

Início > Full stack

2 anos atrás

# Introdução a Kotlin: um guia para começar







Os criadores de Kotlin tinham duas opções: adaptar novas funcionalidades em linguagens já existentes ou criar uma linguagem nova.

Por mais que a segunda opção demande todo um investimento no ecossistema, criação de bibliotecas, documentação, ferramentas de apoio, grupos e tudo que for necessário para a linguagem ser usada e adotada... a primeira opção era inviável.

Assim nasceu Kotlin, pensada para ter as características da linguagem Scala e Java, porém eliminando tudo aquilo que é desvantagem nesses ambientes.

Tá com uma cara boa, não é? Então acompanhe esse guia para conhecer essa linguagem moderna e queridinha de grandes empresas!

**Conteúdo** [ ocultar ]

- 1 O que é Kotlin?
- 2 Por que utilizar Kotlin?
- 3 Características do Kotlin vs. Java
  - 3.1 Códigos equivalentes de Java vs. Kotlin
  - 3.2 Sintaxes Kotlin
  - 3.3 Variáveis
  - 3.4 Tipos básicos
  - 3.5 Classes
  - 3.6 Funções
  - 3.7 Expressões condicionais
  - 3.8 Arrays
  - 3.9 Loops
- 4 Kotlin for JavaScript
-  5 Kotlin Native
-  6 Kotlin for Data Science
-  7 Kotlin playground
-  8 Exemplo de Projeto Android com Kotlin (Android Studio)
  - 8.1 Abrindo o Android Studio
  - 8.2 Template do projeto
  - 8.3 Configuração do projeto
  - 8.4 Estrutura de pastas no Android Studio
- 9 Conclusão

## O que é Kotlin?



Logo da linguagem Kotlin.

Kotlin é uma linguagem de programação open source multiplataforma criada pela JetBrains e amplamente usada por desenvolvedores Android.

É multiparadigma, sendo totalmente orientada a objetos, mas com algumas

✉ características de linguagem funcional (como as funções lambda), com tipagem estática e executada pela Java Virtual Machine (Máquina Virtual do Java).

🐦 As semelhanças com Java são tantas que você até pode usar Java dentro do seu código  
🌐 Kotlin, mas vamos ver isso mais para frente! (~~é a tal de interoperabilidade~~).

🗨 Foi lançada, na sua versão estável, em 2016. Logo em seguida, o Google se juntou a JetBrains e agora, as duas empresas mantêm essa linguagem através da Kotlin Foundation.

Além disso, o Google anunciou que Kotlin faz parte das linguagens oficiais do Android. Então, junto com a JVM, ela pode ser usada para desenvolvimento web, mobile e desktop.



## Por que utilizar Kotlin?

Em primeiro lugar, a **produtividade**. Kotlin é uma linguagem enxuta e intuitiva, utilizando cerca de 40% menos códigos para representar a mesma coisa que o Java.

A linguagem de programação Java é verbosa, e apesar de existir projetos para diminuir essa desvantagem, como o Lombok.

Assim, ela não é tão eficaz quanto a expressividade que Kotlin oferece nativamente. A curva de aprendizagem do Kotlin é rápida e relativamente pequena. 🤖

Entretanto, escrever menos códigos significa que mais regras precisam ser lembradas. Isso vai gerar uma boa carga de estudos aprendendo a melhor forma de escrever um código Kotlin.

Outro ponto a ser destacado é que por ser estaticamente tipada, essa linguagem oferece mais segurança e performance ao programador Kotlin.

Veja só, Kotlin mostra os erros cometidos na hora da escrita do código durante a compilação, em vez de dar erro na hora da execução do código, muito tempo depois. Isso torna a linguagem mais rigorosa, mas evita defeitos não intencionais.

E mais uma coisinha, quer saber quais grandes empresas também estão usando o Kotlin?



- Google;



- Amazon;



- Netflix;



- Pinterest;



- Uber;

- Foursquare;

- Trello;

- Capital One;

- Coursera.

E se você já estiver convencido, essas são as ferramentas de desenvolvimento que já aceitam o Kotlin:

- Spring;

- Gradle;

- Vert.x;

- Spark Java;

- Codename One;

- JetBrains.

## Características do Kotlin vs. Java

Sabia que na comunidade Dev Android, Kotlin é comparada ao Java tanto quanto Swift é comparada ao Objective-C (Dev IOS)?

As semelhanças são grandes, porém tem algumas coisinhas que colocam a linguagem desse artigo na frente, saca só.

- **Interoperabilidade:** Kotlin é totalmente interoperável com o Java. O que isso quer dizer? Quer dizer que podemos utilizar Kotlin quando quisermos em algum projeto Java.

Ou seja, podemos inserir código Kotlin no projeto Java e utilizá-lo a partir do código Java do projeto. Isso mesmo, num mesmo projeto podemos ter as duas linguagens trabalhando juntas 🤖.

E não para por aí. Essa relação é bidirecional, isto é, podemos inserir código Java num projeto Kotlin e utilizar o código a partir de Kotlin.

