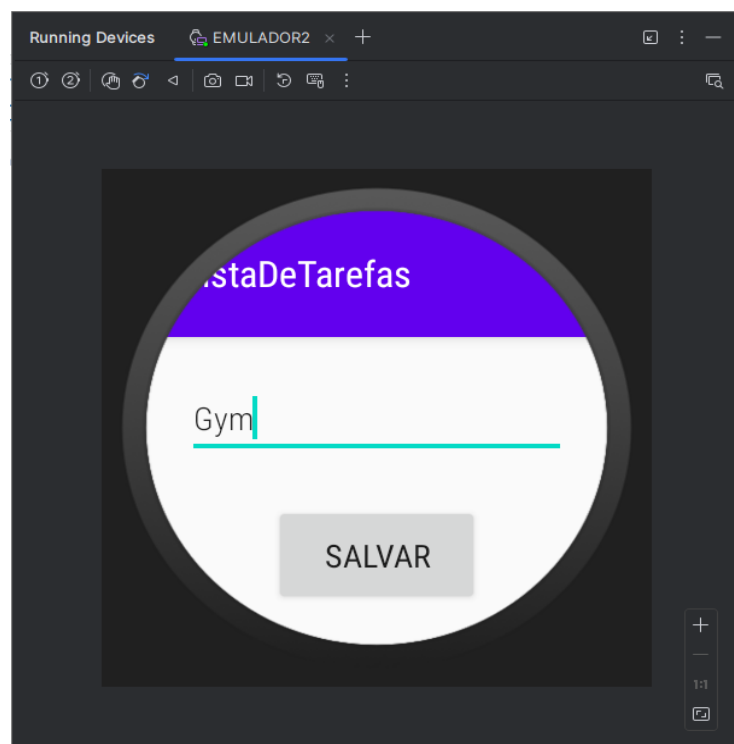
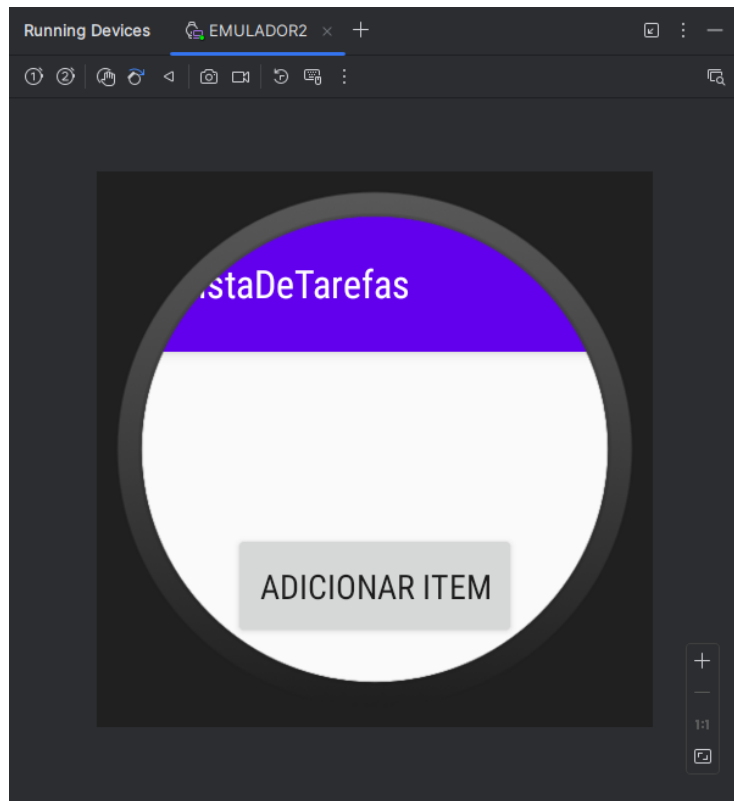


