

Desenvolvimento de Aplicativos Móveis

Professor Maurício Buess

mbuess@up.edu.br

<https://github.com/mauriciobuess>

Desenvolvimento Mobile

Introdução ao Desenvolvimento Android:

- Contextualizar sobre a plataforma Android, sua importância no mercado e as principais características da arquitetura Android.

Desenvolvimento Mobile

Introdução ao Desenvolvimento Android:

- História e Evolução do Android
 - Desenvolvido pela Google;
 - Baseado no kernel do Linux;
 - Target: dispositivos móveis com tela sensível ao toque, como smartphones e tablets;
 - É o sistema operacional móvel mais popular do mundo;
 - Domina o mercado global de smartphones e centrais veicular.

Desenvolvimento Mobile

Origens:

- 2003:
 - Inicialmente desenvolvido por Andy Rubin, Rich Miner, Nick Sears, e Chris White.
 - A empresa Android Inc. foi fundada por eles em Palo Alto, Califórnia, com o objetivo de desenvolver um sistema operacional para câmeras digitais.
 - Após pouco tempo o foco do projeto mudou para dispositivos móveis.

Desenvolvimento Mobile

Origens:

- 2005:
 - A Google adquiriu a Android Inc.;
 - Rubin, Miner, Sears e White passaram a trabalhar para a Google.
 - Google decidiu usar o Android como a base para um sistema operacional móvel que competiria com outros sistemas, como o Symbian da Nokia e o Windows Mobile da Microsoft.

Desenvolvimento Mobile

Origens:

- 2007:
 - Anuncio oficial do Android, junto com a fundação da Open Handset Alliance (OHA), um consórcio de empresas de tecnologia e telefonia móvel que inclui nomes como HTC, Motorola, Qualcomm, Samsung, LG, e outros.
 - O Android foi lançado como uma plataforma de código aberto, permitindo que fabricantes e desenvolvedores contribuíssem para o seu desenvolvimento.

Desenvolvimento Mobile

Origens:

- 2008:
 - O primeiro dispositivo Android, o HTC Dream (também conhecido como T-Mobile G1), foi lançado em setembro de 2008.
 - Este dispositivo apresentava uma tela sensível ao toque, teclado deslizante, e um sistema de navegação por trackball.
 - O sistema operacional oferecia recursos como integração com serviços da Google (como Gmail e Google Maps), um navegador web, e a Android Market (atualmente Google Play Store).

Desenvolvimento Mobile

Origens:

- Desde seu lançamento, o Android passou por várias atualizações significativas, cada uma com um nome de sobremesa em ordem alfabética até a versão 9.0, refletindo a cultura lúdica da equipe de desenvolvimento.

Desenvolvimento Mobile

- 2018: Android 9.0 (Pie): Trouxe navegação por gestos e um foco maior em inteligência artificial para melhorar a experiência do usuário.
- 2019: Android 10: Abandonou o nome de sobremesas e trouxe melhorias significativas na privacidade, com um modo escuro em todo o sistema e permissões aprimoradas.
- 2020: Android 11: Introduziu melhorias nas notificações, bolhas de chat, e controles aprimorados de privacidade.
- 2021: Android 12: Com a nova linguagem de design "Material You", trouxe uma personalização visual mais profunda com cores dinâmicas e uma interface mais fluida e amigável.
- 2022: Android 13: Focado em segurança, privacidade, e novas opções de personalização. Introduziu mudanças como a permissão de notificações por parte de apps.

Desenvolvimento Mobile

Impacto e Popularidade:

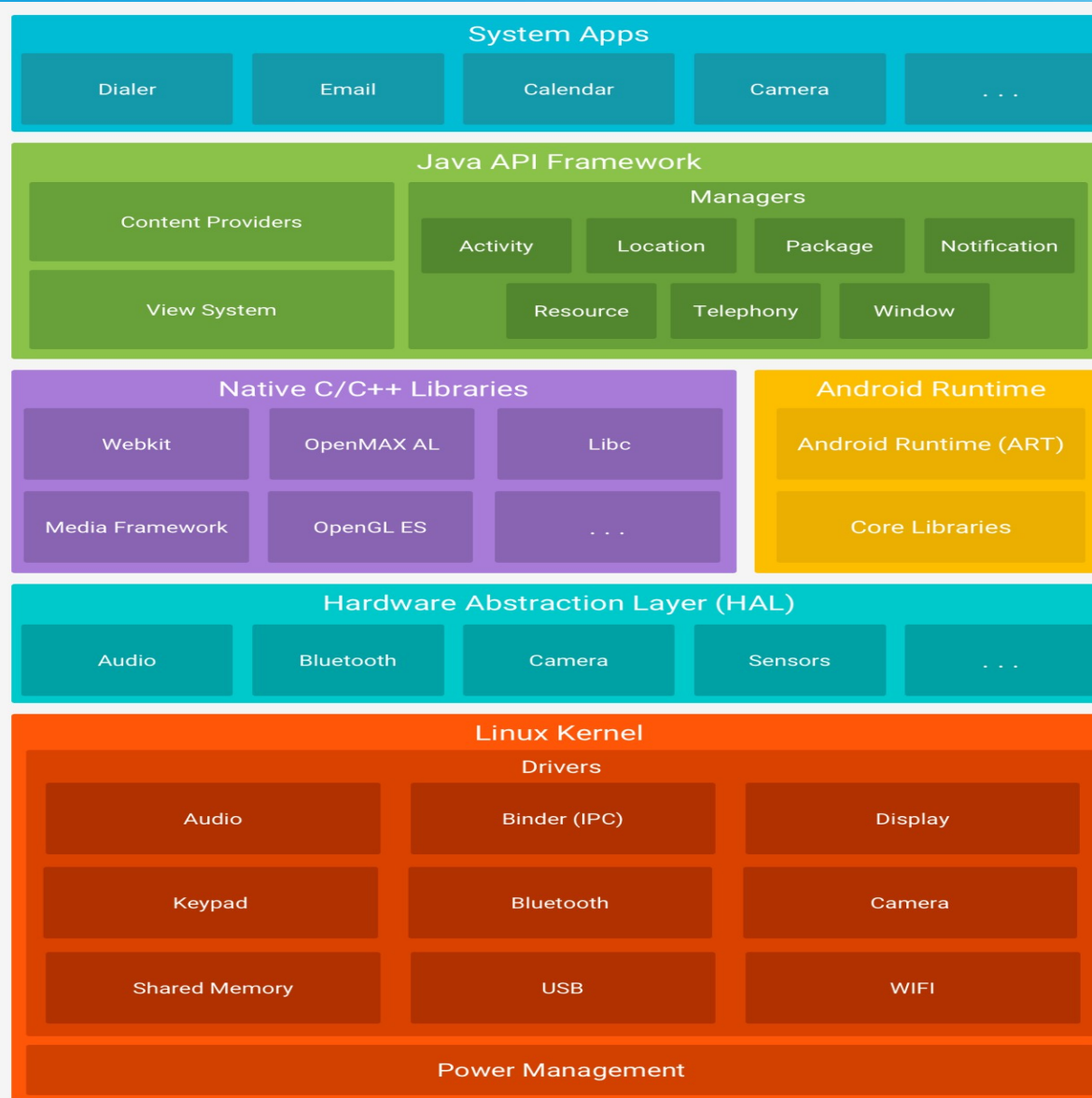
- Android se tornou o sistema operacional móvel mais popular do mundo, com bilhões de dispositivos ativos em todo o mundo. Sua flexibilidade, natureza de código aberto e ampla adoção por fabricantes de hardware foram fatores-chave para o seu sucesso.
- Atualmente, o Android não está apenas presente em smartphones, mas também em uma variedade de outros dispositivos, incluindo tablets, smart TVs, wearables (como smartwatches), e até mesmo automóveis.

Conclusões:

- A evolução do Android é uma história de inovação acelerada, contínua, focada na adaptação às necessidades dos usuários através de uma plataforma que se tornou indispensável na vida moderna;
- Desde suas versões iniciais até seu status atual como líder global, o Android continua a moldar a forma como interagimos com a tecnologia;
- Novos paradigmas e frameworks facilitam a programação;
- Busca implícita para a unificação de plataformas.

Arquitetura do Android:

- É composta por várias camadas que interagem entre si para fornecer uma plataforma robusta e flexível para o desenvolvimento de aplicativos móveis.
- As camadas incluem o Linux Kernel, Android Runtime, Application Framework, e as Aplicações.
- Arquitetura complexa porém muito funcional



Desenvolvimento Mobile

Linux Kernel:

- Fundação do Sistema: O Android é construído sobre o Linux Kernel, que serve como base para todo o sistema operacional. Esta camada interage diretamente com o hardware do dispositivo.
- Gerenciamento de Recursos: O kernel é responsável por gerenciar todos os recursos do sistema, como memória, processos, dispositivos de entrada/saída (teclado, touchscreen, etc.), drivers de hardware, redes, e muito mais.
- Segurança e Abstração de Hardware: O Linux Kernel também fornece uma camada de segurança, incluindo a implementação de modelos de permissões e isolamento entre processos. Ele abstrai as complexidades do hardware para que o Android possa ser executado em uma variedade de dispositivos sem modificações significativas.

Desenvolvimento Mobile

Android Runtime (ART):

- Máquina Virtual ART: A Android Runtime (ART) é o componente responsável por executar aplicativos Android. Ela substituiu a antiga Dalvik Virtual Machine nas versões mais recentes do Android (a partir do Android 5.0 Lollipop).
- Compilação Ahead-of-Time (AOT): Com o ART, os aplicativos são compilados durante a instalação no dispositivo, em vez de serem interpretados em tempo de execução. Isso melhora significativamente o desempenho e a eficiência dos aplicativos.
- Bibliotecas Centrais: A ART inclui um conjunto de bibliotecas centrais que fornecem funcionalidades essenciais para os aplicativos Android. Essas bibliotecas são escritas em Java e C/C++ e fornecem serviços como coleta de lixo, manipulação de strings, e operações matemáticas.

Desenvolvimento Mobile

Application Framework:

- Camada de Abstração: O Application Framework oferece um conjunto de APIs que os desenvolvedores podem usar para construir aplicativos Android. Ele abstrai os detalhes complexos do sistema e fornece ferramentas para interagir com o hardware, o sistema operacional e outros aplicativos.
- API - Application Programming Interface (Interface de Programação de Aplicação)

Desenvolvimento Mobile

Principais Componentes:

- Activity Manager: Gerencia o ciclo de vida das atividades e mantém o controle de quais atividades estão em execução.
- Content Providers: Permitem que aplicativos compartilhem dados entre si, utilizando um modelo padronizado.
- Resource Manager: Gerencia recursos não-código, como strings, layouts e gráficos.
- Notification Manager: Lida com a exibição de notificações na barra de status.
- View System: Um conjunto de componentes UI que permitem construir a interface gráfica dos aplicativos.
- Package Manager: Gerencia a instalação, atualização e remoção de aplicativos.

Desenvolvimento Mobile

Aplicações:

- Camada do Usuário: Esta é a camada onde os usuários interagem diretamente com o sistema Android. Inclui tanto os aplicativos pré-instalados (como Telefone, Mensagens, Navegador, etc.) quanto os aplicativos de terceiros baixados da Google Play Store.
- Sandboxing e Isolamento: Cada aplicação Android é executada em sua própria instância da ART e é isolada das outras aplicações. Isso melhora a segurança e a estabilidade, impedindo que um aplicativo comprometa o funcionamento de outro.

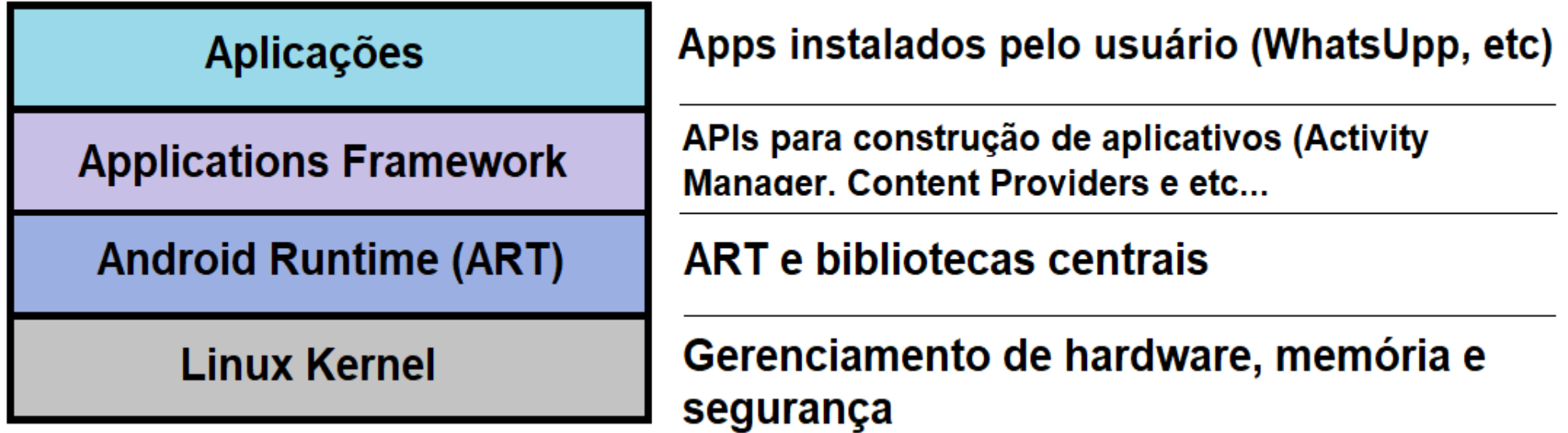
Desenvolvimento Mobile

Componentes de Aplicativos:

- Activities: Representam uma única tela com uma interface de usuário, semelhante a uma "página" em um aplicativo.
- Services: Componentes que executam operações em segundo plano sem interface de usuário.
- Broadcast Receivers: Permitem que os aplicativos respondam a mensagens ou eventos do sistema.
- Content Providers: Gerenciam o acesso a um banco de dados centralizado ou a outros armazenamentos de dados compartilhados entre aplicativos.

Desenvolvimento Mobile

Resumo Arquitetura:



Conclusão:

- A arquitetura do Android é projetada para ser modular, flexível, e segura. Cada camada desempenha um papel crucial no funcionamento do sistema, desde a interação com o hardware até a execução de aplicativos. Ao entender essas camadas, os desenvolvedores podem criar aplicativos mais eficientes, seguros, e otimizados para uma ampla gama de dispositivos Android.

Instalação e Configuração do Android Studio:

- Requisitos mínimos:
 - Sistema Operacional:
 - Windows: Windows 10/11 (64-bit)
 - macOS: Mac OS X 10.14 (Mojave) ou superior
 - Linux: Qualquer distribuição moderna de 64 bits com o ambiente gráfico GNOME ou KDE
 - Memória: Pelo menos 8 GB de RAM (16 GB recomendado)
 - Espaço em Disco: Pelo menos 4 GB de espaço disponível, mais 1 GB para o cache do Android Emulator
 - Resolução de Tela: 1280 x 800 (mínimo)

Desenvolvimento Mobile

Instalação e Configuração do Android Studio:

- Download do Android Studio
- Acesse o site oficial do Android Studio:
 - <https://developer.android.com/studio>
- Clique no botão “Download Android Studio”.
- Revise e aceite os Termos e Condições de uso.