Outra coisa, não precisamos nos preocupar se bibliotecas Java funcionarão ou não. Isso porque as APIs que executam no Java também executam em Kotlin 🍪.

- **Sintaxe:** essa característica refere-se à legibilidade que, se comparada a do Java, é mais simples e concisa e, portanto, é vantajosa em relação ao Java. Mais abaixo, dois exemplos mostram isso melhor.
- **Null Safety:** essa característica fantástica de Kotlin diferencia referências que podem ser nulas daquelas que não podem ser, de acordo com o tipo definido.

Sabe aquele problema chato do *NullPointerException*, então... é justamente para tratar disso. O trecho de código abaixo esclarece melhor essa característica.

```
var stringValue: String = "Hello World!"  
stringValue= null; // error
```

```
var stringValue: String? = "Hello World!"  
stringValue = null // ok
```

- **Coroutines:** são rotinas cuja execução pode ser interrompida permitindo que outra rotina seja executada naquele instante.

Não significa que a rotina interrompida não voltará a ser executada, pelo contrário, será retomada, porém em um momento futuro, cooperando entre si. Não são as threads que o Java utiliza, mas é algo parecido.

No Android, as corrotinas são utilizadas para simplificar o código que é executado de forma assíncrona, ajudando a gerenciar tarefas de longa duração que podem bloquear a linha de execução principal e fazer com que o seu app pare de responder.

- Suporte total pelo o Android Studio. 🤖



- **Comunidade:** Kotlin tem um grande apoio e muitas contribuições da comunidade, que vem crescendo em todo o mundo. De acordo com o Google, mais de 60% dos 1000 principais aplicativos da Play Store utilizam Kotlin.  
(<https://kotlinlang.org/docs/reference/android-overview.html>)

- **Suporte no Android Jetpack e outras bibliotecas:** as extensões KTX adicionam recursos a Kotlin, como corrotinas, funções de extensão, lambdas e parâmetros nomeados, às bibliotecas Android existentes.

“ **Programação Mobile**: tudo para começar hoje!

Agora que você já está por dentro das principais características que dão brilho ao Kotlin, vou te mostrar algumas diferenças entre Kotlin vs. Java.

VAGAS PARA DESENVOLVEDORES

Encontre as melhores vagas para Pessoas  
Desenvolvedoras na **GeekHunter**

Saber Mais

# Códigos equivalentes de Java vs. Kotlin

## Exemplo geral

Nesse exemplo, você verá como mais de uma dezena de linhas em Java consegue ser facilmente representada em Kotlin usando somente uma. Em Java:

```
public class Pais {  
    private String nome;  
    private Long populacao;  
  
    public Pais (String nome, Long populacao) {  
        this.nome = nome;  
        this.populacao = populacao;  
    }  
  
    public String getNome ( ) {  
        return nome;  
    }  
  
    public void setNome (String nome) {  
        this.nome = nome;  
    }  
  
    public Long getPopulacao ( ) {  
        return populacao;  
    }  
}
```



Agora em Kotlin:

```
data class Pais(var nome: String; var populacao: Long)
```

## Função Soma

Na função Soma, você também consegue ver essa redução de linhas para a mesma função. Em Java:

```
public class FuncaoSoma {
```

```
public static int soma (int numero1, int numero2) {  
    return a + b;  
}  
}
```

Em Kotlin:

```
fun soma (numero1: Int, numero2: Int): Int {  
    return a + b;  
}
```

Com os dois exemplos acima, fica evidente a vantagem de sintaxe que Kotlin possui em relação ao Java.

## Sintaxes Kotlin



Antes de utilizarmos a linguagem, vamos aprender um pouco da sua sintaxe. Um tutorial breve para iniciantes dessa linguagem de programação.



## Variáveis



Existem dois tipos principais: **val** e **var**.

- **Val:** utilizada para declarar constantes. É a forma mais recomendada para declaração de variáveis;
- **Var:** declarar qualquer variável cujo valor pode ser alterado.

A primeira requer sua inicialização logo que é declarada. A segunda não precisa ser inicializada no momento da sua declaração.

Além disso, tanto a primeira quanto a segunda são declaradas utilizando o *pascal notation*, que é da forma abaixo:

**nome\_da\_variável : tipo\_da\_variável**



Outra questão é a inferência de tipos. O tipo de uma variável não precisa ser explicitamente definido, ou seja, não é obrigatório.

Porém, no momento que um valor é atribuído, o tipo da variável é inferido de acordo com o tipo desse valor 😊. Vamos ver alguns tipos básicos de Kotlin.

## Tipos básicos

Todos os tipos básicos do Java estão presentes aqui. O detalhe é que, em Kotlin, todos os tipos são objetos.

Isso quer dizer que apenas se trabalha com os wrappers do Java, mas não com seus tipos primitivos correspondentes 😞.

- **Long:** inteiro de 64 bits.



- **Int:** inteiro de 32 bits.



- **Short:** inteiro de 16 bits.



- **Byte:** inteiro de 8 bits.



- **Double:** ponto flutuante de 64 bits.

- **Float:** ponto flutuante de 32 bits.

- **Boolean:** mesmo que o do Java. Pode receber o true ou false.