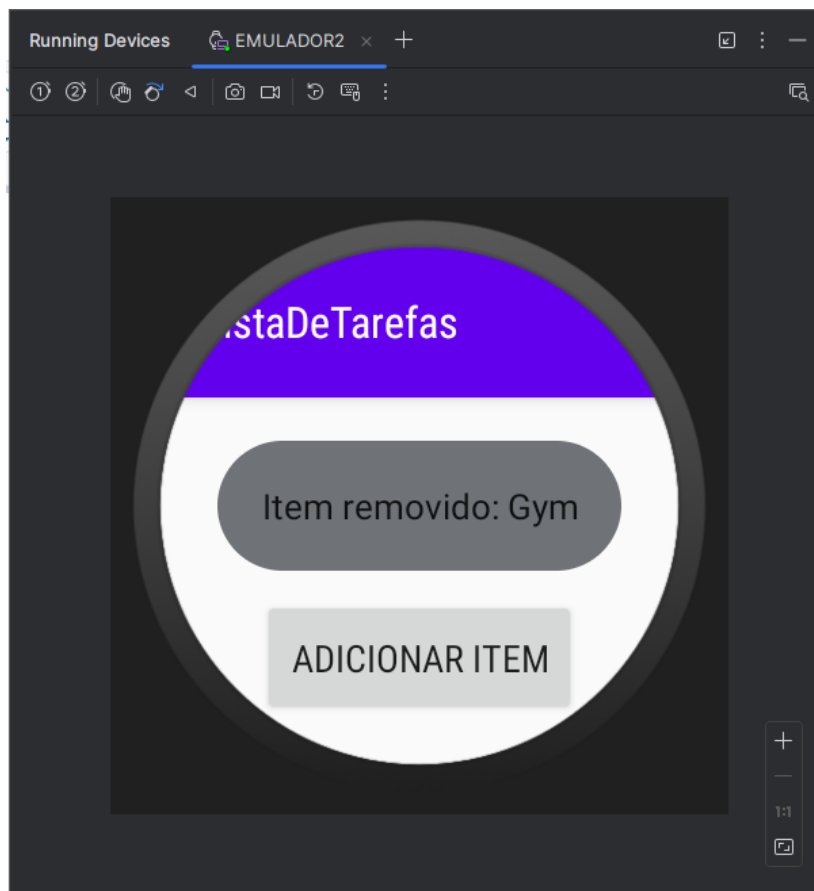
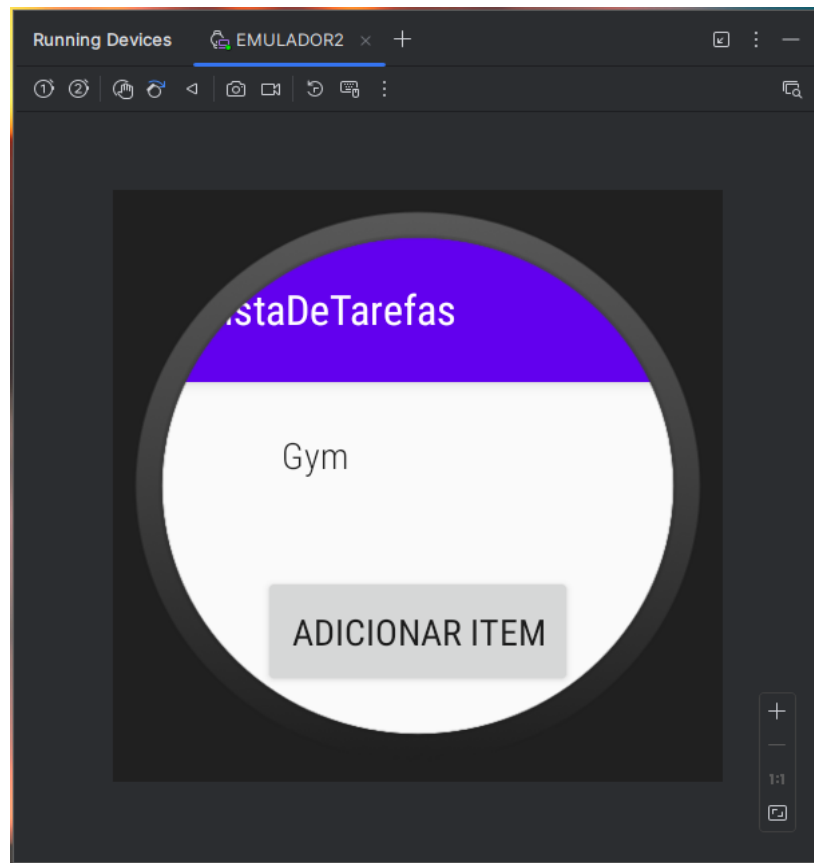
```
1 package com.example.listadetarefas;
2
3 import java.util.ArrayList;
4 import androidx.appcompat.app.AppCompatActivity;
5
6 public class MainActivity extends AppCompatActivity {
7
8     private ArrayList<String> itemList; // Lista de itens 5 usages
9     private ArrayAdapter<String> adapter; // Adaptador para o ListView 4 usages
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15
16         // Inicializa a lista e o adaptador
17         itemList = new ArrayList<>();
18         adapter = new ArrayAdapter<>(context=this, android.R.layout.simple_list_item_1, itemList);
19
20         // Configura o ListView
21         ListView listView = findViewById(R.id.listView);
22         listView.setAdapter(adapter);
23
24         // Configura o botão para abrir a tela de adicionar item
25         Button addButton = findViewById(R.id.button);
26         addButton.setOnClickListener(v -> {
27             Intent intent = new Intent(packageContext=MainActivity.this, AddItemActivity.class);
28             startActivityForResult(intent, requestCode=1); // Abre a nova tela com espera de resultado
29         });
30
31         // Configura a exclusão de itens ao clicar em um deles
32         listView.setOnItemClickListener((AdapterView<?> parent, View view, int position, long id) -> {
33             String item = itemList.get(position);
34             itemList.remove(position); // Remove o item da lista
35             adapter.notifyDataSetChanged(); // Atualiza o ListView
36             Toast.makeText(context=MainActivity.this, text="Item removido: " + item, Toast.LENGTH_SHORT);
37         });
38
39     @Override
40     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
41         super.onActivityResult(requestCode, resultCode, data);
42         if (requestCode == 1 && resultCode == RESULT_OK) {
43             String newItem = data.getStringExtra(name="item");
44             if (newItem != null && !newItem.isEmpty()) {
45                 itemList.add(newItem); // Adiciona o novo item à lista
46                 adapter.notifyDataSetChanged(); // Atualiza o ListView
47             }
48         }
49     }
50 }
```

- Resultados esperados ✨

Nesta microatividade o aluno aprenderá os primeiros passos para criação do aplicativo. No Android Studio precisamos configurar alguns arquivos com informações do aplicativo e do dispositivo para o qual iremos desenvolver.

Resultados dos Codigos Acimas:





Micro atividade 4:

4º Criando um emulador

1- Objetivo da Prática:

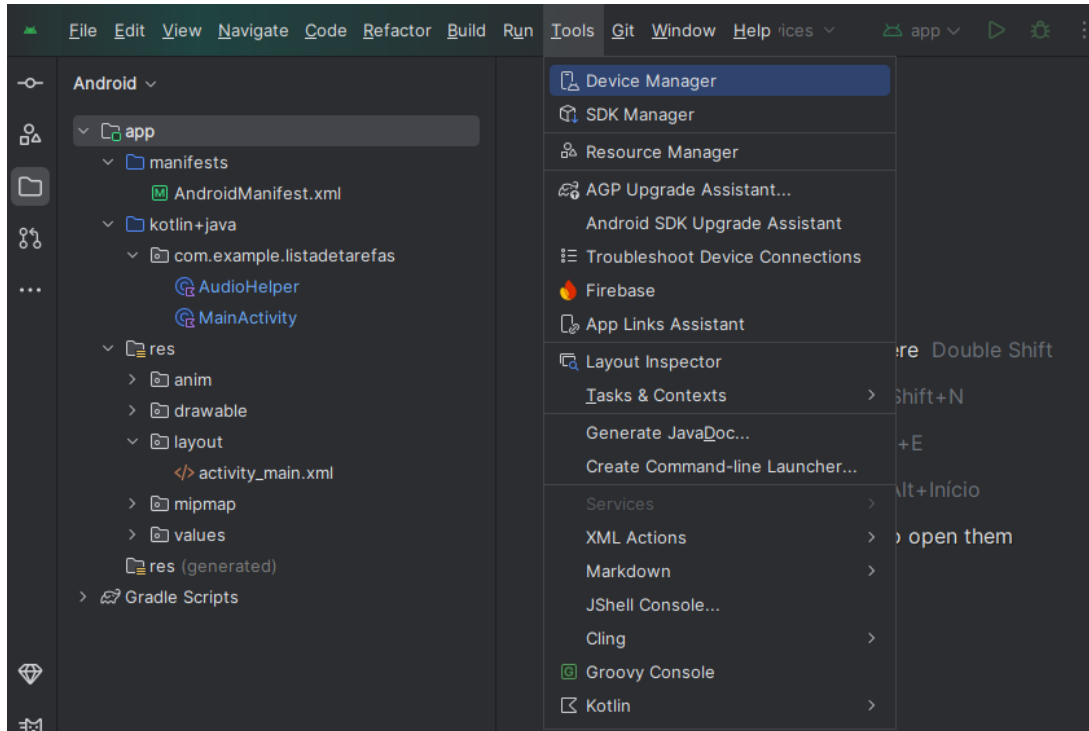
- Instalação do Android Studio e do emulador;
- Criar um app para Wear OS;
- Executar um app no emulador;
- Fazer capturas de telas no Android Studio;
- Fazer capturas de telas com app complementar.