Instalação e Configuração do Android Studio:

- Instalando o Android Studio no Windows:
 - Abra o instalador baixado (.exe).
 - Siga as instruções na tela, aceitando as configurações padrão recomendadas.
 - Escolha os componentes adicionais que deseja instalar (como o Android Virtual Device (AVD)).
 - Clique em “Install” para iniciar a instalação.

Configuração Inicial do Android Studio:

- Quando o Android Studio é executado pela primeira vez, ele inicia o "Android Studio Setup Wizard", que o guiará pelas etapas iniciais de configuração:
- Bem-vindo ao Android Studio:
 - Clique em "Next" para iniciar a configuração.
- Configuração do Tipo de Instalação:
 - Escolha entre a instalação padrão ("Standard") ou personalizada ("Custom").
 - A instalação padrão é recomendada para iniciantes, pois configura automaticamente as opções mais comuns.
 - A instalação personalizada permite configurar detalhes como o tema (Light ou Dark) e a localização do SDK.

Download do Android SDK:

- Se você optou pela instalação padrão, o wizard fará o download do Android SDK automaticamente.
- Se você escolheu a instalação personalizada, poderá selecionar qual versão do SDK deseja instalar. É recomendado instalar a versão mais recente, além de qualquer versão adicional que possa ser necessária para compatibilidade.
- **Android SDK ou Kit de Desenvolvimento de Software para Android é um pacote com diversas ferramentas utilizadas pelo Android Studio e pelos desenvolvedores Android, incluindo componentes como o SDK Tools, Build Tools e o Platform Tools.**

Desenvolvimento Mobile

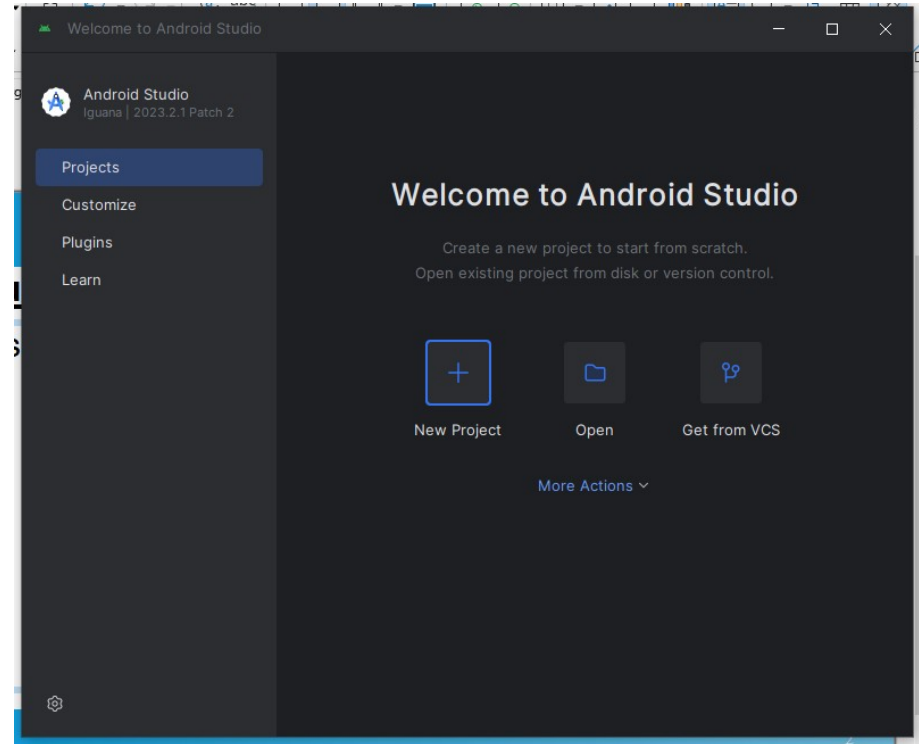
Configuração do Android Virtual Device (AVD):

- O wizard oferecerá a opção de configurar o Android Virtual Device (emulador), permitindo testar aplicativos em uma variedade de dispositivos virtuais.
- Selecione um dispositivo padrão para o emulador (como o Pixel 5) e baixe as imagens de sistema necessárias.
- Conclusão da Instalação:
 - Após a configuração do SDK e do AVD, clique em “Finish” para concluir o setup.

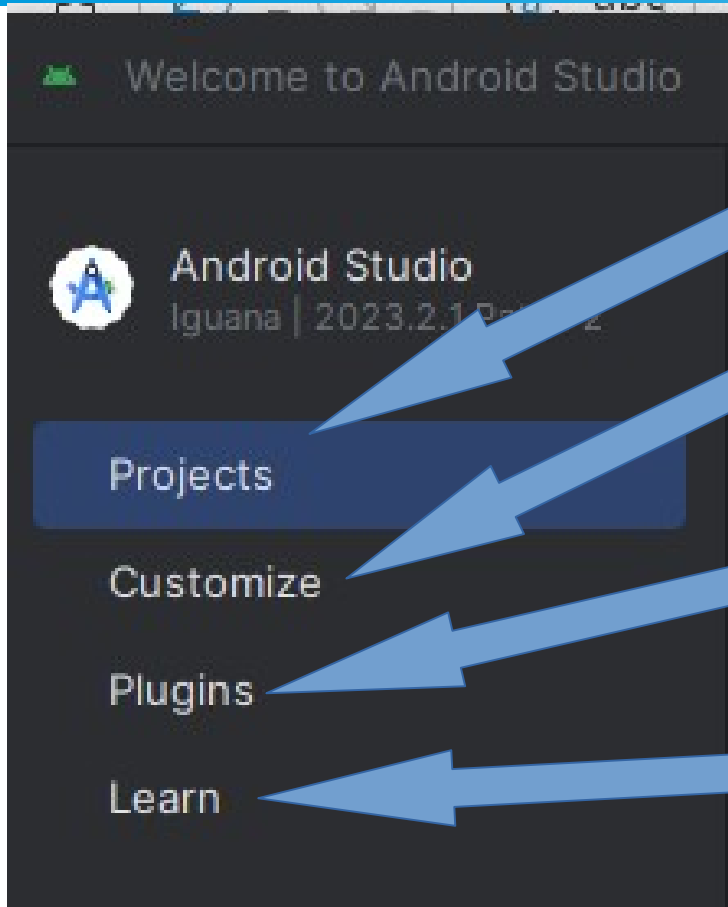
Desenvolvimento Mobile

Introdução ao Android Studio

- Acessar o Android Studio:



Desenvolvimento Mobile



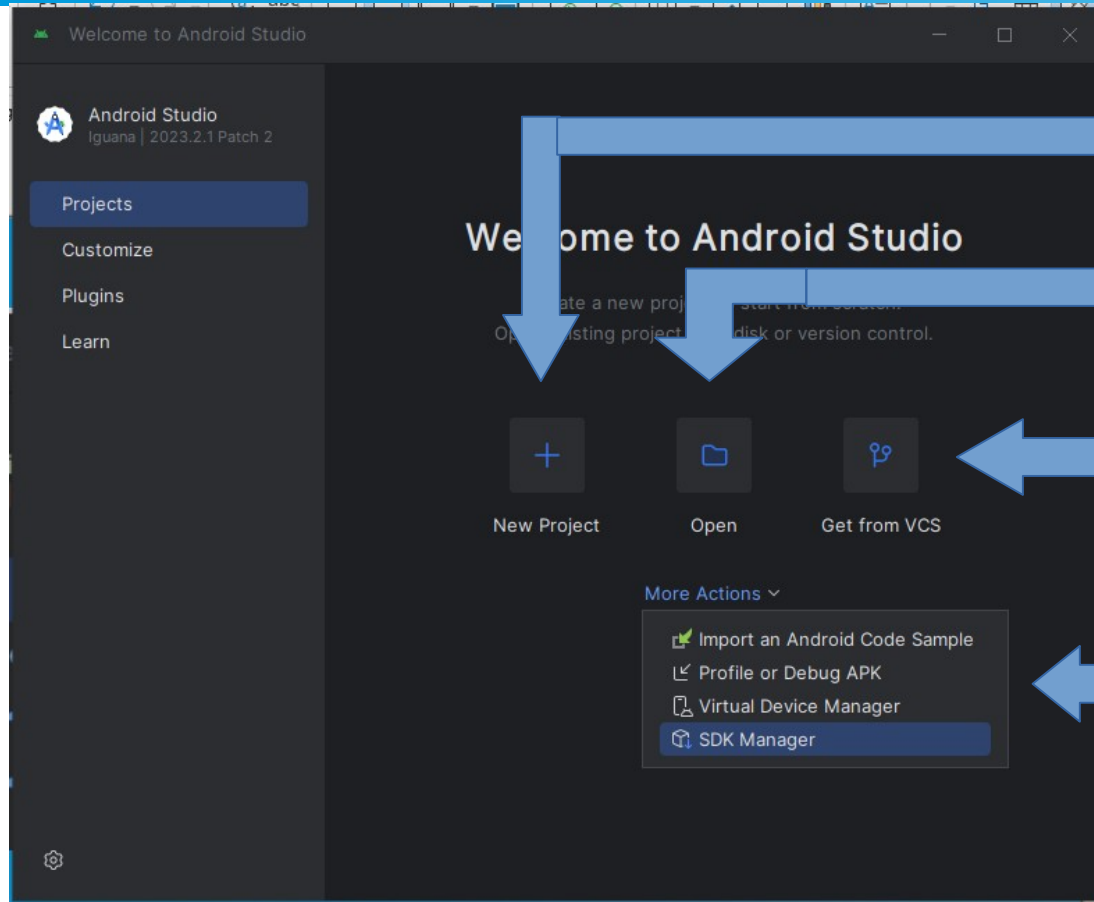
Apresenta os projetos já trabalhados

Opções de configurações rápidas da IDE

- Pesquisa e instalação de plug-ins;
- Atualização e novidades dos plug-ins Instalados;
- Configurações gerais para os plug-ins

- Comunidades Android Studio

Desenvolvimento Mobile



Cria projeto

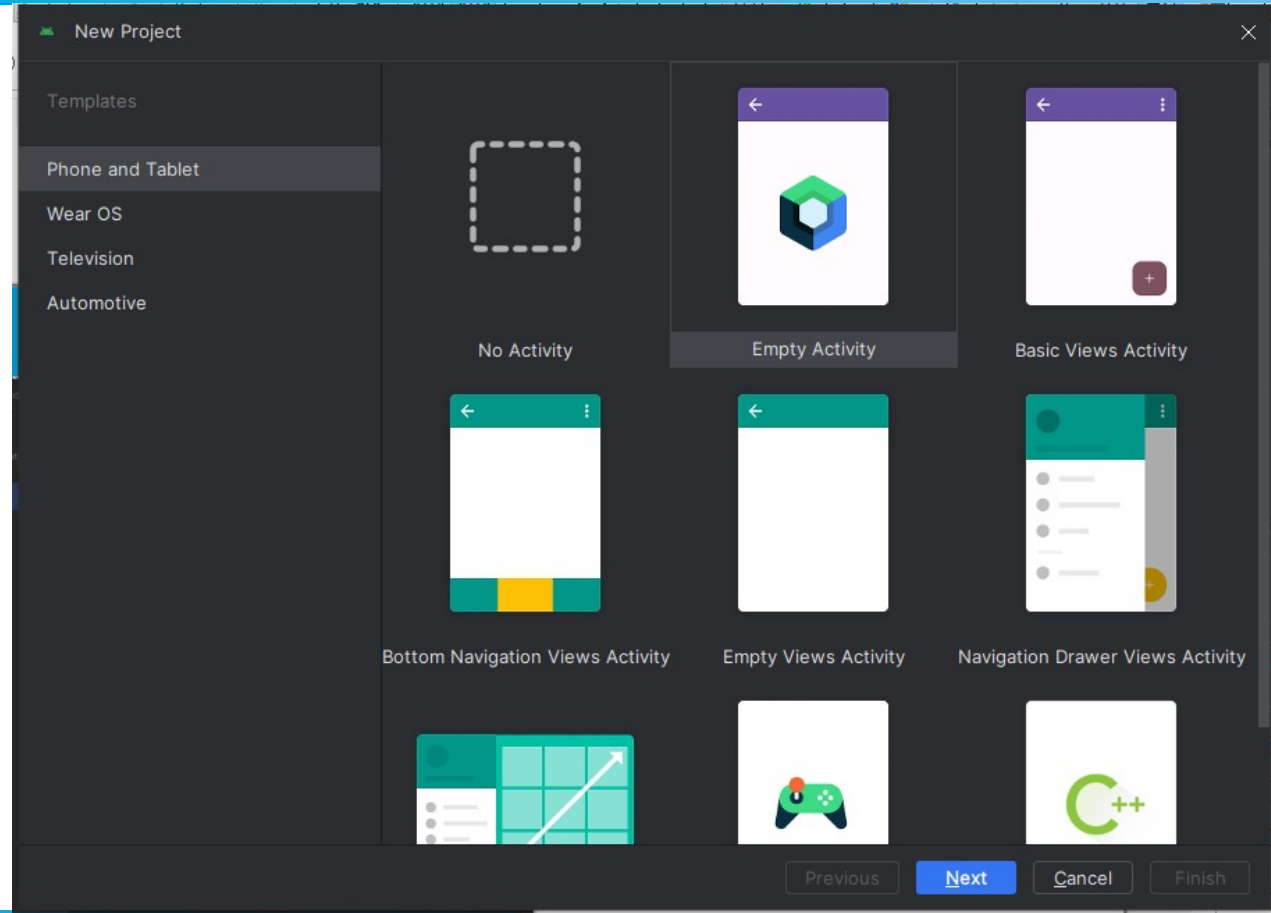
Busca projetos em pastas

Configuração versionadores

Opções auxiliares

Pressione [+] para criar um novo projeto

Desenvolvimento Mobile



Escolha
Empty
Activity

Desenvolvimento Mobile

New Project

Empty Activity

Create a new empty activity with Jetpack Compose

Name: My Application

Package name: com.ti4all.myapplication

Save location: ::\Mauricio\Fapi\Fapi2024\Desenv_Web_Mobile_202401\20240429\MyApplication