Algo de importante a mencionar é as operações suportadas pelo tipo booleano, que são:

- **|** – disjunção (ou)
- **&&** – conjunção (e)
- **!** – negação
- **String:** em Kotlin, podem ser criadas com aspas duplas ou triplas.

- **Char:** os caracteres são representados por esse tipo e não podem ser tratados diretamente como números, sejam eles inteiros ou de ponto flutuante.

## Classes

As classes em Kotlin são **final** por default, isso significa que elas não podem ser extendidas. O tipo classe é definido como a seguir.

### class NomeDaClasse

O modo como foi feito acima segue utilizando a palavra reservada **class** seguido do nome da classe. Abaixo temos uma estrutura de classe mais completa com atributos e métodos.

```
class NomeDaClasse ( var atributo1: String; var atributo2: String;  
var atributo3: Int ) {
```

```
    fun funcao1(): String {  
        return atributo1;  
    }
```

```
    fun funcao2(): String {  
        return atributo2;  
    }
```

```
    fun funcao3(): Int {  
        return atributo3;  
    }
```

```
    // ...  
}
```

Percebeu algo estranho? Você deve ter notado a declaração dos três atributos entre os parêntesis logo em seguida do nome da classe. Esse é o construtor da classe. É uma forma diferente da que estamos acostumados no Java, mas tem o mesmo efeito.

Além disso, mesmo que você não especifique os atributos do construtor da classe, ela disponibilizará um construtor padrão sem nenhum parâmetro 😊.

## Funções

Para declarar uma função em Kotlin, utilizamos a palavra reservada **fun** seguida pelo nome da função, e logo em seguida os parênteses com a definição dos tipos de entrada da função. Também é necessário declarar o tipo de saída.

```
fun printERetorna(frase: String): String {  
  
    print(frase);  
    return frase;  
}
```

## Expressões condicionais

O mecanismo mais comum de lógica condicional é a estrutura **if-else**, comum em muitas outras linguagens de programação, embora ela possua vários mecanismos para implementar a lógica condicional.

```
if (/*condição*/ ) {  
  
} else {  
    // bloco de código que será executado caso a condição seja  
    falsa...  
}
```

## Arrays

Conforme dito na documentação, Arrays são representados pela classe **Array** possuindo de funções **get** e **set** e outras funções membro úteis, além da propriedade **size**.

Diferente do Java, não?

```
val numerosQuadrados = Array(5) { i -> ( i * i ) }  
  
val numerosDobrados = Array<Int> (5, { i -> 2 * i } )
```

## Loops

Em relação aos loops dentro de Kotlin, não há muitas diferenças do que já estamos acostumados no Java.

### While

```
while (/*condição de parada*/) {  
  
    // expressões  
    ...  
}
```

### For

```
for ( i in <valor inicial>..  
    // expressões  
    ...  
}
```

Lembrando que utilizamos o while quando temos um número indefinido de repetições e o for quando termo a quantidade determinada 🤪.

**VAGAS MOBILE**  
**Encontre vagas para**  
**Desenvolvedor Kotlin na Geekhunter**  
**Saber Mais**

Entendido esse geralção sobre as sintaxes e conceitos do Kotlin, vou apresentar algumas outras formas de se trabalhar com ele que podem ser interessantes para projetos futuros.

# Kotlin for JavaScript

Kotlin for JavaScript fornece a capacidade de transpilar código Kotlin, a biblioteca padrão e quaisquer dependências compatíveis para JavaScript 🤖. Isso pode acontecer tanto do lado do cliente como do lado do servidor

E outra coisa, a biblioteca padrão de Kotlin está disponível para uso através do NPM (Node Package Manager), que é o gerenciador de pacotes do Node.js padrão 🧑🏻.

Nesse sentido, o site oficial recomenda utilizar Kotlin/JS através de plugins Gradle, que é gerenciador de dependências e build de aplicações back-end baseadas na JVM.

Os plugins são: **kotlin.js** e **kotlin.multiplatform**. São esses plugins que fornecem uma maneira central e conveniente de configurar e controlar projetos Kotlin direcionados ao JavaScript.



Neste link você pode encontrar essa peculiaridade mais detalhada e estudar mais a fundo



: <https://kotlinlang.org/docs/reference/js-project-setup.html>.



## Kotlin Native

Essa é uma tecnologia para compilar código Kotlin para binários nativos, que podem ser executados sem uma máquina virtual. É um backend baseado em LLVM para o compilador Kotlin e implementação nativa da biblioteca padrão.

Kotlin/Native foi projetado principalmente para permitir a compilação para plataformas onde as máquinas virtuais não são desejáveis ou possíveis, por exemplo, dispositivos embarcados ou IOS. Ele resolve as situações em que um desenvolvedor precisa produzir um programa independente que não requer um tempo de execução adicional ou máquina virtual.

## Kotlin for Data Science

Algumas ferramentas disponibilizam integração da linguagem com dados e pesquisa exploratória.