Material necessário para a prática















- **Android Studio:** Para o desenvolvimento de aplicativos Android.
- **Simulador Android ou iOS:** Para testar aplicativos no ambiente simulado.
- **Navegador Web:** Google Chrome, Firefox, MS Edge, Safari ou Opera.

- Procedimentos

1. No [Android Studio](#), acesse o Device Manager pelo caminho Tools > Device Manager. É um botão do lado direito da barra de ferramentas que mostra um Android abrindo a cabeça ao lado de um dispositivo com um display roxo.

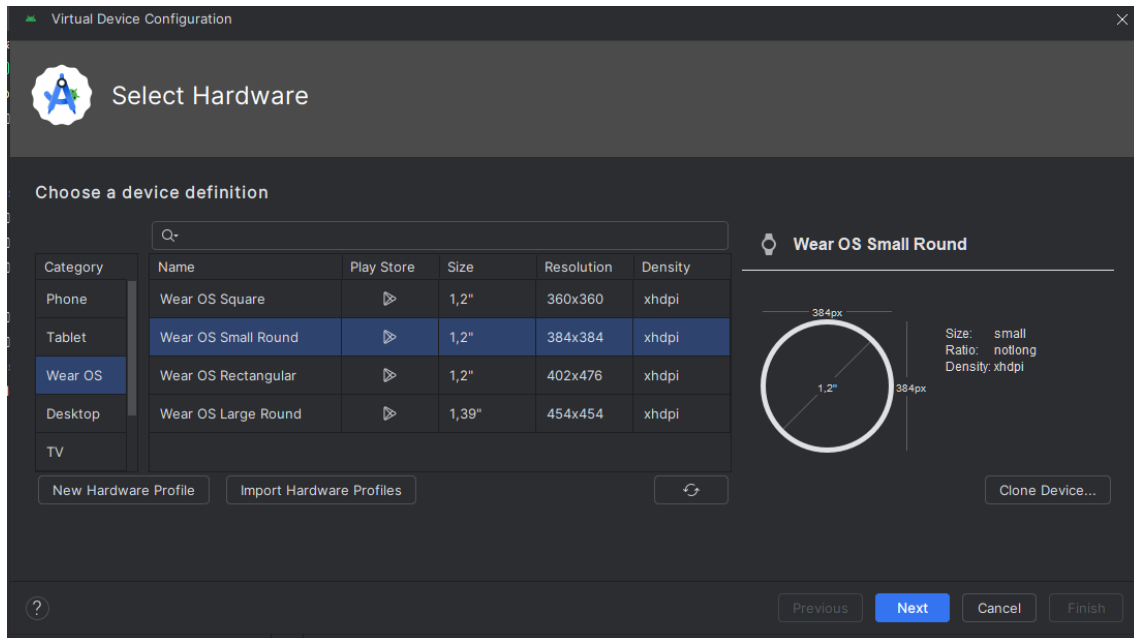


2. Depois que o Device Manager abrir, provavelmente você verá um emulador já criado e alguns detalhes sobre ele, principalmente o tipo de emulador, a API que está sendo usada e o tipo de CPU. Importante o Device Manager pode abrir como uma janela dentro do Android Studio ou como uma janela flutuante.

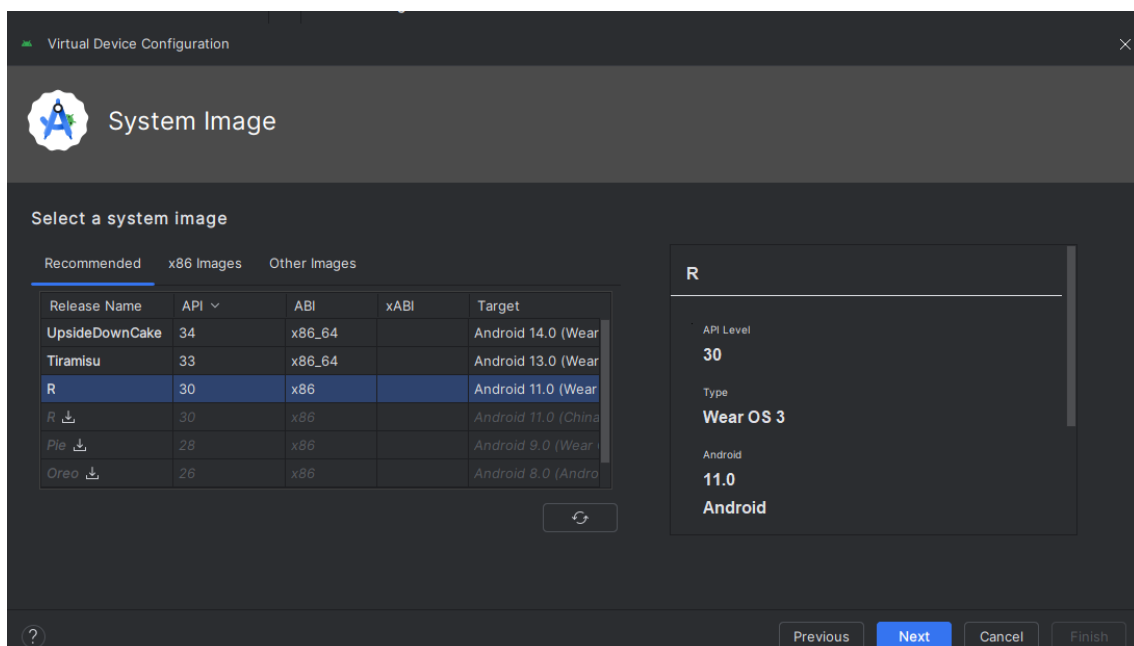
Device Manager				
				
Name	API	Type		
 Wear OS Small Round API 34 Android 14.0 ("UpsideDownCake") x86_64	34	Virtual	▶	⋮
 Pixel 3a API 34 Android 14.0 ("UpsideDownCake") x86_64	34	Virtual	▶	⋮
 Wear OS Small Round API 33 Android 13.0 ("Tiramisu") x86_64	33	Virtual	▶	⋮
 Pixel 5 API 35 Android API 35 x86_64	35	Virtual	▶	⋮
 EMULADOR_TESTE Android 14.0 ("UpsideDownCake") x86_64	34	Virtual	▶	⋮
 EMULADOR-WEAROS Android 13.0 ("Tiramisu") x86_64	33	Virtual	▶	⋮
 MEU EMULADOR Android 11.0 ("R") x86	30	Virtual	▶	⋮
 EMULADOR2 Android 11.0 ("R") x86	30	Virtual	▶	⋮
 EMULADO_TRABALHO Android API 35 x86_64	35	Virtual	▶	⋮
 Pixel 4 API 30 Android 11.0 ("R") x86	30	Virtual	▶	⋮
 EMULADOR3 Android 11.0 ("R") x86	30	Virtual	▶	⋮
 emulador34 Android 14.0 ("UpsideDownCake") x86	34	Virtual	▶	⋮
 Wear OS Small Round API 34 2 Android 14.0 ("UpsideDownCake") x86_64	34	Virtual	▶	⋮