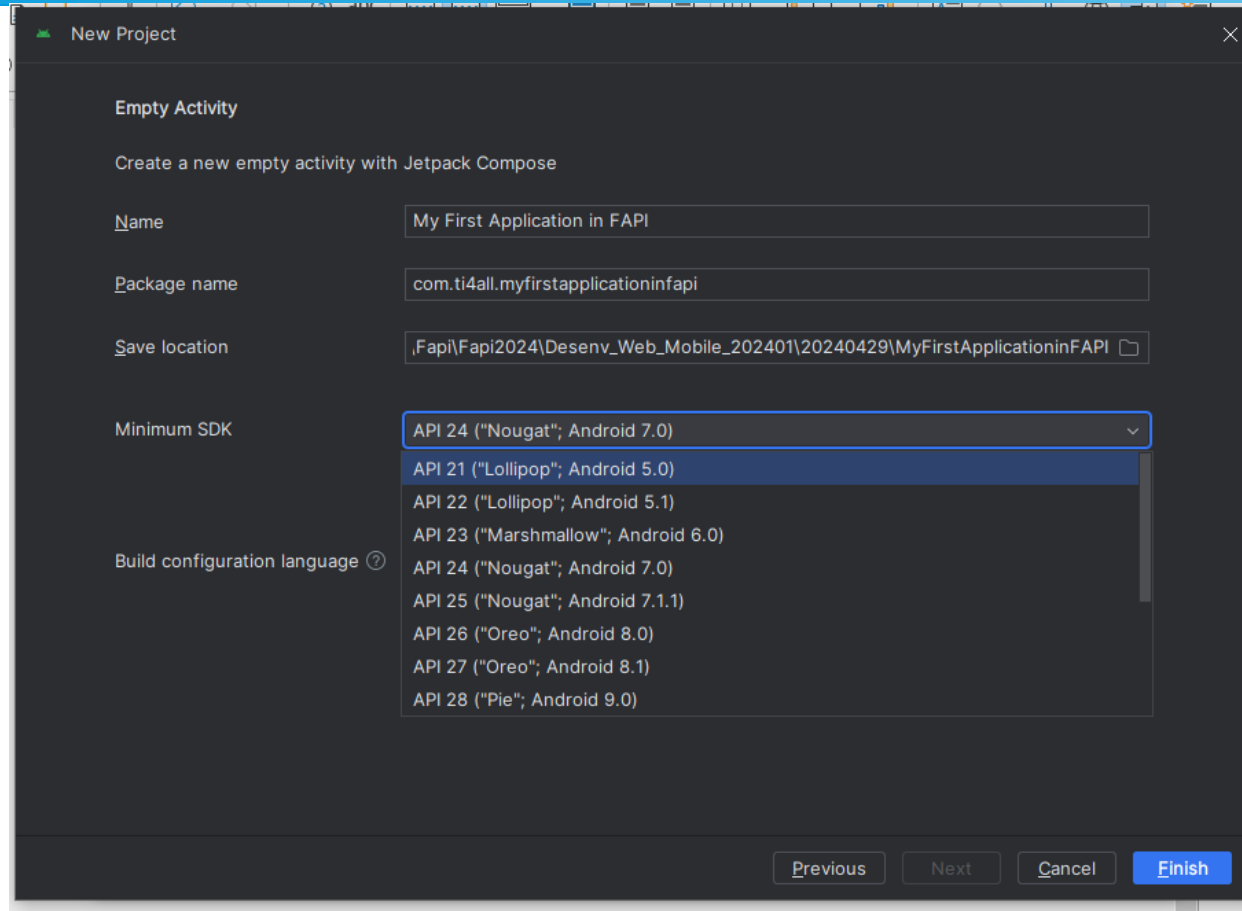
Minimum SDK: API 24 ("Nougat"; Android 7.0)

i Your app will run on approximately 96,3% of devices.
[Help me choose](#)

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

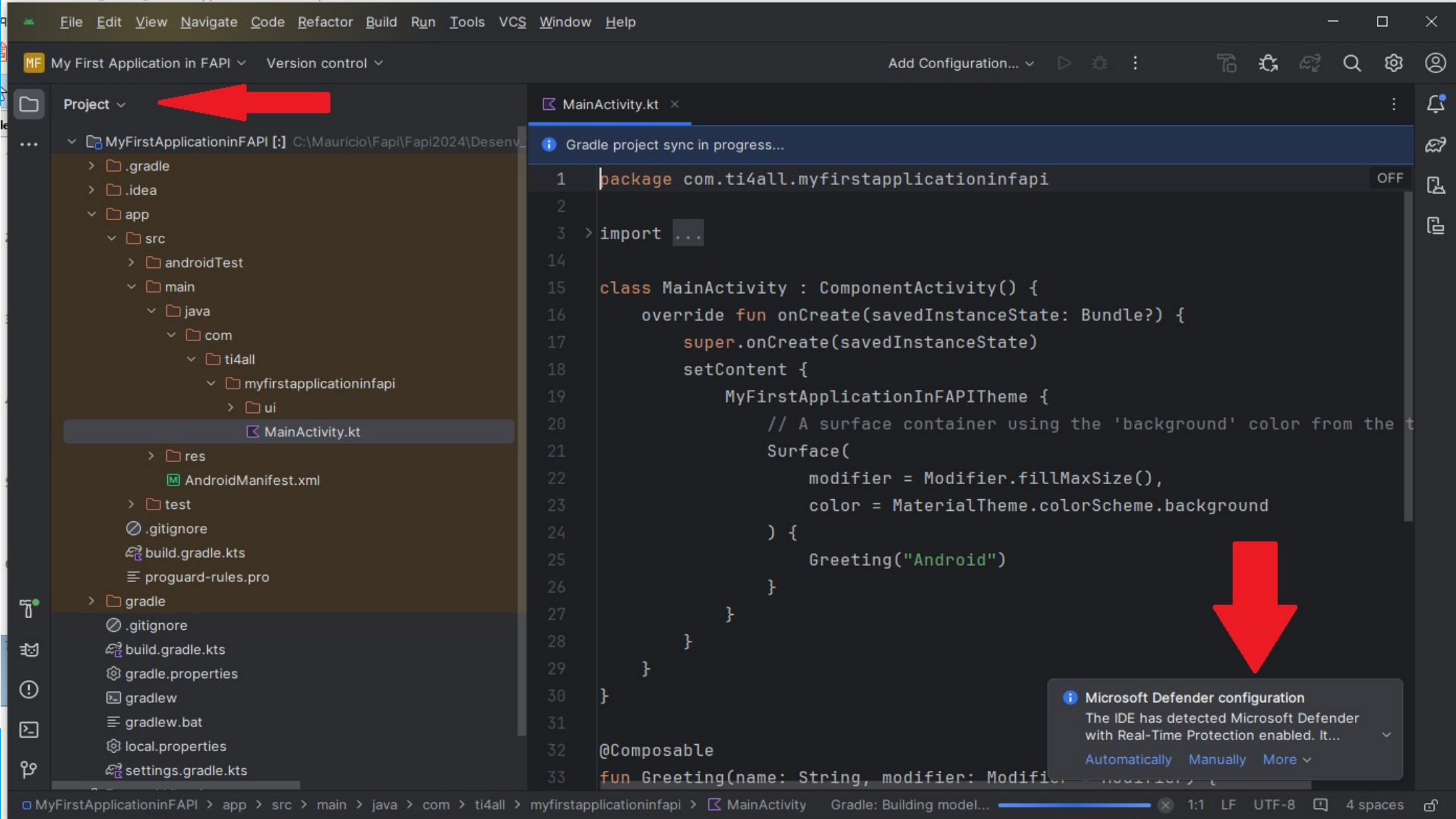
Desenvolvimento Mobile



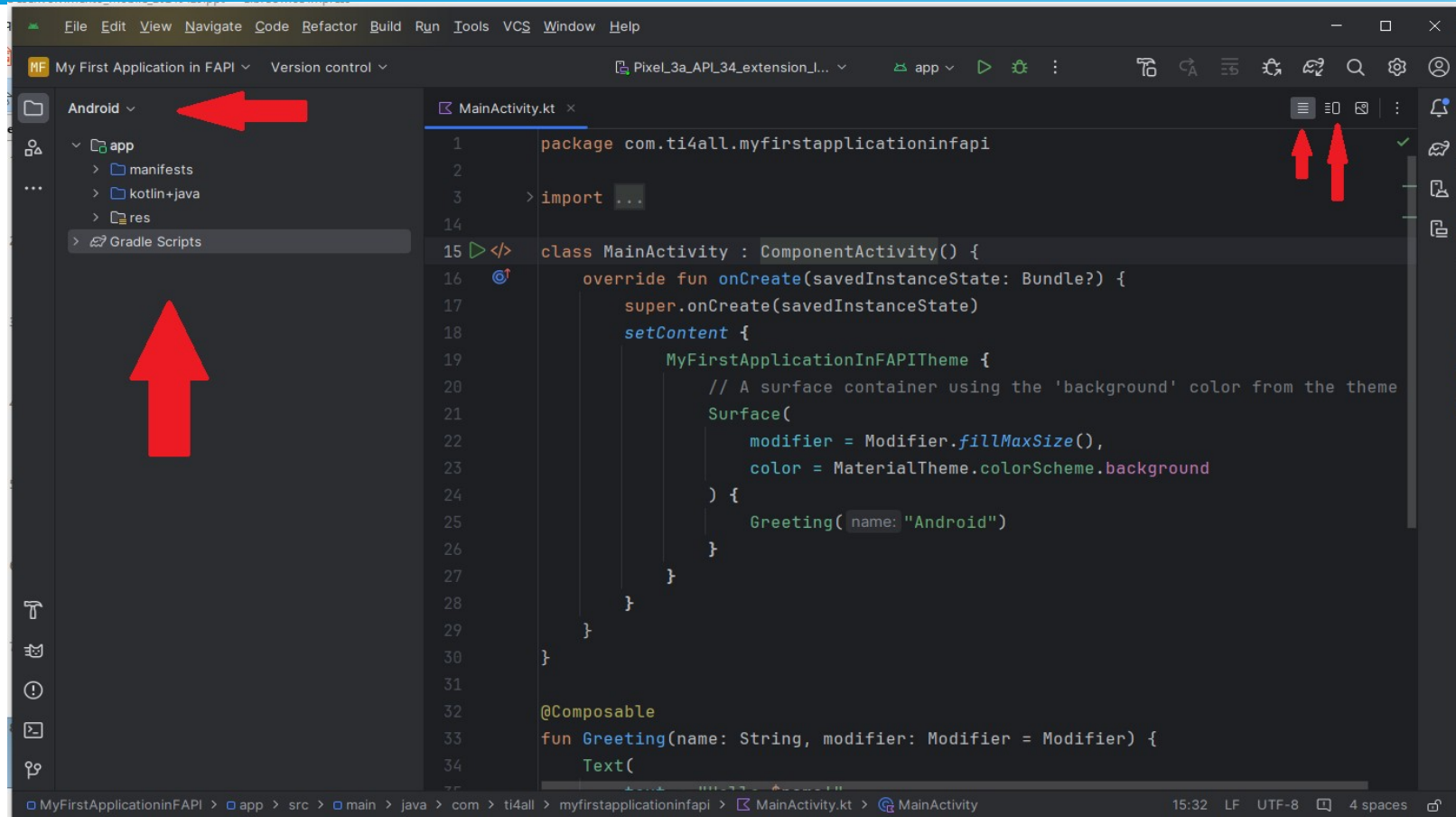
Selecione o
“minimum SDK”

SDK – Software
Development Kit

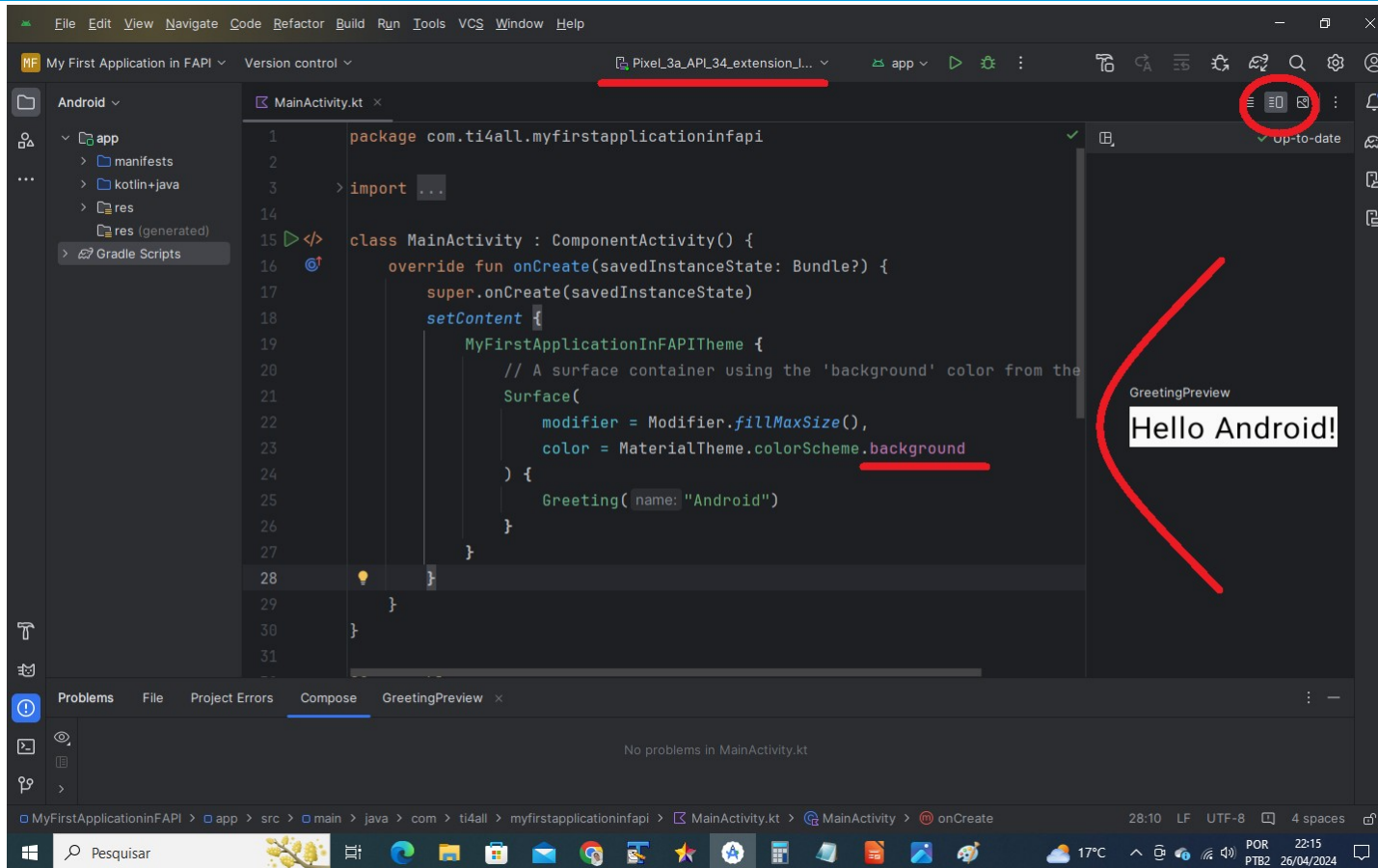
Minimum SDK
definirá a
compatibilidade da
aplicação com os
dispositivos
Android existentes