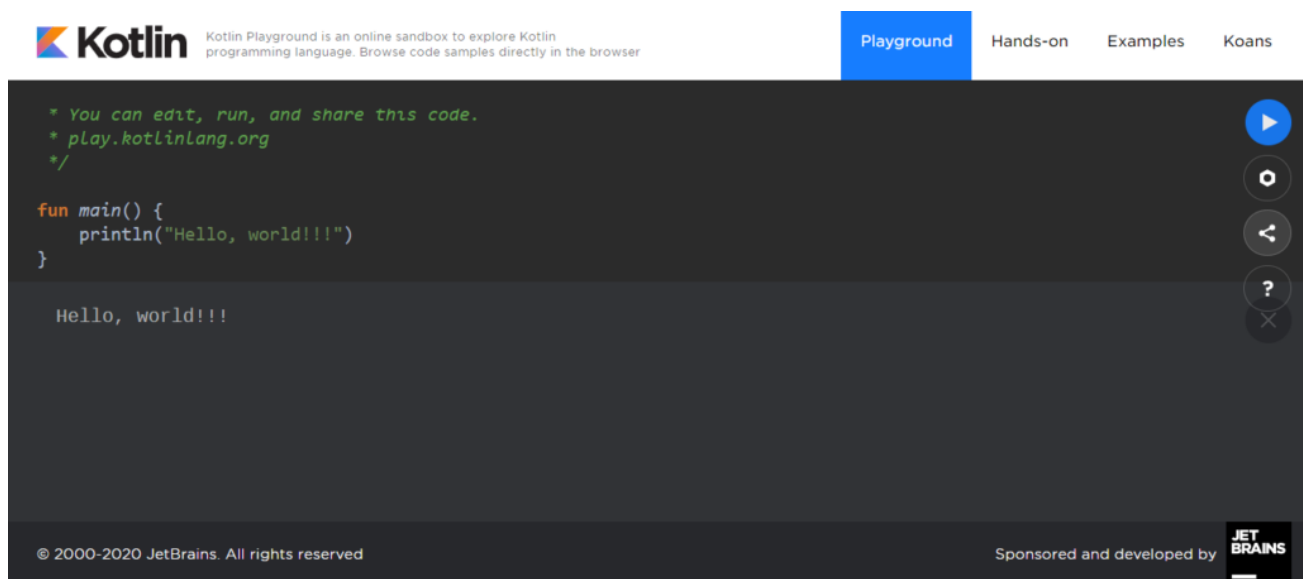
O Kotlin se integra a essas ferramentas para ajudá-lo a explorar dados, compartilhar suas descobertas com colegas ou desenvolver suas habilidades de ciência de dados e aprendizado de máquina.

- **Jupyter** é um app web de código aberto que permite criar e compartilhar documentos que podem conter código, visualizações e texto marcado.
- **Apache Zeppelin** é uma solução popular baseada na web para análise de dados interativa. Fornece forte suporte para o sistema de computação de cluster Apache Spark, que é particularmente útil para engenharia de dados.

## Kotlin playground

Kotlin playground é um sandbox online para explorar a linguagem, podendo auxiliar o desenvolvimento com Kotlin. O link para a ferramenta é: <https://play.kotlinlang.org/>

✉ Podemos escrever e executar código Kotlin bem como compartilhar o código através de um link gerado pela ferramenta. Abaixo, a imagem da ferramenta.



O playground kotlin é um projeto open source. O link do repositório oficial do projeto no GitHub é: <https://github.com/JetBrains/kotlin-playground>.