3. Para entender melhor o entendimento desse processo, vamos criar um novo dispositivo virtual:

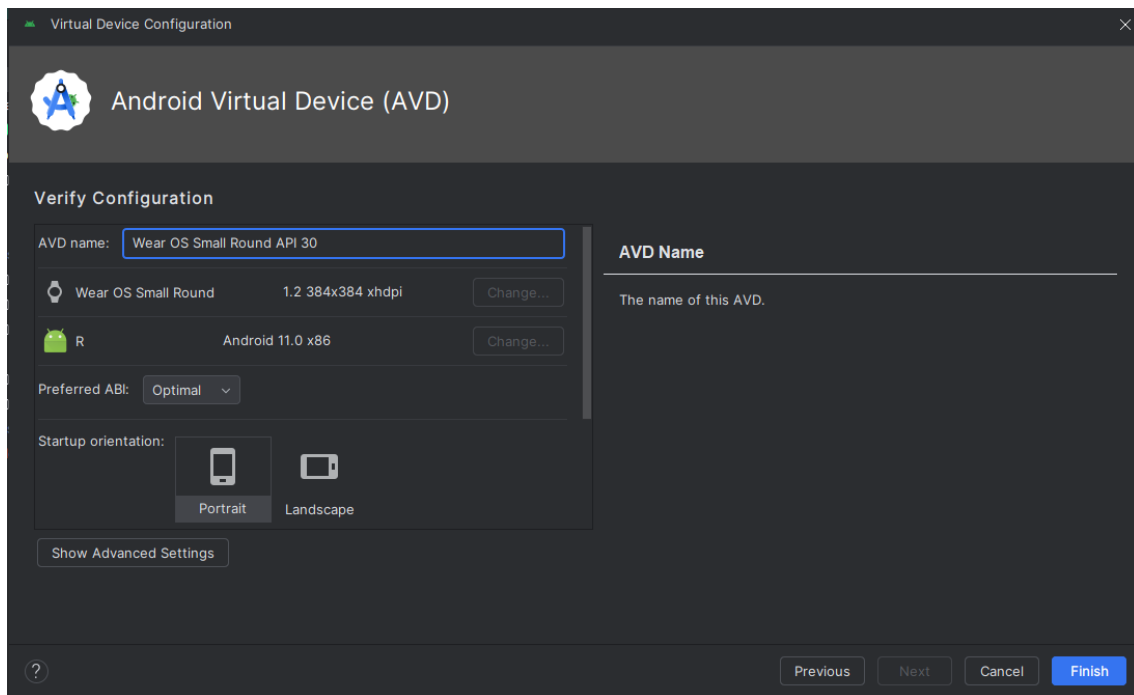
- Clique em Create Device, escolha a categoria Wear OS no lado esquerdo. Selecione o hardware que deseja emular (no nosso exemplo, Wear OS Small Round). Clique em Next



4. Escolha o sistema operacional que você deseja emular (por exemplo, Wear OS API 30). Se a imagem do sistema não estiver disponível, clique no link "Download" ao lado do nome para baixá-lo. Após selecionar a imagem do sistema, clique no botão Next.



5. A última tela permite confirmar suas escolhas e oferece opções para configurar algumas outras propriedades, como nome do dispositivo, orientação de inicialização e tamanho da memória RAM. Por enquanto, use os padrões e clique em Finish.



- Resultados esperados ✨

Esta microatividade destaca como criar emuladores de dispositivos Wearable, permitindo testar o funcionamento de aplicativos. Isso é útil para o desenvolvimento e teste de aplicativos Wear OS antes de implantá-los em dispositivos reais.

Micro atividade 5:

👉 5º Fazer capturas de telas com app complementar

1- Objetivo da Prática:

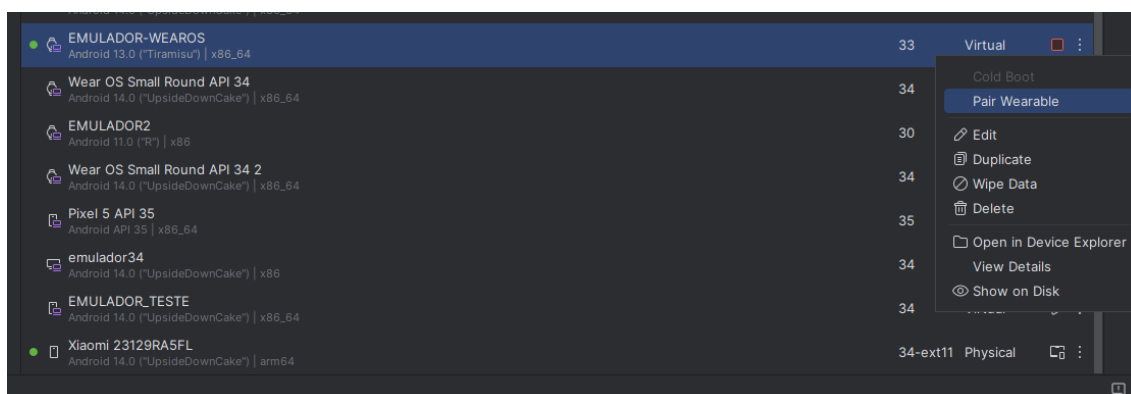
- Instalação do Android Studio e do emulador;
- Criar um app para Wear OS;
- Executar um app no emulador;
- Fazer capturas de telas no Android Studio;
- Fazer capturas de telas com app complementar.