Desenvolvimento Mobile



Desenvolvimento Mobile



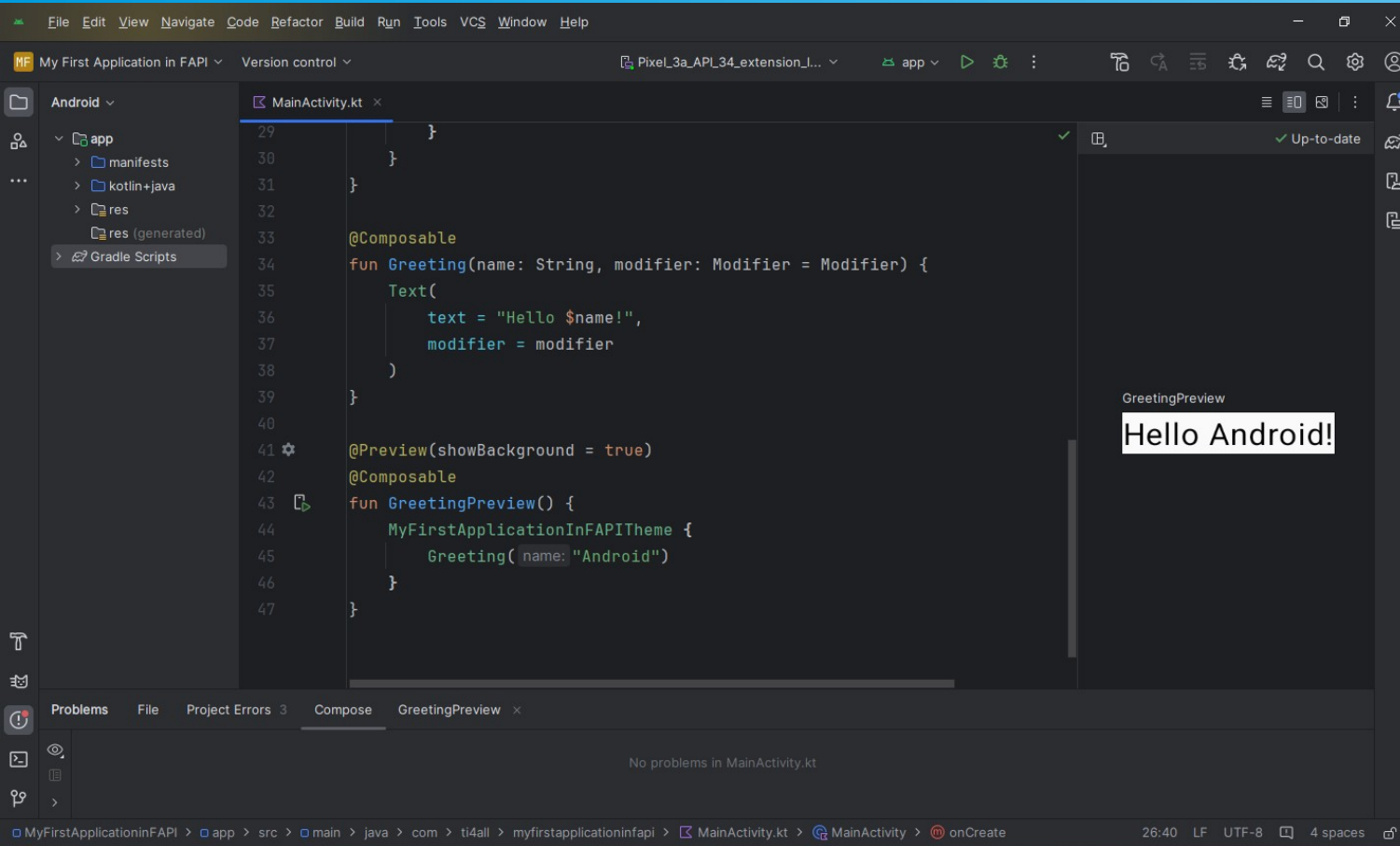
GreetingPreview

Permite visualização prévia da UI em vários dispositivos

OnCreate() → entrada para o APP chamando outras funções para construir a UI.

setContent() junto da onCreate() é usada para definir seu layout por meio de funções que podem ser compostas.

Desenvolvimento Mobile

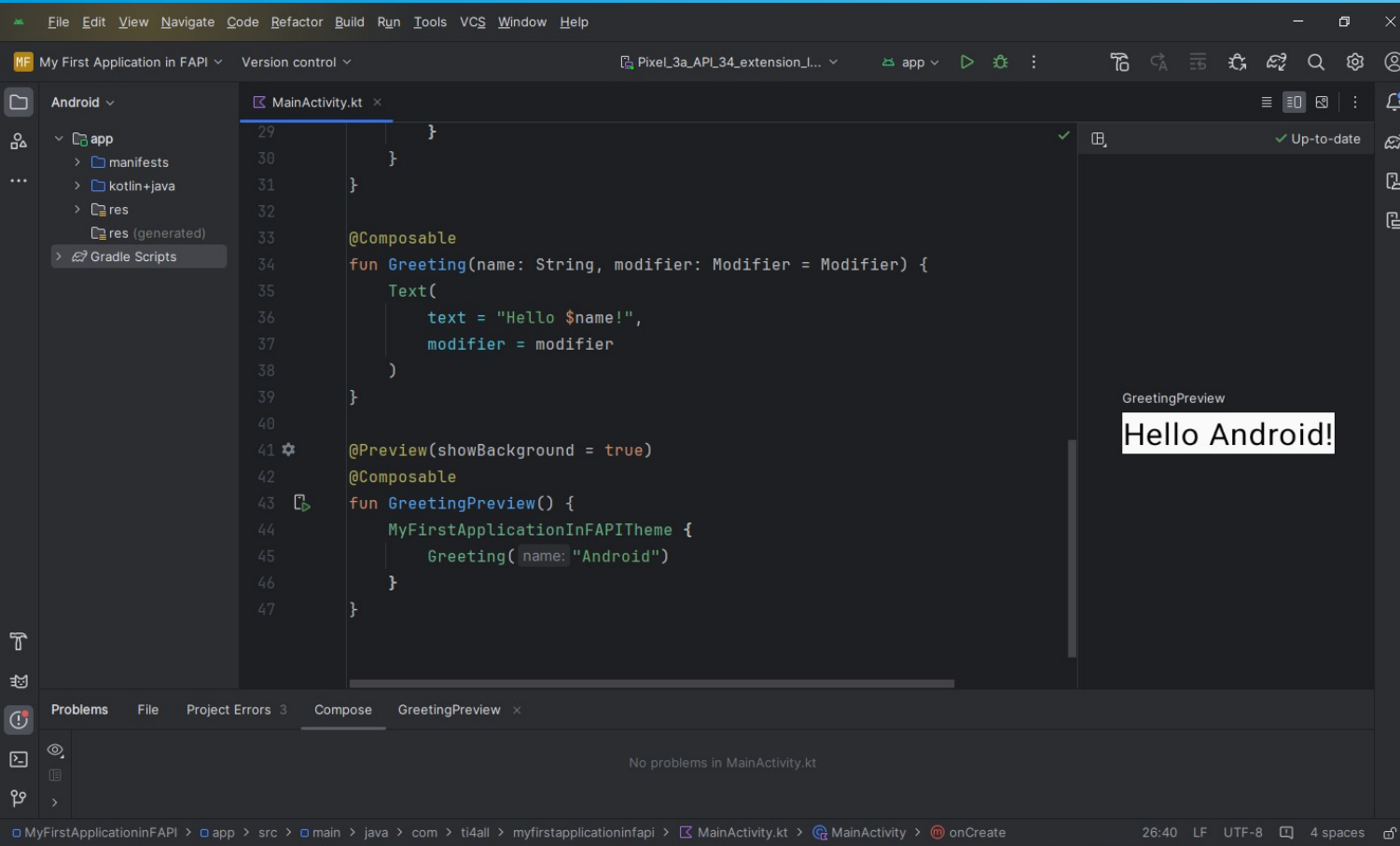


@Composable

As funções compostas são como funções normais, com algumas diferenças:

- os nomes das funções iniciam com maiúsculos
- anotação `@Composable` antes da função
- as funções `@Composable` não podem retornar nada.

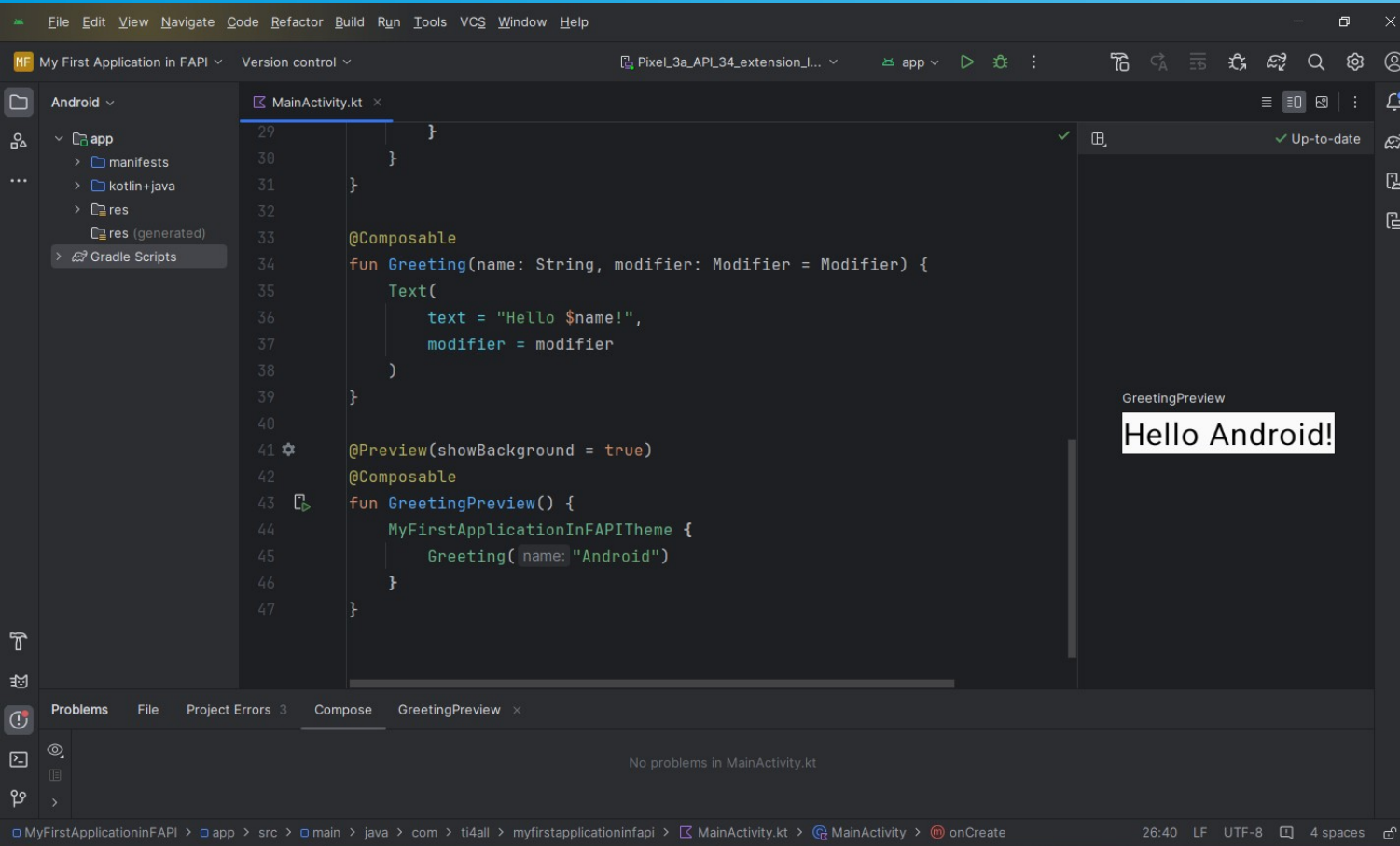
Desenvolvimento Mobile



Todas as funções marcadas com a anotação `@Composable` podem ser chamadas a partir da função `setContent()` ou de outras funções Composable.

A anotação informa ao compilador Kotlin que essa função é usada pelo Jetpack Compose para gerar a IU.

Desenvolvimento Mobile





A função `Greeting()` função `@Composable` recebe algumas informações e gera o que é mostrado na tela.

`GreetingPreview()` mostra a aparência do elemento composto sem criar o APP.

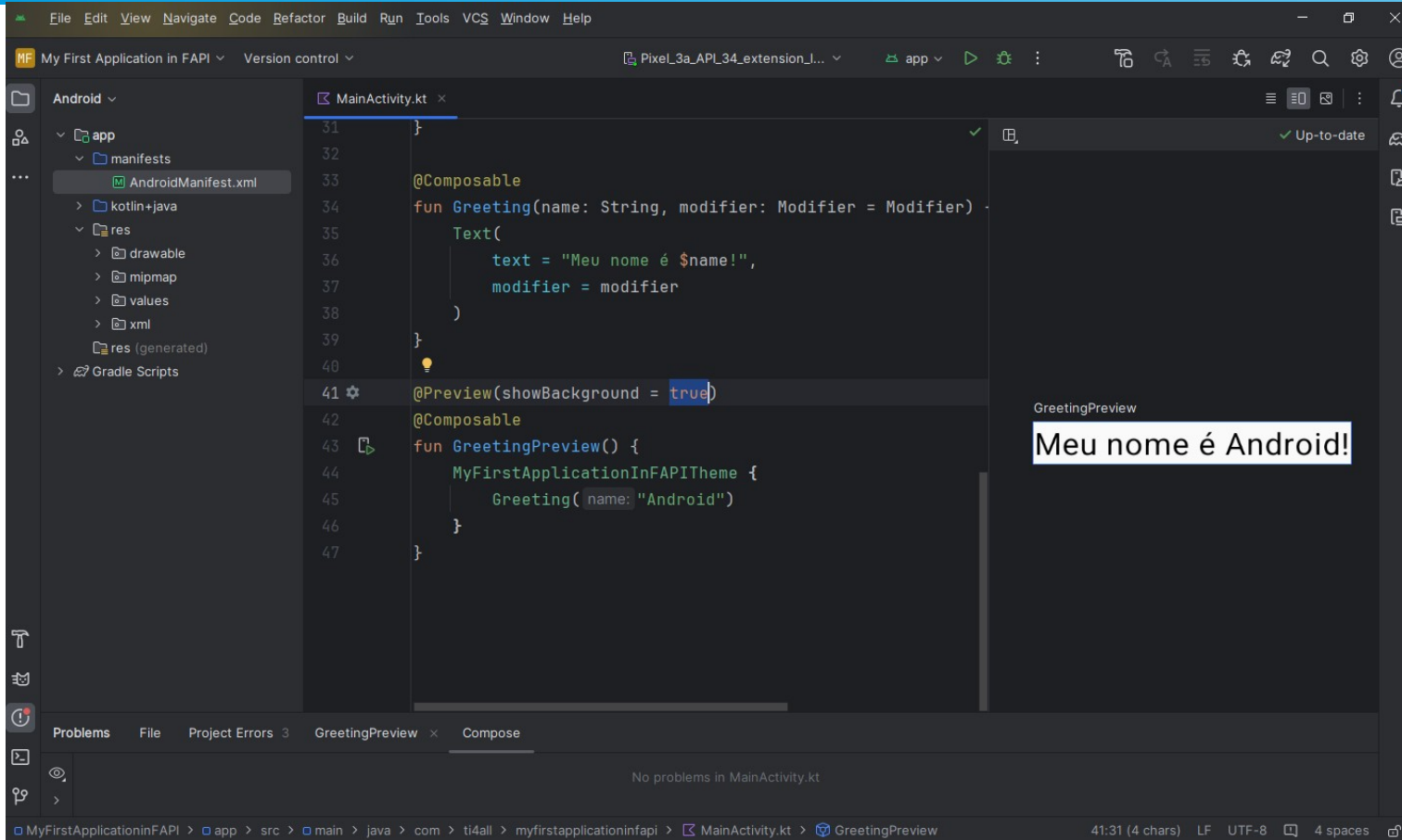
Ativação da visualização de um elemento: anote `@Composable` e `@Preview`.