## Exemplo de Projeto Android com Kotlin (Android Studio)

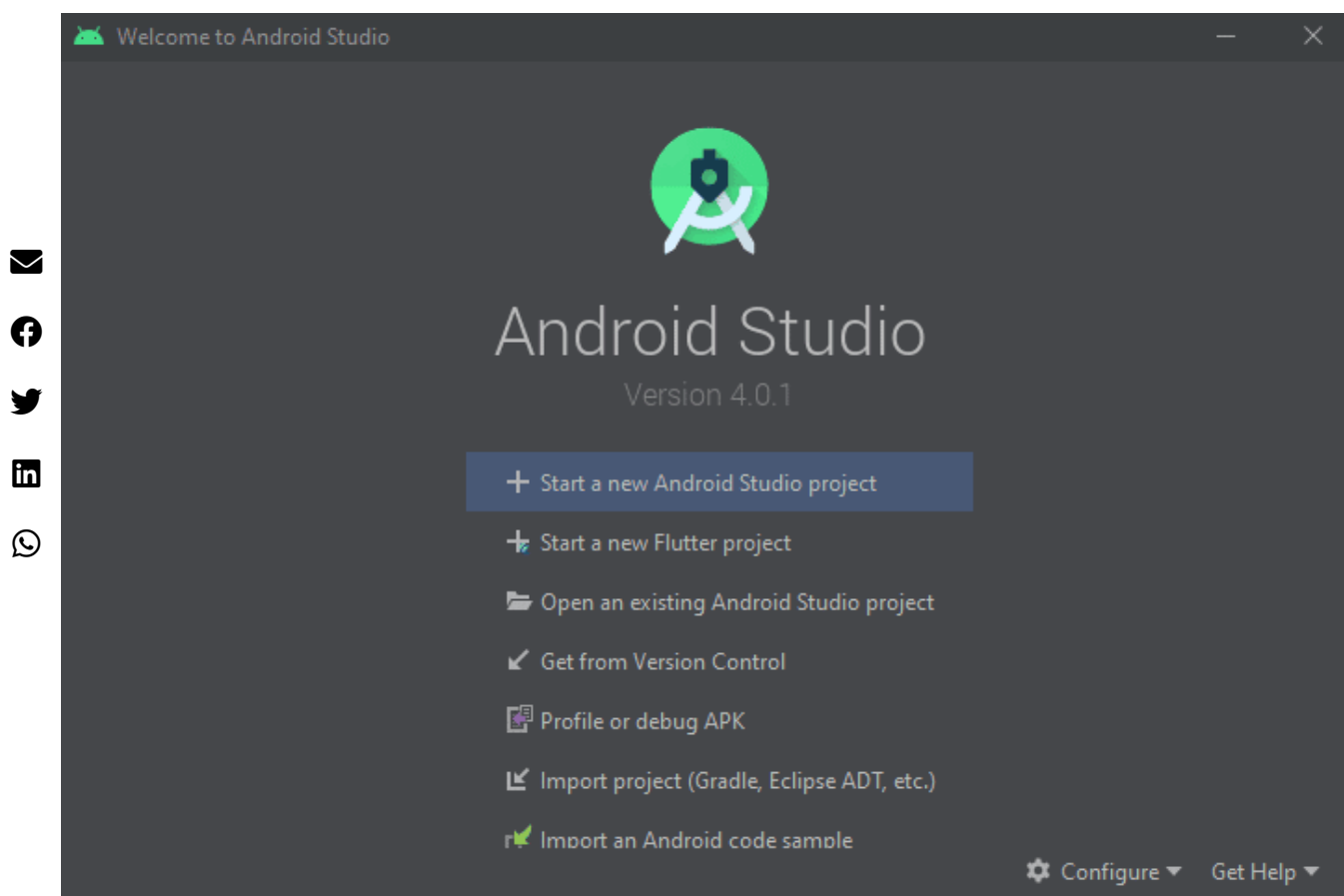
Para iniciarmos, precisamos obviamente do Android Studio instalado na máquina 🤖. O download pode ser feito aqui: [Android Studio](#).

Obs: A versão do Android Studio que estou utilizando é a 4.0.1, porém a versão atual é 4.1.3, o que não muda a forma de criação desse projeto.

Depois de baixado e instalado, o fluxo de telas a seguir mostra como criar o projeto.

## Abrindo o Android Studio

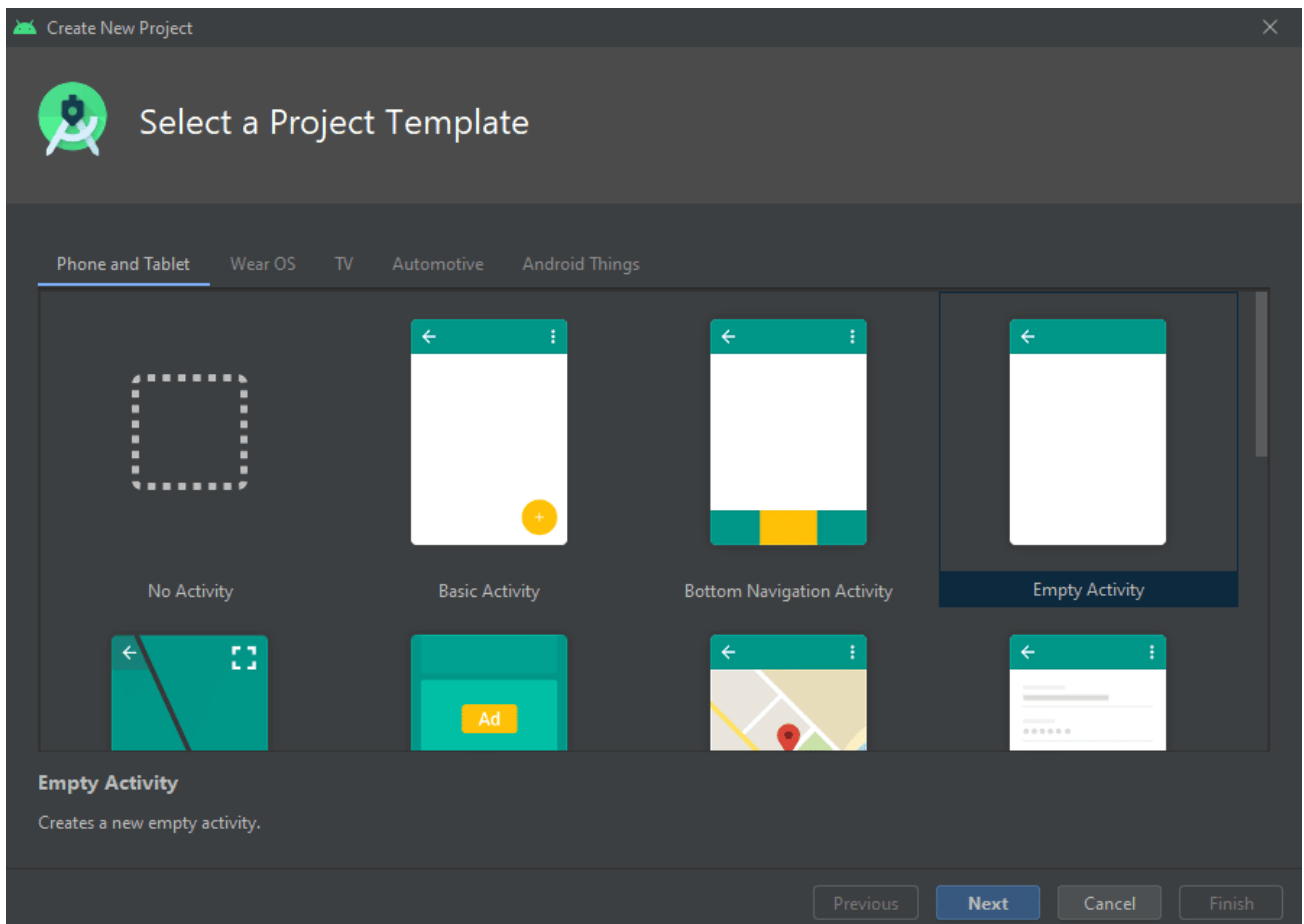
Ao abrir o Android Studio, uma tela parecida com essa é aberta:



Selecione a primeira opção: “Start a new Android Studio project”.

## Template do projeto

Se você está começando com o Android Studio agora, o melhor é começar pela simplicidade. Portanto, selecione o Basic Activity.

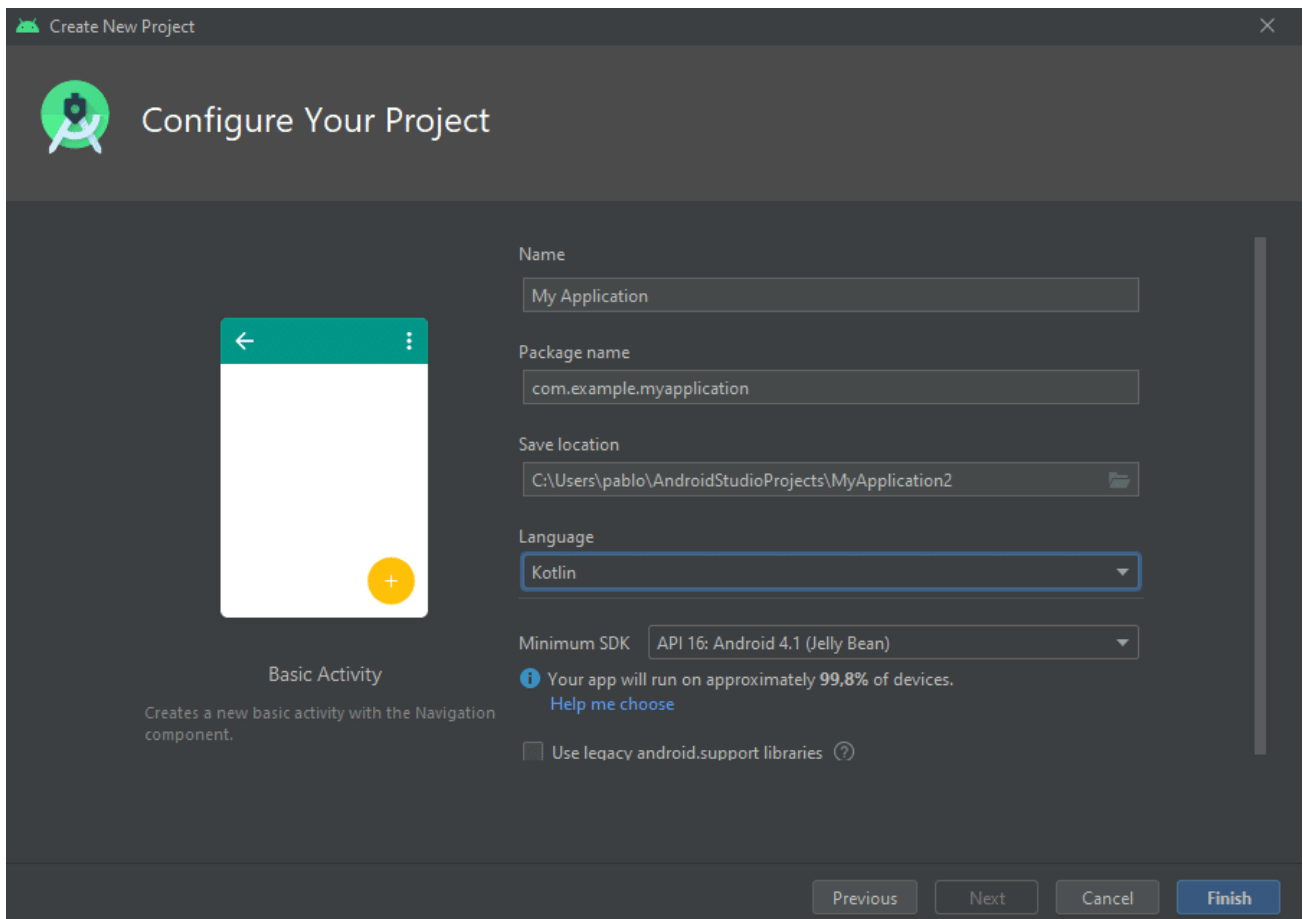


Clique em "Next".