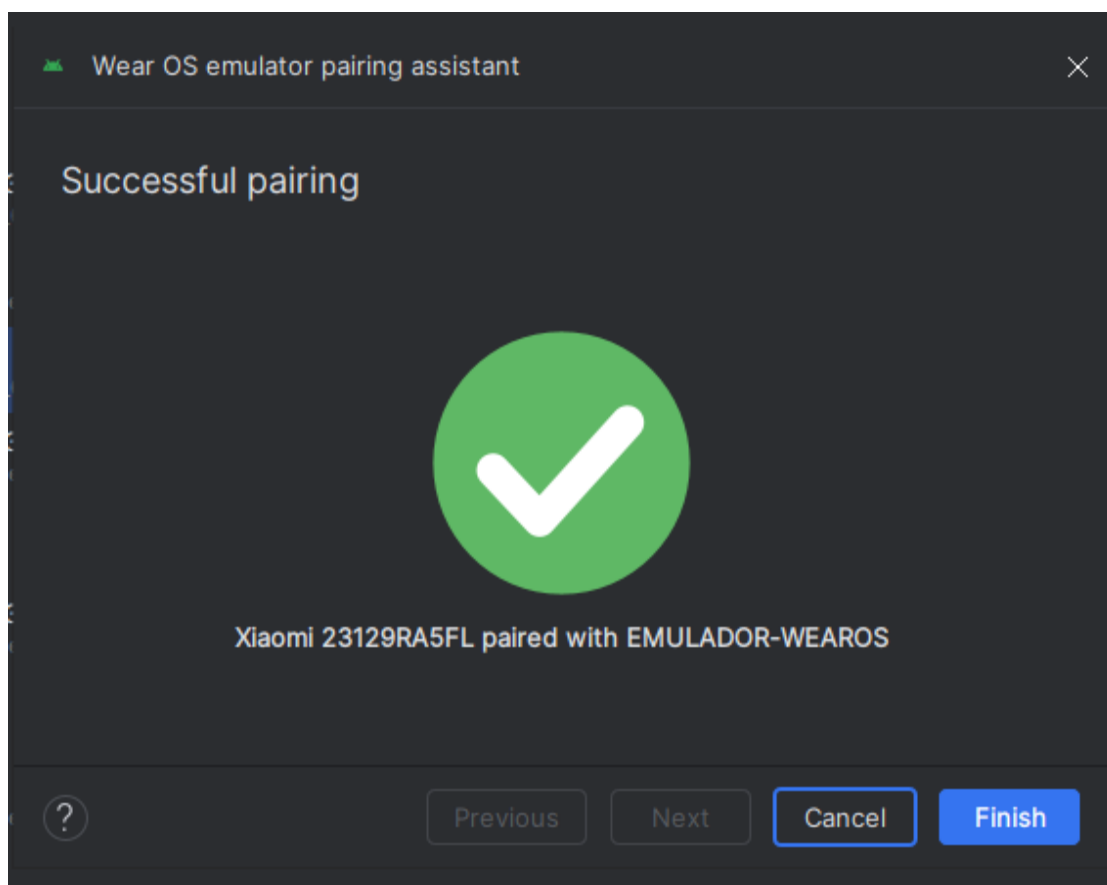
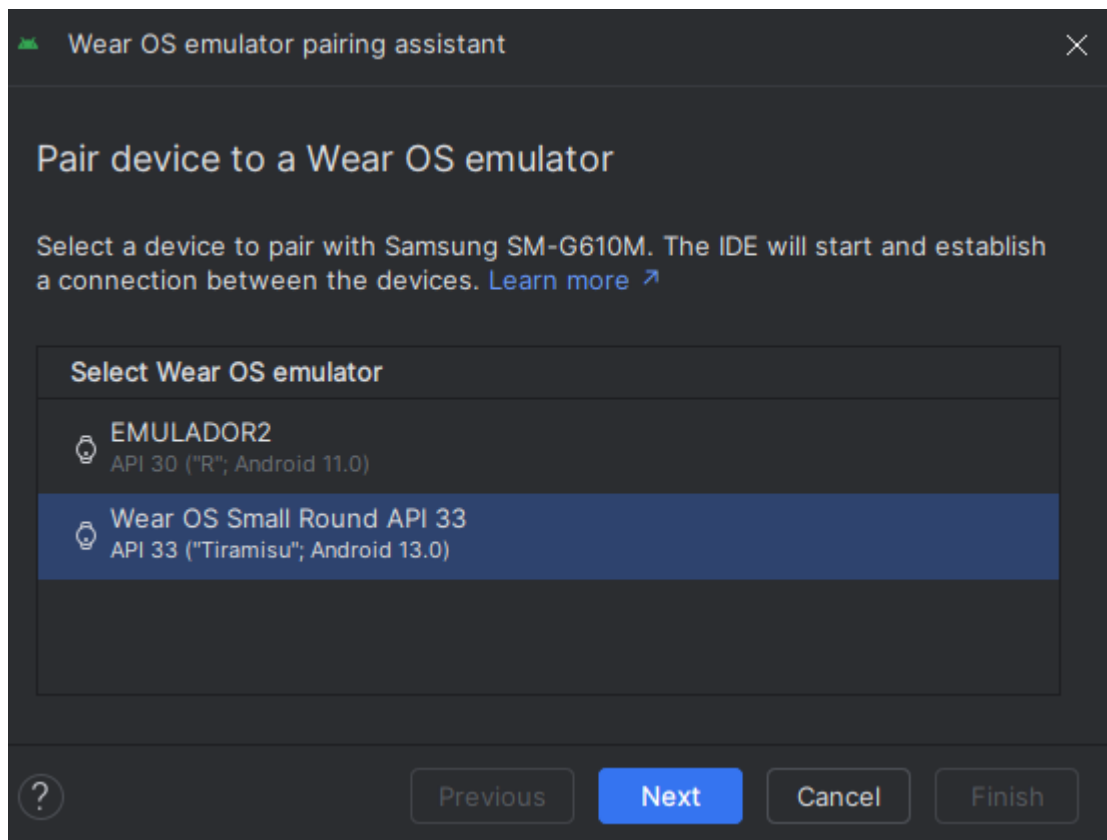
Material necessário para a prática

- **Android Studio:** Para o desenvolvimento de aplicativos Android.
- **Simulador Android ou iOS:** Para testar aplicativos no ambiente simulado.
- **Navegador Web:** Google Chrome, Firefox, MS Edge, Safari ou Opera.

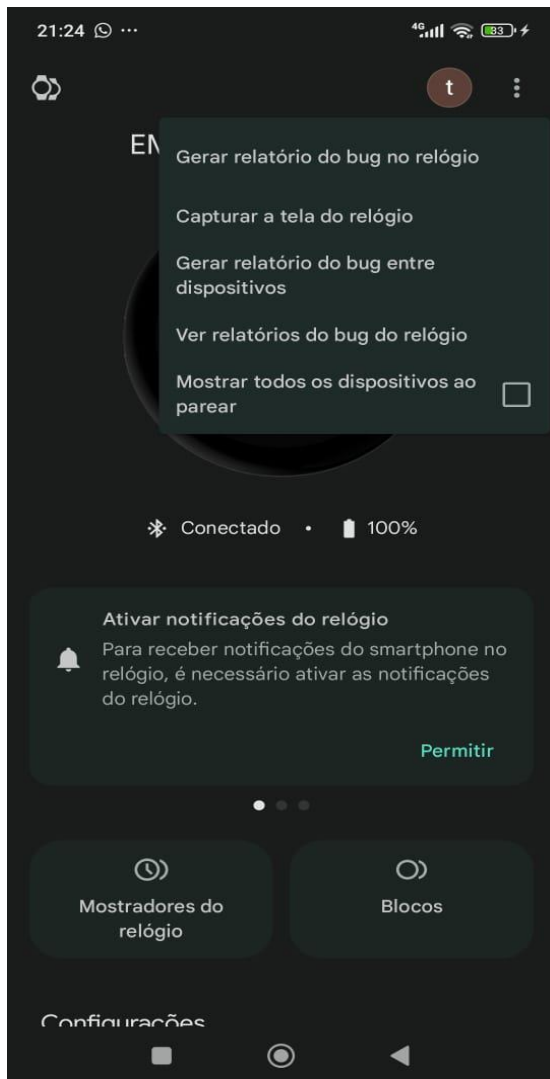
- Procedimentos

1. Na IU, encontre a tela que você quer capturar.
2. No smartphone Android, ative as Opções do desenvolvedor, se ainda não estiverem ativadas. Para isso, acesse Configurações > Sobre o telefone e toque em Número da versão sete vezes.
3. Abra o app complementar do Wear no smartphone.
4. Toque no botão flutuante de três pontos no canto superior direito para abrir o menu.
5. Toque em “Fazer captura de tela do wearable”. Esta mensagem vai aparecer: Solicitação de captura de tela enviada. Depois, você receberá estas notificações: Pronto para enviar uma captura de tela do relógio e Toque para enviar.
6. Toque na notificação para conferir as opções de envio ou compartilhamento da captura de tela por Bluetooth, Gmail ou outros meios.