Desenvolvimento Mobile

```
40
41  @Preview(showBackground = true)
42 @Composable
43  fun GreetingPreview() {
44     MyFirstApplicationInFAPITheme {
45         Greeting(name: "Android")
46     }
47 }
```

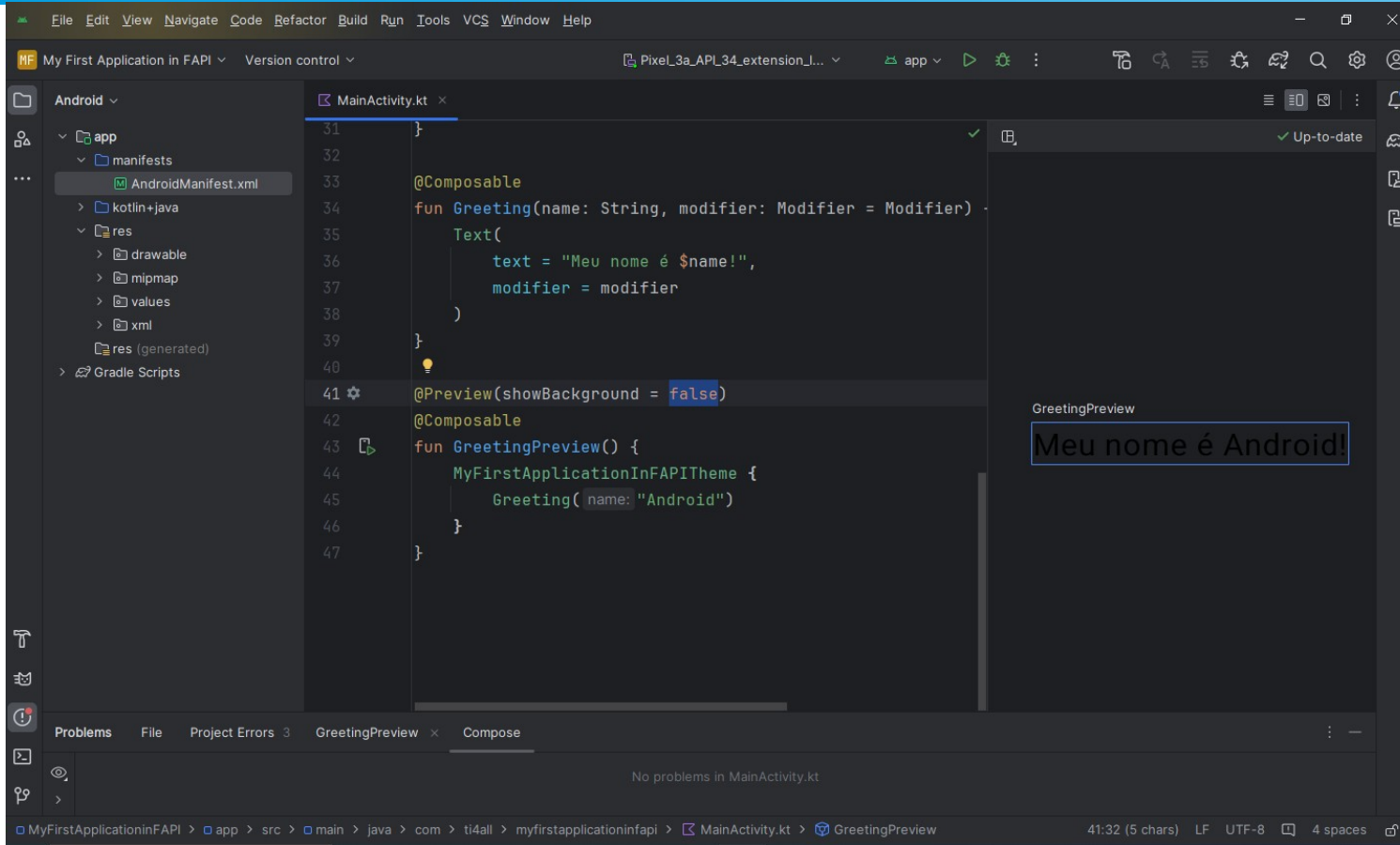
@Preview recebe um parâmetro chamado showBackground. Se showBackground for verdadeiro adicionará um plano de fundo à visualização.

Desenvolvimento Mobile



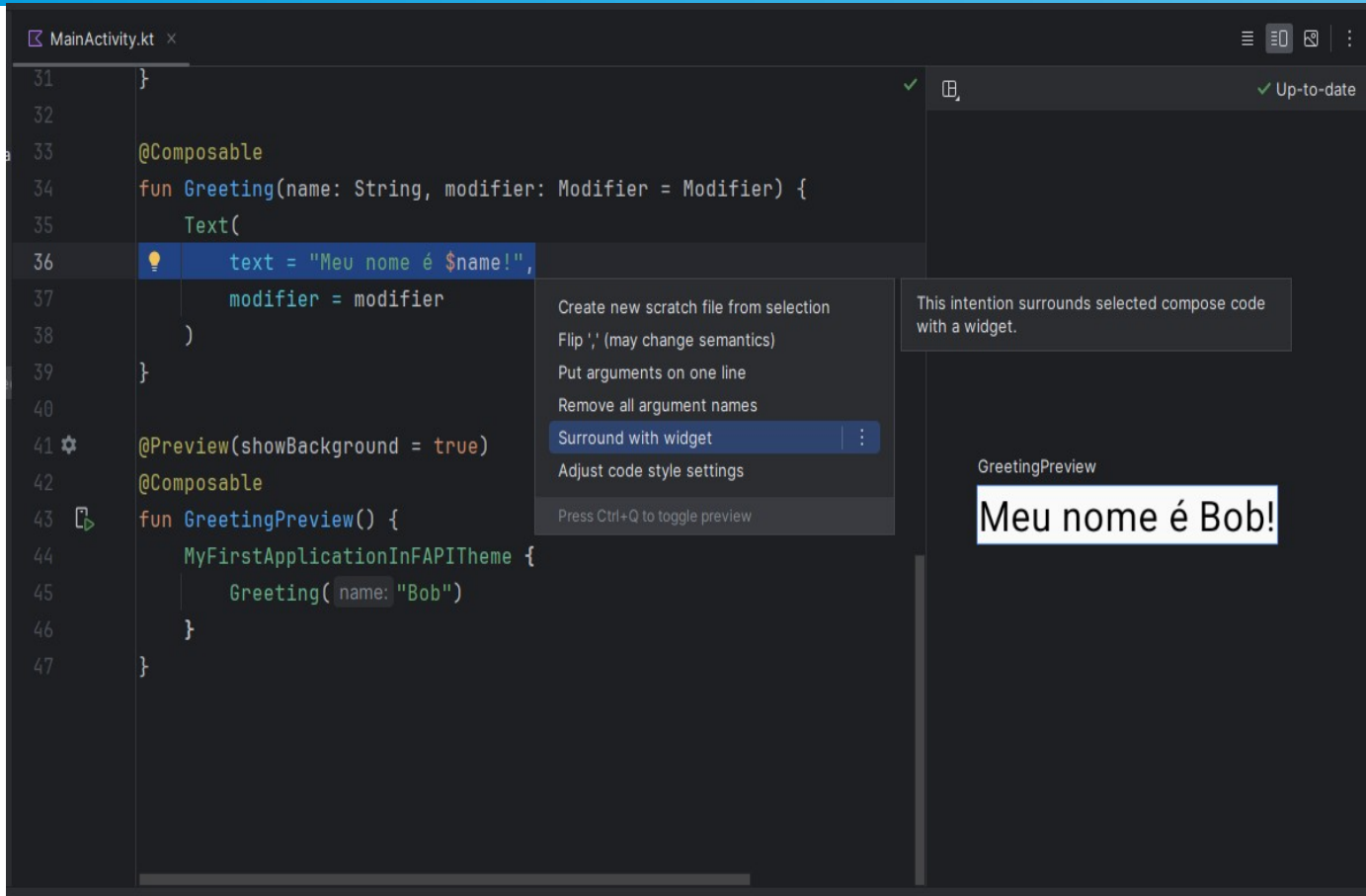
O parâmetro
showBackground
Do @Preview
tem o valor igual
a "true".

Desenvolvimento Mobile



O parâmetro `showBackground` do `@Preview` tem o valor igual a "false".

Desenvolvimento Mobile



Acionar a "superfície" do componente.

Superfície → contêiner que representa uma seção da IU que pode ter a aparência alterada, como a cor de fundo ou a borda.

Selecione a linha e [Alt]+[Enter]

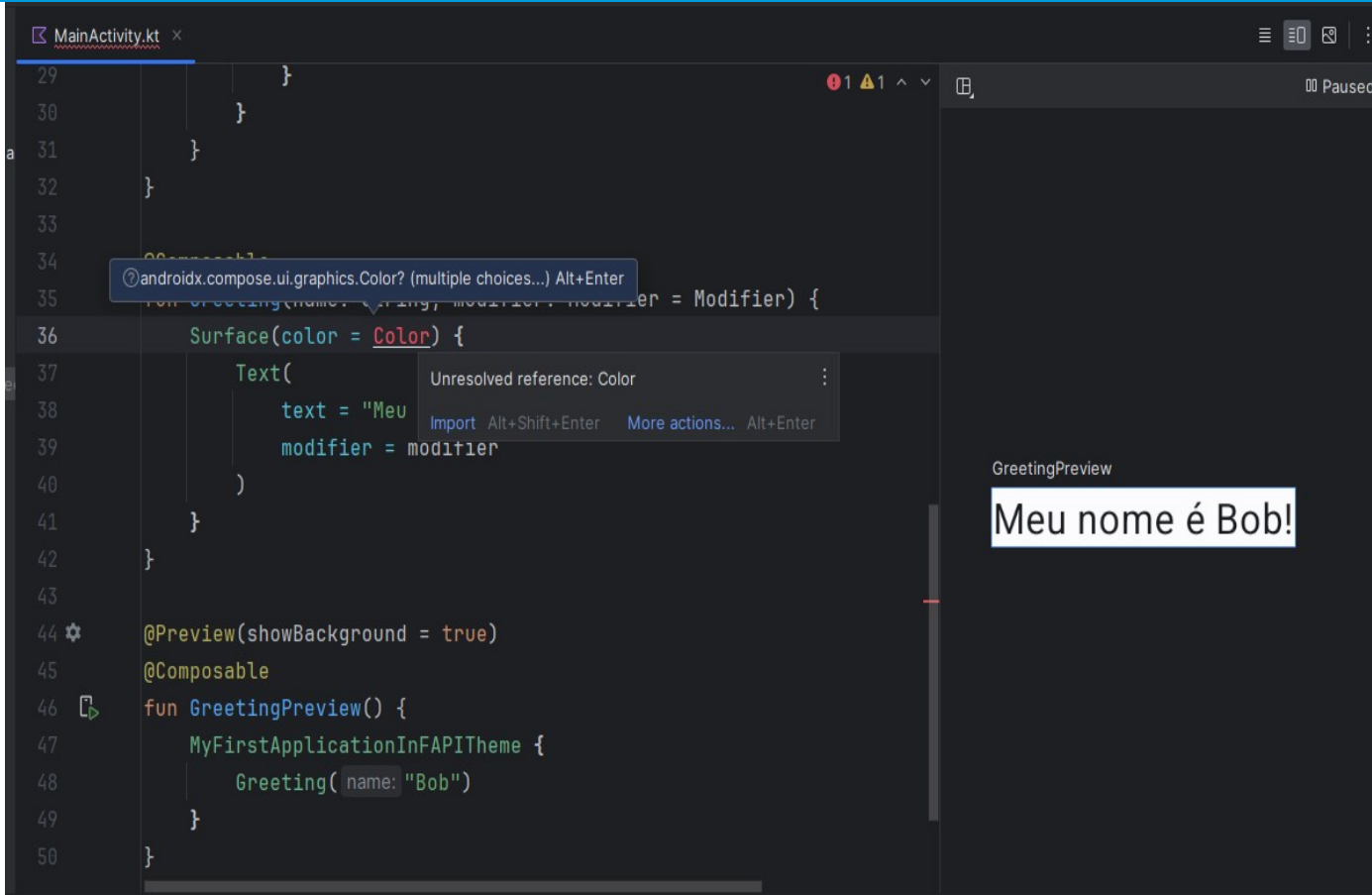
Escolha "Surround with widget".