## Configuração do projeto

Depois de selecionar o template e clicar em Next, a janela abaixo aparecerá. Assim, faremos a seleção da linguagem que utilizaremos para codificar. Também podemos escolher o nome, o pacote e a localização do projeto também.

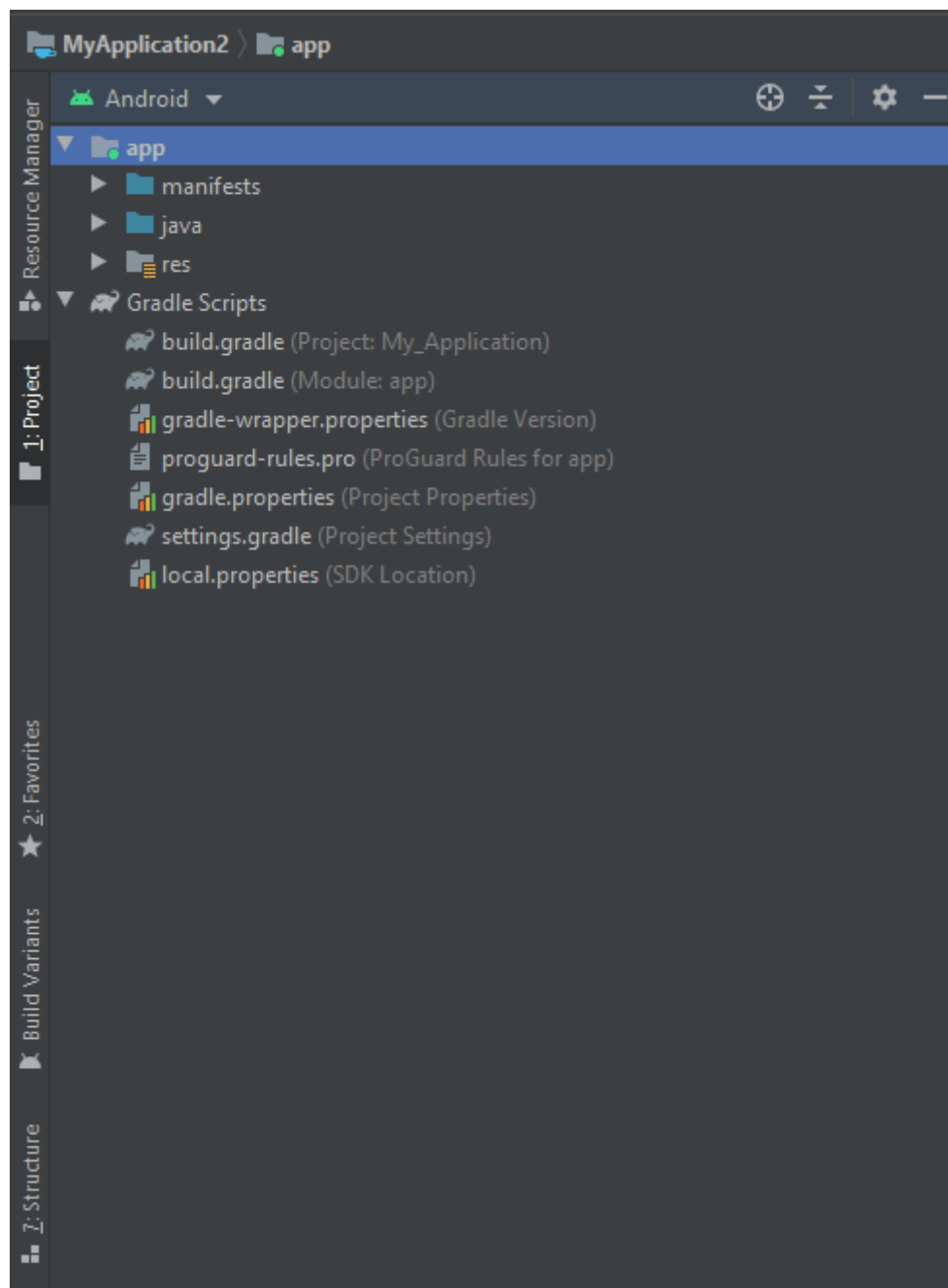




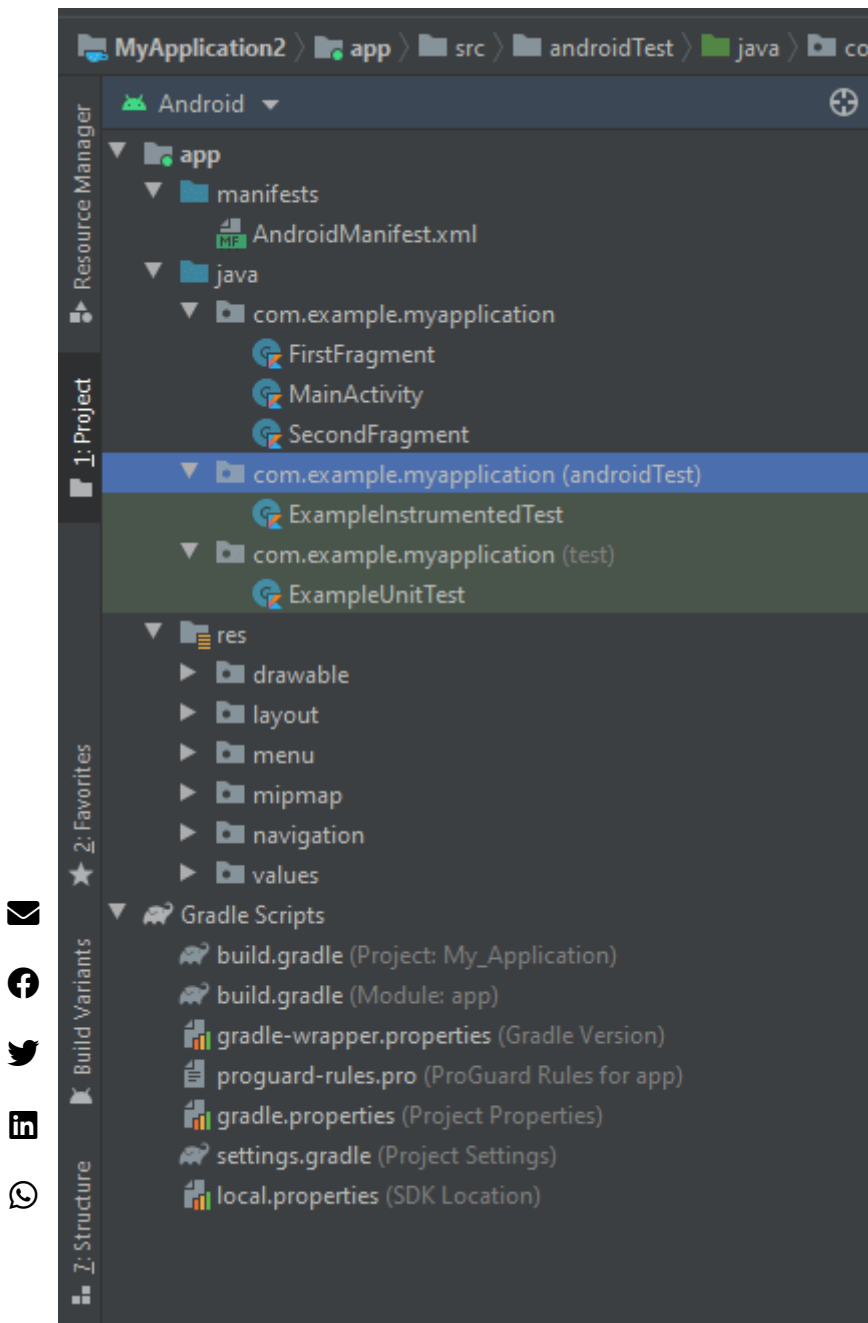
Clique em "Finish" e logo em seguida o projeto será criado.

## Estrutura de pastas no Android Studio

Dessa forma, se tudo deu certo, o projeto foi criado e a estrutura padrão do projeto Android será mostrada.



Expandindo as pastas, você vê algo igual ou semelhante como a figura abaixo:



Se acaso o suporte a Kotlin não venha quando baixamos e instalamos o Android Studio, você pode fazer isso manualmente através do Plugin Manager (Preferences > Plugins).

No IntelliJIDEA — outra IDE criada pela JetBrains da qual o Android Studio foi baseada — pode ser preciso baixar e instalar o Plugin de Kotlin para começar a usá-la. Para isso, você deve ir no Marketplace do IntelliJIDEA, procurar pelo plugin e clicar em Instalar.

## Conclusão

Por fim, esse foi um grande punhado de informações para você se situar e entender alguns conceitos básicos sobre Kotlin.

Vimos algumas características, comparações com Java, principais características

Tenho certeza que você saiu desse artigo com mais conhecimentos do que entrou, não é? Se isso te interessou e você quiser se aprofundar mais nessa linguagem de programação, eis alguns cursos gratuitos para continuar os seus estudos 🥰🤓😊:

- <https://kotlinlang.org/docs/home.html>
- <https://www.udemy.com/course/desenvolvedor-kotlin-iniciante/>
- <https://developer.android.com/courses/android-basics-kotlin/unit-1>

E tem também uma fonte mais “primitiva” de conhecimento, o livro!

- **Programando com Kotlin**, de Stephen Samuel e Stefan Bocutiu.



Livro Programando com Kotlin, de Stephen Samuel.

Bons estudos 😊

VAGAS MOBILE

## Encontre vagas para Desenvolvedor Kotlin na Geekhunter

Saber Mais