Depois de Parel o Celular Segue abaixo o Resultado da Captura da Tela do Relógio:



- Resultados esperados ✨

Com esta microatividade o aluno compreenderá outra forma de realizar a captura de tela da UI app para wearables.

Missão Prática | Lidando com sensores em dispositivos móveis!

Nesta atividade a seguir compreenderemos que os apps do Wear OS podem funcionar como um dos principais frameworks para o desenvolvimento de aplicações mobile. Um aplicativo Wearable pode ter várias especialidades, desde entretenimento e comunicação.

Contextualização

Para uma melhoria na eficiência e na comunicação interna, a empresa “Doma” quer desenvolver um aplicativo Wear OS para assistência aos funcionários que têm necessidades especiais, uma forma de solidificar a interação entre os mesmos.

Assim, com os aplicativos wearables podem usar áudio para fornecer informações em tempo real, como leitura de mensagens de texto, notificações, lembretes e respostas a comandos de voz. Isso pode ser especialmente útil para pessoas com deficiência visual.

Além de serem úteis para treinamento e educação. Aplicativos podem usar áudio para fornecer instruções, dicas e feedbacks durante o aprendizado ou a prática de novas habilidades.

Outra funcionalidade que a empresa quer adotar, é um aplicativo wearable que pode usar o áudio para fornecer alertas de segurança, como notificações de emergência, alertas de tempestades, notícias importantes ou informações críticas.

Roteiro de prática

- Material necessário para a prática

- Editor de texto ou IDE sendo opções sugeridas: VS Code;
- Flutter SDK, o arquivo que permite utilizar a ferramenta;
- Android Studio e/ou xCode;
- Simulador Android ou iOS.
- Navegador Web: Google Chrome, Firefox, MS Edge, Safari ou Opera.

- Procedimentos

1. Configuração do Ambiente:

- Certifique-se de ter seu ambiente configurado.
- Prepare um ambiente de simulação para Wear OS ou conecte um dispositivo wearable real.

2. Implementação de Saídas de áudio :

- `AudioDeviceInfo.TYPE_BUILTIN_SPEAKER`, em dispositivos com um alto-falante integrado.
- `AudioDeviceInfo.TYPE_BLUETOOTH_A2DP` quando um fone de ouvido Bluetooth estiver pareado e conectado.
- Utilize o método `getDevices()` com o valor de `FEATURE_AUDIO_OUTPUT` para enumerar todas as saídas de áudio:

3. Detecção Dinâmica de Dispositivos de Áudio:

- Seu app pode registrar um callback para detectar quando isso acontece usando `registerAudioDeviceCallback`:

4. Facilitando a Conexão Bluetooth:

- Se o app exigir que um fone de ouvido seja conectado para continuar, em vez de mostrar uma mensagem de erro, ofereça a opção de direcionar o usuário diretamente às configurações do Bluetooth para facilitar a conexão. Para isso, envie uma intent com `ACTION_BLUETOOTH_SETTINGS`:

5. **Reprodução de Áudio:**

- Depois de detectar uma saída de áudio adequada, o processo para tocar áudio no Wear OS é o mesmo usado em dispositivos móveis ou outros dispositivos.

6. **Uso de Alto-falantes em Dispositivos Wear OS:**

- Para dispositivos Wear OS que incluem alto-falantes, incorpore funcionalidades de áudio para enriquecer a experiência do usuário.
- Exemplos de uso incluem alarmes de relógio com notificações sonoras, apps de fitness com instruções de voz para exercícios, e apps educativos com feedback auditivo.

Códigos Usados no Trabalho:

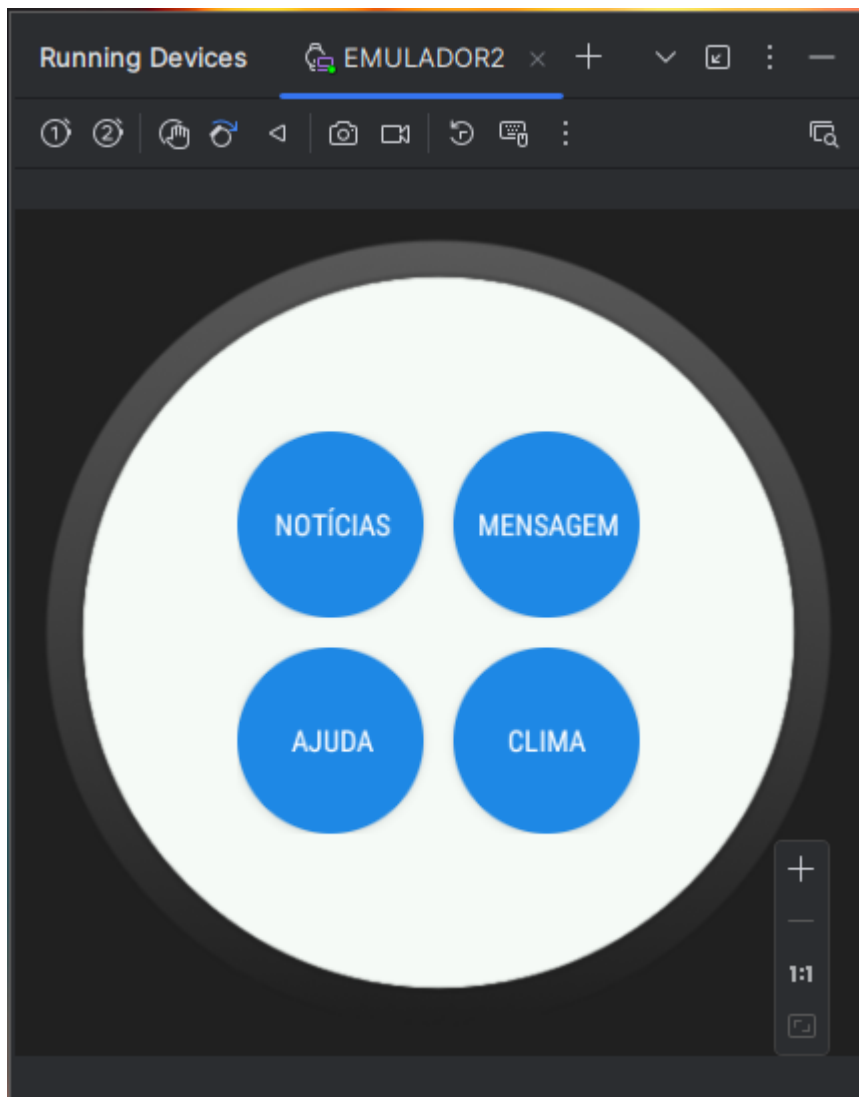
MainActivity:

```
1 package com.example.listadeterrefos
2
3 import android.Manifest
4 import android.content.Intent
5 import android.content.pm.PackageManager
6 import android.media.AudioDeviceInfo
7 import android.os.Bundle
8 import android.speech.RecognizerIntent
9 import android.speech.RecognitionListener
10 import android.speech.SpeechRecognizer
11 import android.speech.SpeechRecognizer
12 import android.view.animation.AnimationUtils
13 import android.widget.Button
14 import android.widget.Toast
15
16 import androidx.activity.result.contract.ActivityResultContracts
17 import androidx.appcompat.app.AppCompatActivity
18 import androidx.core.content.ContextCompat
19
20 class MainActivity : AppCompatActivity() {
21     private lateinit var audioHelper: AudioHelper
22     private lateinit var speechRecognizer: SpeechRecognizer
23
24     private val newsList = listOf(
25         "A copula climática da ONU reforça a necessidade de ações globais para conter os impactos climáticos.",
26         "Identificadas mudanças climáticas generalizadas no contexto de elevação dos níveis do mar.",
27         "A economia global enfrenta desafios com a instabilidade nos mercados financeiros.",
28         "Tecnologia de inteligência artificial revoluciona setores como saúde e educação em 2024.",
29         "Novas missões espaciais estão sendo planejadas para explorar Marte nos próximos anos."
30     )
31
32     private val weatherList = listOf(
33         "Hoje está ensolarado com temperatura máxima de 28 graus e mínima de 18 graus.",
34         "Amanhã há uma chance de chuva, temperatura entre 20 e 25 graus.",
35         "O clima hoje está instável, com pancadas de chuva à tarde.",
36         "Hoje o dia está parcialmente nublado com temperatura máxima de 30 graus.",
37         "Temperatura amena hoje, variando entre 15 e 25 graus, com céu limpo."
38     )
39
40     private val messageList = listOf(
41         "Parabéns Carlos Almeida! Foi uma honra confirmar nossa reunião para amanhã às 10 horas.",
42         "Parabéns Ana Silva! Não se esqueça de enviar o relatório até sexta-feira.",
43         "Parabéns João Pedro! Parabéns pelo excelente trabalho no projeto!",
44         "Parabéns Maria Clara! Podemos agendar a reunião para as 14h?",
45         "Parabéns Lucas Mendes! Feliz aniversário! Que você tenha um dia incrível!"
46     )
47
48     private val helpMessage = "Aqui estão as ações disponíveis: para ouvir uma notícia, digite 'notícia'. Para o clima, digite 'clima'."
49
50     private val requestPermissionLauncher = registerForActivityResult(
51         ActivityResultContracts.RequestPermission()
52     ) { isGranted: Boolean ->
53         if (isGranted) {
54             startVoiceRecognition()
55         } else {
56             Toast.makeText(this, "Permissão necessária para uso do microfone.", Toast.LENGTH_SHORT).show()
57         }
58     }
59
60     override fun onCreate(savedInstanceState: Bundle?) {
61         super.onCreate(savedInstanceState)
62         setContentView(R.layout.activity_main)
63         audioHelper = AudioHelper(this)
64
65         val animation = AnimationUtils.loadAnimation(this, R.anim.button_press)
66         findViewById<Button>(R.id.voiceCommandButton).setOnClickListener {
67             it.startAnimation(animation)
68             if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO) == PackageManager.PERMISSION_GRANTED) {
69                 startVoiceRecognition()
70             } else {
71                 requestPermissionLauncher.launch(Manifest.permission.RECORD_AUDIO)
72             }
73         }
74
75         findViewById<Button>(R.id.weatherButton).setOnClickListener {
76             it.startAnimation(animation)
77             executeWeather()
78         }
79
80         findViewById<Button>(R.id.sendMessageButton).setOnClickListener {
81             it.startAnimation(animation)
82             executeMessage()
83         }
84
85         findViewById<Button>(R.id.helpButton).setOnClickListener {
86             it.startAnimation(animation)
87             executeHelp()
88         }
89     }
90
91     private fun startVoiceRecognition() {
92         val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH).apply {
93             putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
94             putExtra(RecognizerIntent.EXTRA_LANGUAGE, "pt-BR")
95         }
96
97         speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this)
98         speechRecognizer.setRecognitionListener(object : RecognitionListener {
99             override fun onReadyForSpeech(params: Bundle?) {}
100             override fun onBeginningOfSpeech() {}
101             override fun onEndOfSpeech() {}
102             override fun onBufferReceived(data: ByteArray?) {}
103             override fun onError(error: Int) {
104                 Toast.makeText(this@MainActivity, "Erro ao reconhecer voz: $error", Toast.LENGTH_SHORT).show()
105             }
106             override fun onResults(results: Bundle?) {
107                 val command = results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION)?.get(0)
108                 handleVoiceCommand(command?.trim() ?: "")
109             }
110             override fun onPartialResults(partialResults: Bundle?) {}
111             override fun onEvent(evtType: Int, params: Bundle?) {}
112         })
113         speechRecognizer.startListening(intent)
114     }
115
116     private fun handleVoiceCommand(command: String) {
117         when {
118             command.contains("notícia", ignoreCase = true) -> executeNews()
119             command.contains("clima", ignoreCase = true) -> executeWeather()
120             command.contains("mensagem", ignoreCase = true) -> executeMessage()
121             command.contains("ajuda", ignoreCase = true) -> executeHelp()
122             else -> Toast.makeText(this, "Comando não reconhecido.", Toast.LENGTH_SHORT).show()
123         }
124     }
125
126     private fun executeNews() {
127         val randomNews = newsList.random()
128         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)) {
129             audioHelper.speak(randomNews)
130         } else {
131             Toast.makeText(this, "Dispositivo de áudio não disponível.", Toast.LENGTH_SHORT).show()
132         }
133     }
134
135     private fun executeWeather() {
136         val randomWeather = weatherList.random()
137         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)) {
138             audioHelper.speak(randomWeather)
139         } else {
140             Toast.makeText(this, "Dispositivo de áudio não disponível.", Toast.LENGTH_SHORT).show()
141         }
142     }
143
144     private fun executeMessage() {
145         val sender = messageList.random()
146         val formattedMessage = "Mensagem de $sender"
147         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)) {
148             audioHelper.speak(formattedMessage)
149         } else {
150             Toast.makeText(this, "Dispositivo de áudio não disponível.", Toast.LENGTH_SHORT).show()
151         }
152     }
153
154     private fun executeHelp() {
155         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)) {
156             audioHelper.speak(helpMessage)
157         } else {
158             Toast.makeText(this, "Dispositivo de áudio não disponível.", Toast.LENGTH_SHORT).show()
159         }
160     }
161
162     override fun onDestroy() {
163         super.onDestroy()
164         speechRecognizer.destroy()
165         audioHelper.release()
166     }
167 }
```