Desenvolvimento Mobile

```
35     @Composable
36     fun Greeting(name: String, modifier: Modifier = Modifier) {
37         Surface(color = Color) {
38             Text(
39                 text = "Meu nome é $name!",
40                 modifier = modifier
41             )
42         }
43     }
```

- Repare que após a inclusão da biblioteca `androidx.compose.ui.graphics.Color`, `Color` ficou grifada em vermelho;
- Isso significa que ainda está faltando algum parâmetro ou método;
- Vá para o final da palavra `Color` e pressione “.” : `Surface(color = Color.Green)`

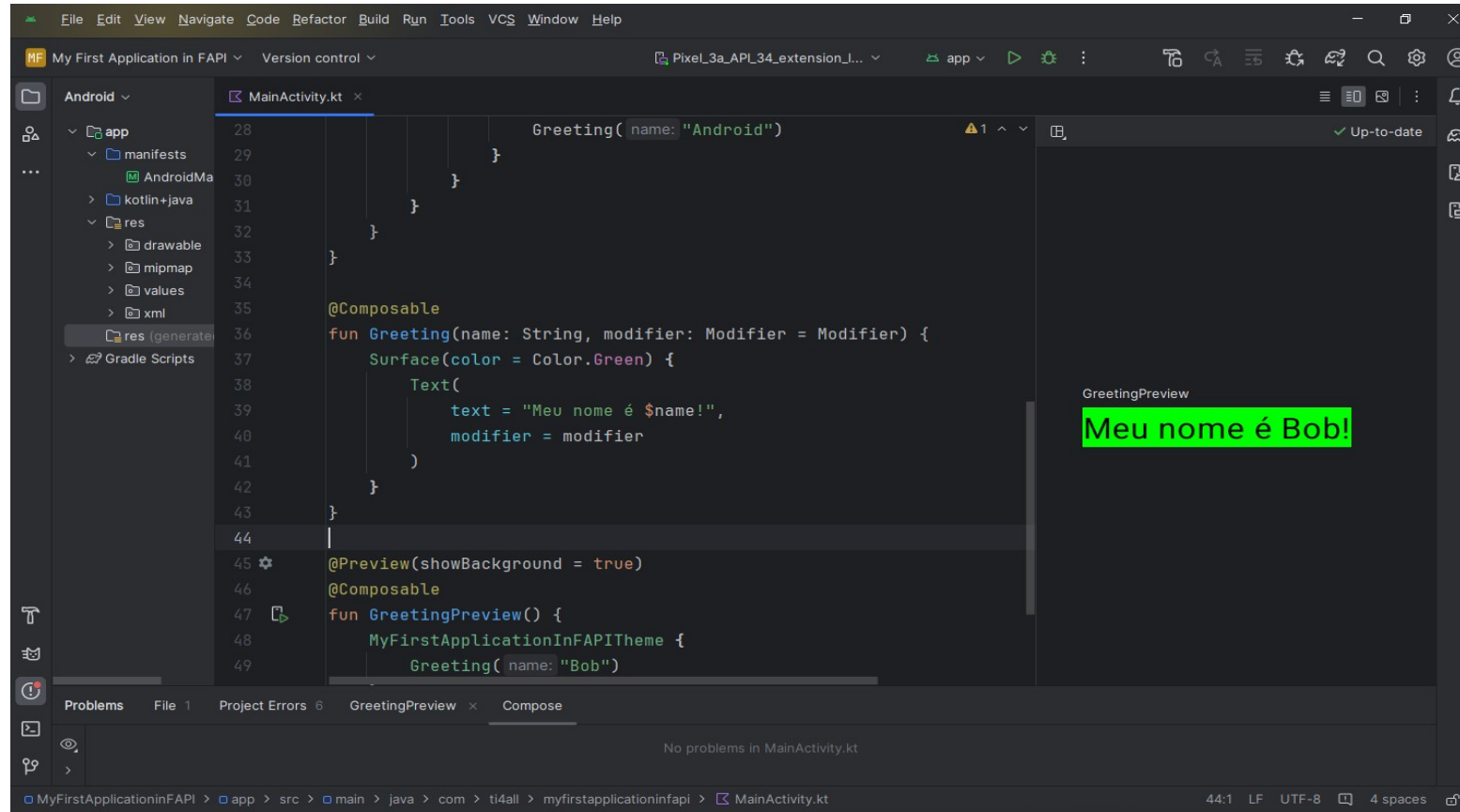
Desenvolvimento Mobile



- Informe o parâmetro color = Color
- Repare que Color está na cor vermelha;
- Posicione o apontador sobre Color;
- Escolha "Import"
- Localize a biblioteca

androidx.compose.ui.graphics.Color

Desenvolvimento Mobile



Repita esse processo (a partir do slide 17) até que o mesmo se torne natural para você;

Desenvolvimento Mobile

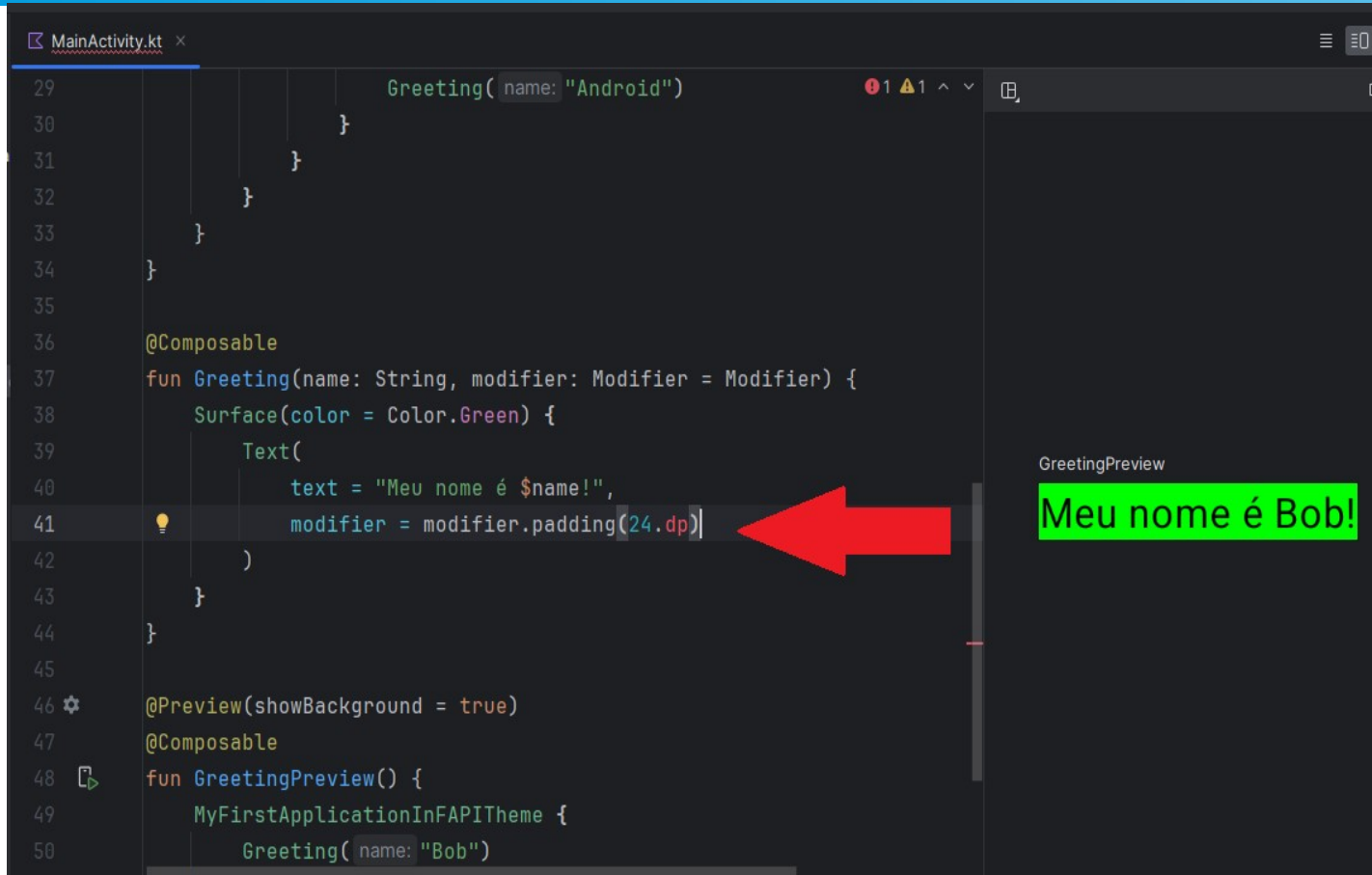


Adicionando
preenchimento
ao redor do texto:

Alterar o
modificador
"modifier",
inserindo o
método padding:

Modifier.padding

Desenvolvimento Mobile



```
29      Greeting( name: "Android")
30    }
31  }
32  }
33  }
34  }
35
36  @Composable
37  fun Greeting(name: String, modifier: Modifier = Modifier) {
38    Surface(color = Color.Green) {
39      Text(
40        text = "Meu nome é $name!",
41        modifier = modifier.padding(24.dp)
42      )
43    }
44  }
45
46  @Preview(showBackground = true)
47  @Composable
48  fun GreetingPreview() {
49    MyFirstApplicationInFAPITheme {
50      Greeting( name: "Bob")
```

GreetingPreview

Meu nome é Bob!

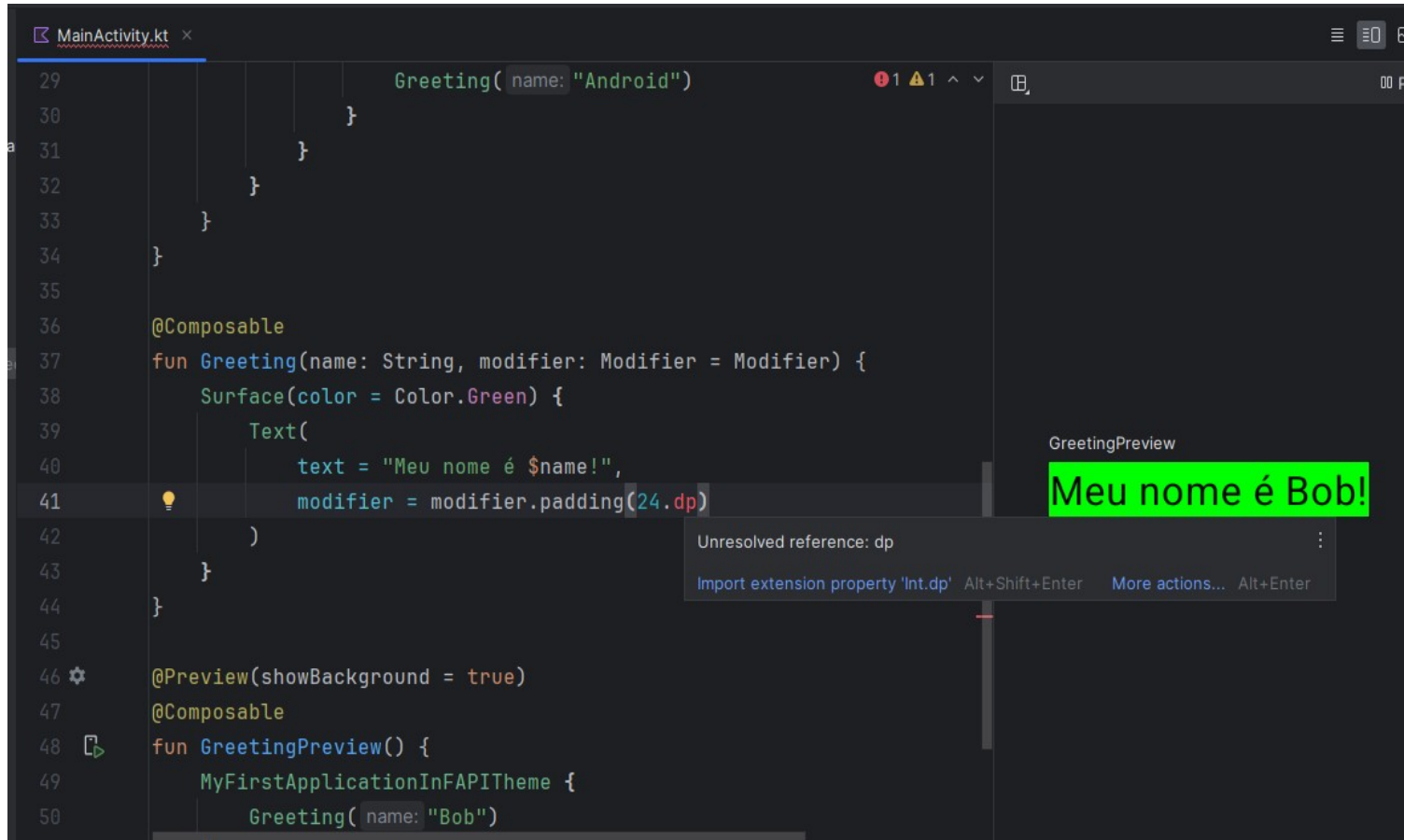
Note que “dp” está em vermelho.

Isso significa que está ocorrendo um erro.

Para saber o erro e a sugestão para correção:

Posicionar o apontador do mouse sobre a palavra em destaque:

Desenvolvimento Mobile



```
29      Greeting( name: "Android")
30    }
31  }
32 }
33
34 }
35
36 @Composable
37 fun Greeting(name: String, modifier: Modifier = Modifier) {
38     Surface(color = Color.Green) {
39         Text(
40             text = "Meu nome é $name!",
41             modifier = modifier.padding(24.dp)
42         )
43     }
44 }
45
46 @Preview(showBackground = true)
47 @Composable
48 fun GreetingPreview() {
49     MyFirstApplicationInFAPITheme {
50         Greeting( name: "Bob")
51     }
52 }
```

GreetingPreview

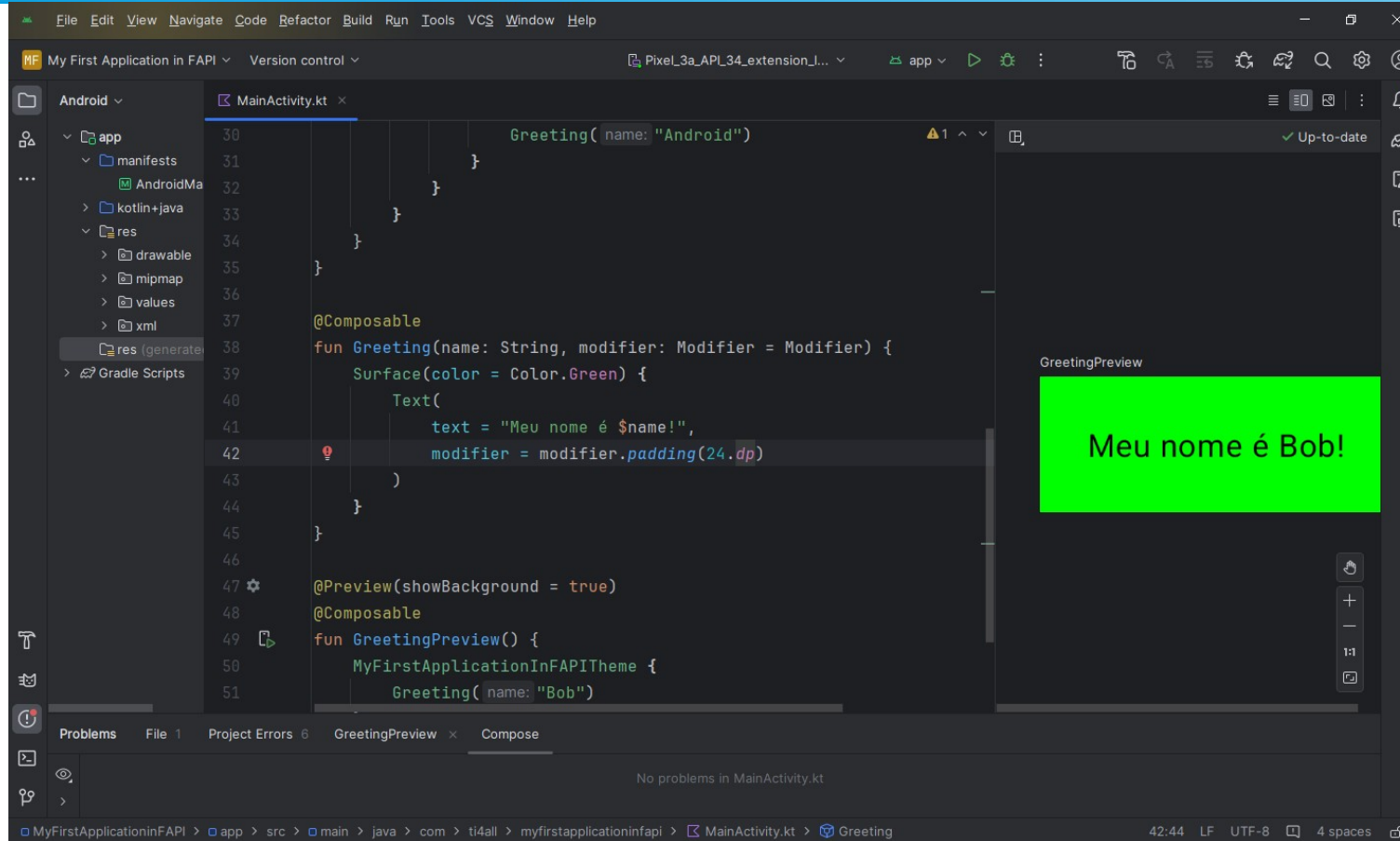
Meu nome é Bob!

Unresolved reference: dp

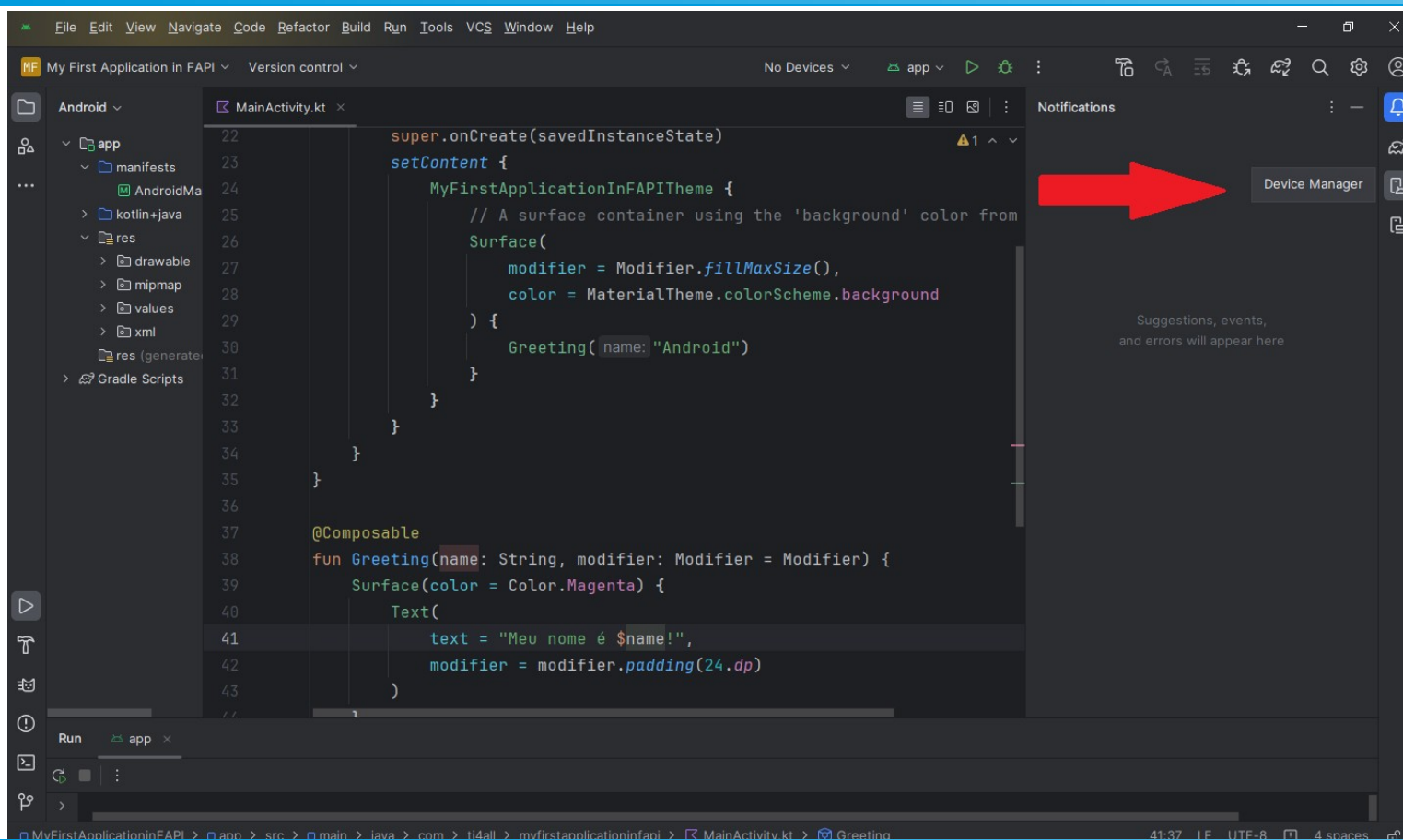
Import extension property 'Int.dp' Alt+Shift+Enter More actions... Alt+Enter

Escolha a opção
“Import extension
property “int.dp”

Desenvolvimento Mobile



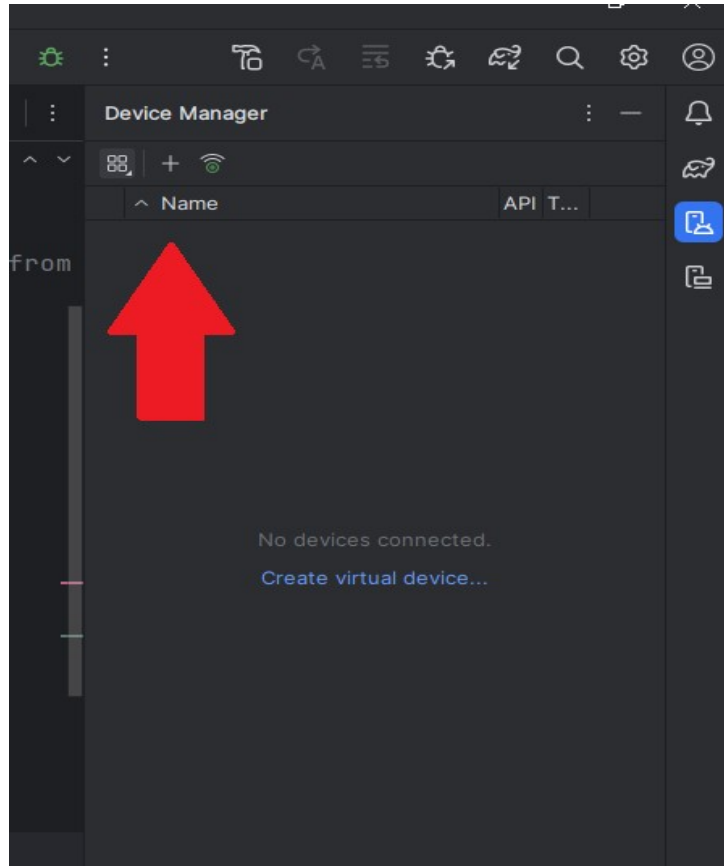
Desenvolvimento Mobile



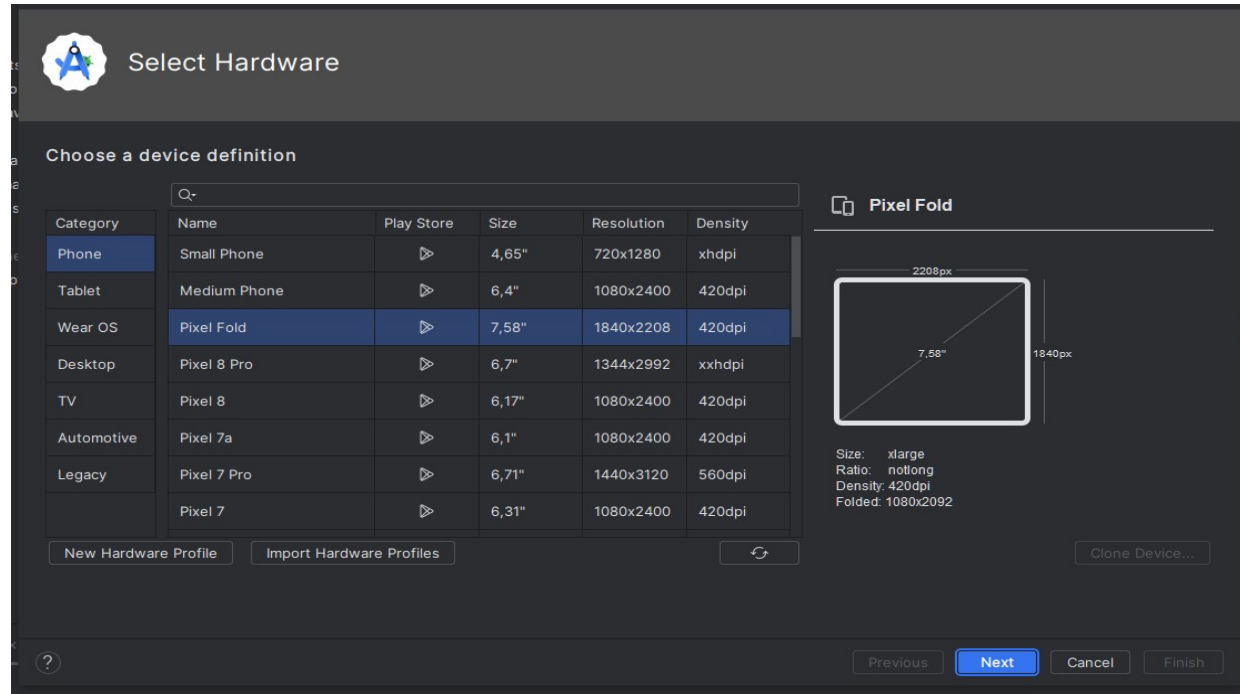
Configurando o Emulador Android:

Clique em “Device Manager”

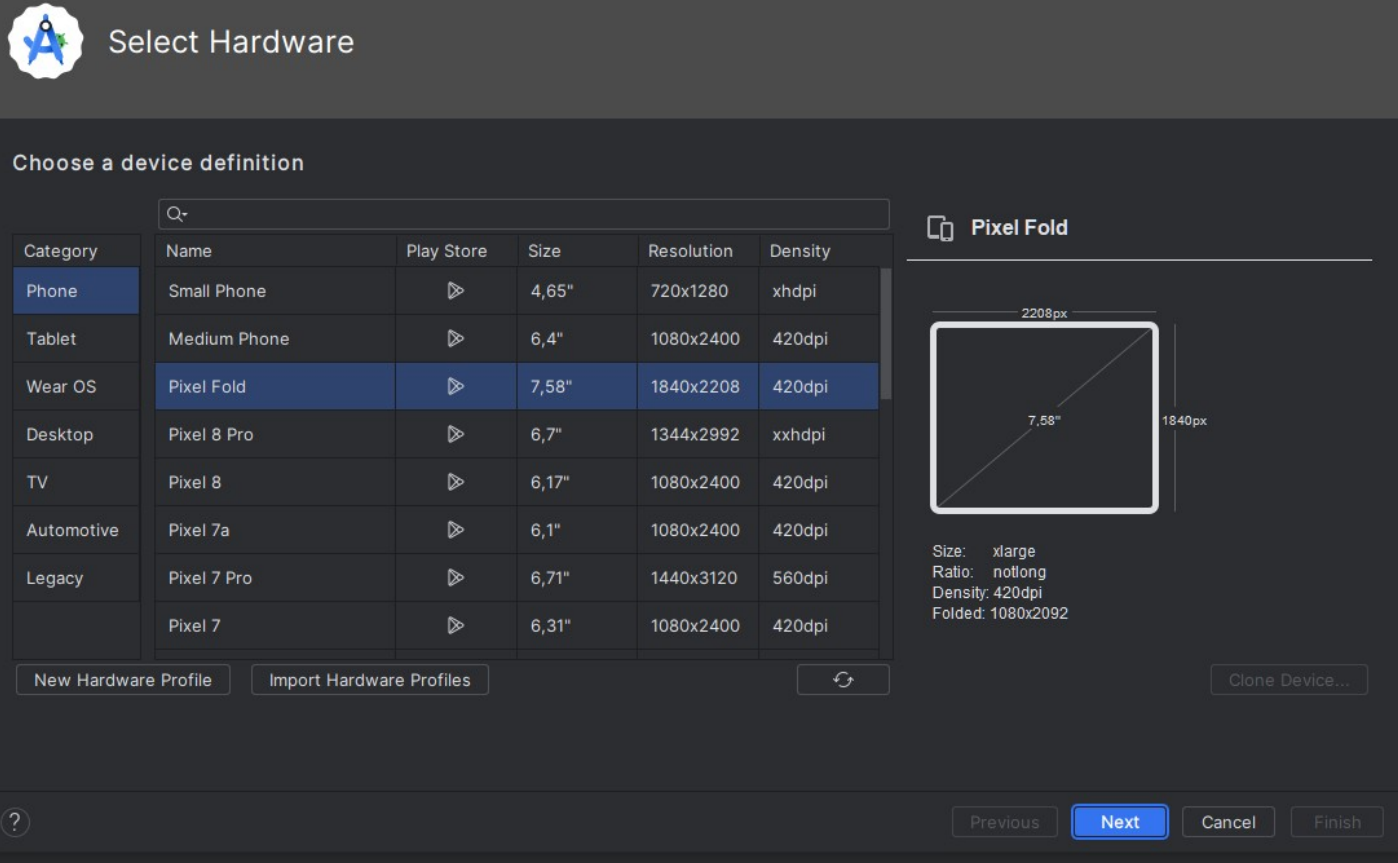
Desenvolvimento Mobile



Configurando o Emulador Android:
No “Device Manager”, busque a opção [+]



Desenvolvimento Mobile



Configurando o Emulador Android:

Normalmente usa-se as opções sugeridas pela IDE mas estas podem ser alteradas pelo usuário

Tablet

Wear OS

TV

Automotive

Desenvolvimento Mobile



System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
VanillaIceCream	VanillaIceCream	x86_64	Android API VanillaIceCream (G
UpsideDownCakePrivacy...	UpsideDownCak	x86_64	Android API UpsideDownCakeP
TiramisuPrivacySandbox	TiramisuPrivacyS	x86_64	Android 14.0 (Google Play)
UpsideDownCake	34	x86_64	Android 14.0 (Google Play)
Tiramisu	33	x86_64	Android 13.0 (Google Play)
Sv2	32	x86_64	Android 12L (Google Play)
S	31	x86_64	Android 12.0 (Google Play)
R	30	x86	Android 11.0 (Google Play)
Q	29	x86	Android 10.0 (Google Play)
Pie	28	x86	Android 9.0 (Google Play)

R

API Level

30

Type

Google Play

Android

11.0

Google Inc.