AudioHelper:

```
1 package com.example.listadetanefas
2
3 import android.content.Context
4 import android.content.pm.PackageManager
5 import android.media.AudioDeviceInfo
6 import android.media.AudioManager
7 import android.speech.tts.TextToSpeech
8 import android.util.Log
9 import android.widget.Toast
10 import java.util.Locale
11
12 class AudioHelper(private val context: Context) {
13     private val audioManager = context.getSystemService(Context.AUDIO_SERVICE) as AudioManager
14     private var tts: TextToSpeech? = null
15
16     init {
17         initializeTextToSpeech()
18     }
19
20     private fun initializeTextToSpeech() {
21         tts = TextToSpeech(context) { status: Int ->
22             if (status == TextToSpeech.SUCCESS) {
23                 val result = tts!!.setLanguage(Locale(language = "pt", country = "BR"))
24                 if (result == TextToSpeech.LANG_MISSING_DATA || result == TextToSpeech.LANG_NOT_SUPPORTED) {
25                     Log.e(tag, "TTS", msg = "Idioma não suportado para TTS")
26                     Toast.makeText(context, msg = "Idioma para TTS não suportado", Toast.LENGTH_SHORT)
27                         .show()
28                 } else {
29                     Log.d(tag, "TTS", msg = "TextToSpeech inicializado com sucesso em Português")
30                 }
31             } else {
32                 Log.e(tag, "TTS", msg = "Falha ao inicializar TextToSpeech")
33                 Toast.makeText(context, msg = "Erro ao inicializar TTS", Toast.LENGTH_SHORT).show()
34             }
35         }
36     }
37
38     fun audioOutputAvailable(type: Int): Boolean {
39         if (!context.packageManager.hasSystemFeature(PackageManager.FEATURE_AUDIO_OUTPUT)) {
40             Log.e(tag, "AudioHelper", msg = "Recurso de saída de áudio indisponível.")
41             return false
42         }
43
44         val devices = Array<AudioDeviceInfo?>() = audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS)
45         var isAvailable = false
46         for (device in devices) {
47             if (device.type == type) {
48                 isAvailable = true
49                 break
50             }
51         }
52
53         Log.d(tag, "AudioHelper", msg = "Saída de áudio disponível (tipo): $isAvailable")
54         return isAvailable
55     }
56
57     fun speak(text: String) {
58         if (tts == null) {
59             Log.e(tag, "AudioHelper", msg = "Erro: TTS não inicializado.")
60             Toast.makeText(context, msg = "Erro: TTS não inicializado", Toast.LENGTH_SHORT).show()
61             return
62         }
63         if (tts!!.isSpeaking) {
64             Log.d(tag, "AudioHelper", msg = "TTS está ocupado. Tentando falar novamente...")
65         }
66         Log.d(tag, "AudioHelper", msg = "Falando texto: $text")
67         val result = tts!!.speak(text, TextToSpeech.QUEUE_FLUSH, params = null, utteranceId = null)
68         if (result == TextToSpeech.ERROR) {
69             Log.e(tag, "AudioHelper", msg = "Erro ao reproduzir o áudio.")
70         }
71     }
72
73     fun stopSpeaking() {
74         if (tts != null) {
75             tts!!.stop()
76             Log.d(tag, "AudioHelper", msg = "Parando o TTS.")
77         }
78     }
79
80     fun release() {
81         if (tts != null) {
82             tts!!.shutdown()
83             Log.d(tag, "AudioHelper", msg = "Recursos do TTS liberados.")
84         }
85     }
86
87     fun listAudioDevices() {
88         val devices = Array<AudioDeviceInfo?>() = audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS)
89         for (device in devices) {
90             Log.d(tag, "AudioHelper", msg = "Dispositivo: " + device.productName + ", Tipo: " + device.type)
91         }
92     }
93
94     fun registerAudioDeviceCallback(onDeviceAdded: (Int) -> Unit) {
95         audioManager.registerAudioDeviceCallback(object : AudioDeviceCallback() {
96             override fun onAudioDevicesAdded(addedDevices: Array<AudioDeviceInfo?>) {
97                 super.onAudioDevicesAdded(addedDevices)
98                 for (device in addedDevices) {
99                     Log.d(tag, "AudioHelper", msg = "Dispositivo de áudio conectado: " + device.type)
100                     if (device.type == AudioDeviceInfo.TYPE_BLUETOOTH_A2DP) {
101                         onDeviceAdded(device.type)
102                         Toast.makeText(
103                             context,
104                             msg = "Fone de ouvido Bluetooth conectado.",
105                             Toast.LENGTH_SHORT
106                         ).show()
107                     }
108                 }
109             }
110         })
111     }
112
113     override fun onAudioDevicesRemoved(removedDevices: Array<AudioDeviceInfo?>) {
114         super.onAudioDevicesRemoved(removedDevices)
115         for (device in removedDevices) {
116             Log.d(tag, "AudioHelper", msg = "Dispositivo de áudio removido: " + device.type)
117             if (device.type == AudioDeviceInfo.TYPE_BLUETOOTH_A2DP) {
118                 Toast.makeText(
119                     context,
120                     msg = "Fone de ouvido Bluetooth desconectado.",
121                     Toast.LENGTH_SHORT
122                 ).show()
123             }
124         }
125     }
126
127     if (audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)) {
128         Log.d(tag, "AudioHelper", msg = "Ruído para o alto-falante integrado.")
129         audioManager.mode = AudioManager.MODE_NORMAL
130     }
131 }
132 }
```

Resultado do Visual do App



- Resultados esperados 🌟

Ao concluir esta missão, os alunos terão desenvolvido um aplicativo Wear OS que proporciona uma comunicação eficaz e assistência para funcionários com necessidades especiais. O aplicativo deverá ser capaz de ler mensagens e notificações em voz alta, responder a comandos de voz e fornecer alertas de segurança e instruções através de áudio. Este aplicativo não apenas melhora a eficiência e a comunicação interna na empresa "Doma", mas também demonstra a aplicação prática de tecnologias wearables para criar soluções acessíveis e inclusivas no local de trabalho.