System Image

x86

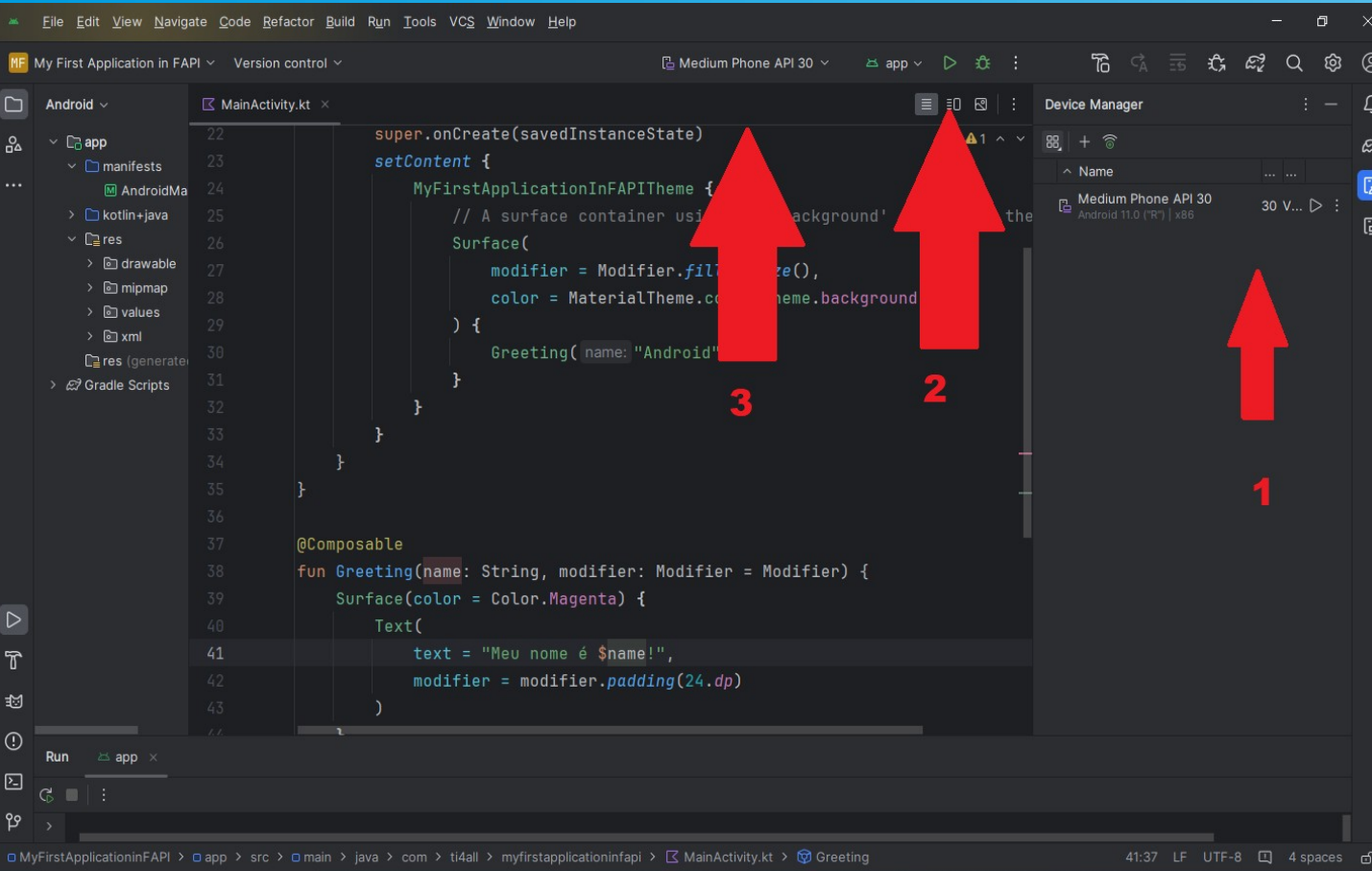
We recommend these Google Play images because this device is compatible with Google Play.

Configurando o Emulador Android:

Selecione a imagem do sistema operacional que irá “rodar” no emulador

Sugiro utilizar o sugerido pela IDE.

Desenvolvimento Mobile

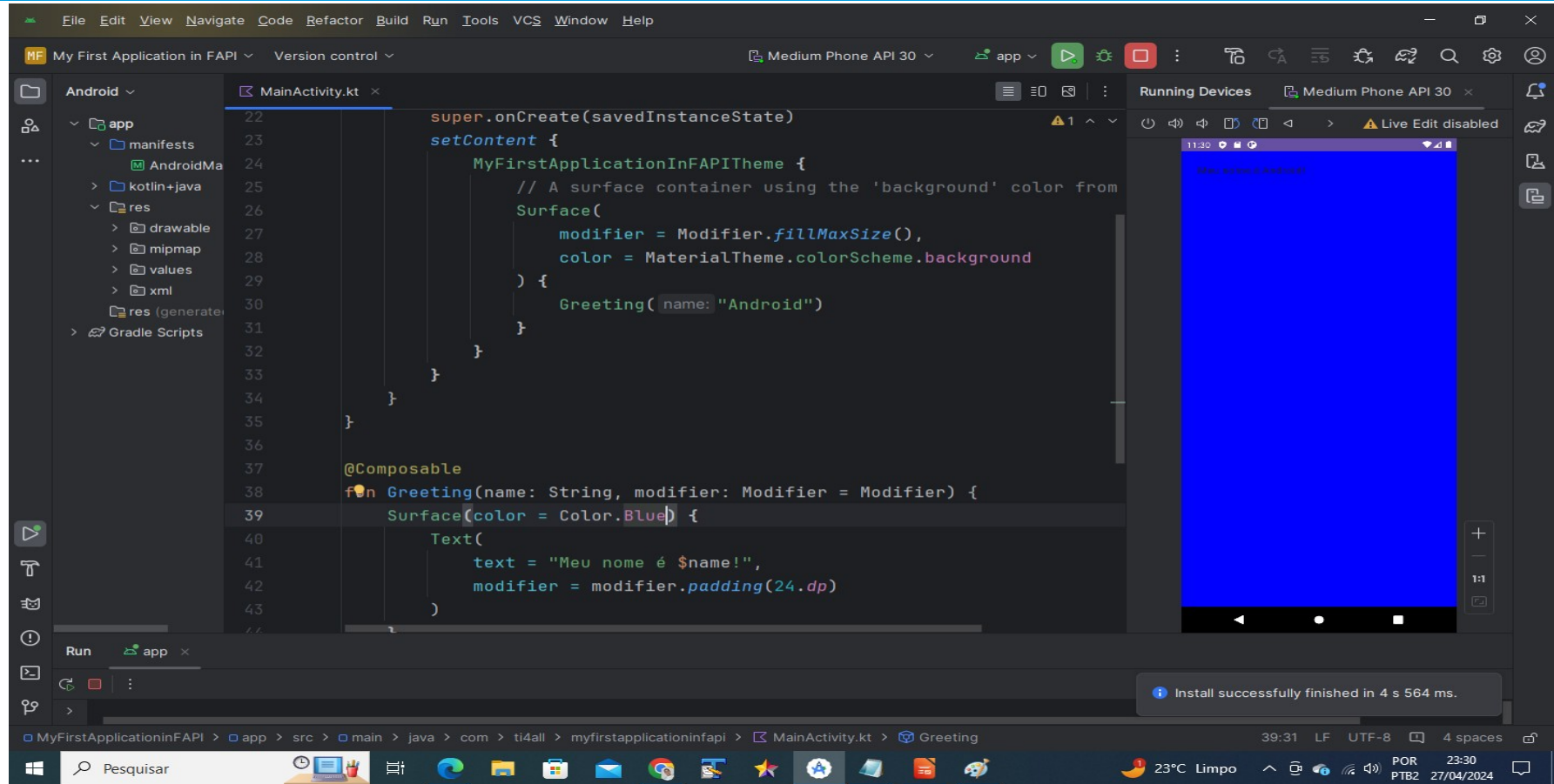


1) Emulador já aparecerá na área “Device Manager”

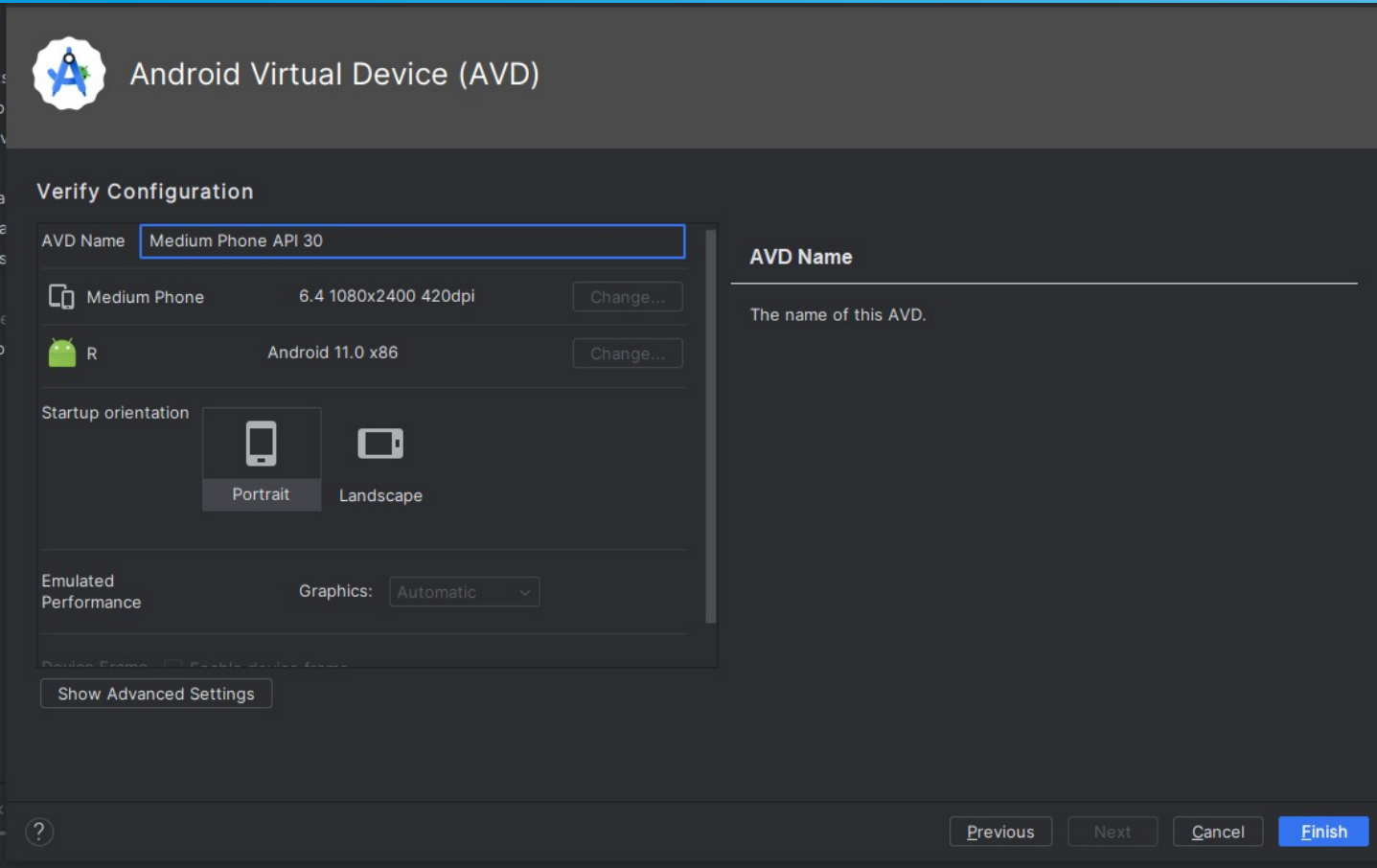
2) Botão “play” do emulador selecionado (inicia o emulador)

3) Seleção do Emulador que será executado

Desenvolvimento Mobile



Desenvolvimento Mobile



Configurando o Emulador Android:

Ajustar a configuração básica

Nome (alias) do Emulador E Orientação.

[Finish]

Desenvolvimento Mobile

Componentes Principais de um Aplicativo Android

- Os aplicativos Android são compostos por vários componentes principais que interagem entre si e com o sistema operacional para fornecer uma experiência completa ao usuário. Esses componentes incluem Activities, Services, Broadcast Receivers, e Content Providers.

Desenvolvimento Mobile

Activities:

- Definição: Uma Activity representa uma única tela com uma interface de usuário (UI) em um aplicativo Android. É a base de quase todas as interações com o usuário em um app.
- Ciclo de Vida: As Activities possuem um ciclo de vida bem definido, que inclui estados como onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(). Isso permite que o Android gerencie a memória de forma eficiente e ajuste a interação do usuário conforme ele navega entre diferentes telas.

Desenvolvimento Mobile

Activities:

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        // Configuração inicial da UI e lógica de negócios  
    }  
    override fun onStart() {  
        super.onStart()  
        // Preparar a Activity para se tornar visível ao usuário  
    }  
    override fun onResume() {  
        super.onResume()  
        // A Activity está no topo da pilha e interagindo com  
        // o usuário  
    }  
}
```

```
    override fun onPause() {  
        super.onPause()  
        // Salvar estados críticos da Activity antes de  
        // ela ser pausada  
    }  
    override fun onStop() {  
        super.onStop()  
        // A Activity não é mais visível ao usuário  
    }  
    override fun onDestroy() {  
        super.onDestroy()  
        // A Activity está sendo destruída, liberar  
        // recursos  
    }  
}
```

Desenvolvimento Mobile

Services:

- São componentes que executam operações em segundo plano, sem interface de usuário.
- São usados para tarefas que precisam continuar funcionando mesmo quando o usuário não está interagindo diretamente com o app (tocar música, baixar arquivos, etc).
- Tipos de Services:
 - Foreground Services: Executam tarefas que o usuário percebe, como tocar música. Devem exibir uma notificação persistente enquanto estão em execução.
 - Background Services: Executam tarefas que o usuário não percebe diretamente, como sincronizar dados em segundo plano.
 - Bound Services: Permitem que componentes como Activities se conectem ao service para interagir com ele. Eles duram apenas enquanto algum componente estiver vinculado a eles.

Desenvolvimento Mobile

Gerenciamento de Ciclo de Vida Services:

- Services têm um ciclo de vida diferente das Activities. Eles são iniciados e executados até que sejam explicitamente parados ou quando o trabalho é concluído.

Desenvolvimento Mobile

Broadcast Receivers:

- Componentes que permitem que um aplicativo responda a mensagens enviadas por outros aplicativos ou pelo próprio sistema Android.
- Essas mensagens (broadcasts) podem ser eventos como a mudança no estado da conectividade, a bateria fraca, ou a finalização do download de um arquivo.

Tipos de Broadcasts:

- Broadcasts Explícitos: Destinados a um aplicativo específico.
- Broadcasts Implícitos: Destinados a todos os aplicativos interessados naquele evento.

Desenvolvimento Mobile

Content Providers:

- Content Providers gerenciam o acesso a um conjunto estruturado de dados, permitindo que diferentes aplicativos compartilhem dados de forma controlada e segura.
- Um exemplo clássico é a aplicação de contatos do Android, que utiliza um Content Provider para fornecer acesso aos dados de contatos para outros aplicativos.
- **Funções Principais:**
 - CRUD Operations: Content Providers permitem a realização de operações de Create, Read, Update, e Delete em seus dados.
 - URI: Os dados de um Content Provider são acessados através de URIs, que funcionam como identificadores únicos dos recursos.