Associação Propagadora Esdeva
Centro Universitário Academia –
UniAcademia
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso – Artigo

UM ESTUDO COMPARATIVO SOBRE AS LINGUAGENS JAVA E KOTLIN PARA O DESENVOLVIMENTO DE APLICATIVOS ANDROID

Victor Hugo Fonseca Barbosa
Centro Universitário Academia, Juiz de Fora, MG

Geraldo Magela Almeida Bessa
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: ENGENHARIA DE SOFTWARE

RESUMO

O ambiente de programação para as plataforma pode sofrer atualização com o passar do tempo. O Android teve durante anos o Java como sua linguagem oficial para desenvolvimento, porém este cenário mudou quando foi anunciado em 2017 que o Kotlin também se tornaria uma linguagem oficial, considerada posteriormente a linguagem preferencial. O presente artigo relata as principais diferenças e características entre as linguagens de programação Java e Kotlin encontradas durante o desenvolvimento de um aplicativo com a mesma arquitetura e funcionalidades. Como complemento, foi realizada uma pesquisa sobre as linguagens propostas, que foi enviada para as comunidades de desenvolvedores. Constatou-se que as diferenças entre as linguagens de programação podem fazer com que os desenvolvedores mudem de linguagem na construção de aplicativos, seja por produtividade, praticidade, novos recursos, facilidade de aprendizado ou ainda para se adequar ao mercado de trabalho.

ABSTRACT

The programming environment for the platforms may be updated over time. Android had Java for years as its official language for development, but this scenario changed when it was announced in 2017 that Kotlin would also become an official language, later considered the preferred language. This article reports the main differences and characteristics between the Java and Kotlin programming languages found during the development of an application with the same architecture and functionality. As a complement, a survey was carried out on the proposed languages, which was sent to the developer communities. It was found that the differences between programming languages can cause developers to change languages in building applications, whether due to productivity, practicality, new features, ease of learning or to adapt to the job market.

Palavras-chave: Desenvolvimento Android. Linguagens de programação. Aplicativo. Diferenças entre Linguagens.

1 INTRODUÇÃO

Este artigo apresenta uma análise com base na construção de um aplicativo chamado “Barreiras Sanitárias”, que tem como objetivo fazer o cadastro de pessoas e captar respostas para um questionário sobre a COVID-19. O aplicativo possui os componentes e funcionalidades básicas para um aplicativo Android, como uso de formulários de cadastro, listas interativas, pesquisa e transição de dados de uma tela para outra.

Hoje o desenvolvimento para Android usa duas linguagens nativas oficiais, que são o Java e o Kotlin, ambas linguagens devem ser capazes de realizar as mesmas funcionalidades para a plataforma, mesmo que sejam implementadas de forma diferente seguindo a sintaxe da linguagem específica, por isso criou-se a necessidade de um estudo comparativo das principais diferenças na implementação e produtividade entre eles, para descobrir as vantagens de qual linguagem utilizar. Neste projeto conceito, onde foi desenvolvido o aplicativo nessas duas linguagens, foi possível mostrar as diferenças observadas na construção do aplicativo e como as características das linguagens podem interferir na produtividade do desenvolvimento.

Durante o desenvolvimento foi possível realizar a análise referente a sintaxe, quanto a forma de declarar variáveis, métodos, características de objetos nulos, instruções *switch*, funções lambdas e parâmetros opcionais. A diferença na construção de classe modelo para trabalhar com dados, e a implementação do padrão Singleton.

Também será apresentado o resultado e análise de um questionário realizado para saber como as comunidades de desenvolvedores estão perante essas linguagens.

2 METODOLOGIA

Como metodologia foi desenvolvido dois aplicativos para a plataforma Android, ambos com a mesma arquitetura e funcionalidades, porém em linguagens diferentes, onde foi possível realizar a análise durante a implementação na plataforma, baseado na

documentação de ambas linguagens, destacando as principais diferenças e características encontradas entre Java e Kotlin durante o desenvolvimento do aplicativo proposto neste artigo. Outro recurso utilizado, foi aplicar um questionário a diversos desenvolvedores Android, com o intuito de saber as opiniões em relação ao desenvolvimento com estas linguagens e uma análise das respostas.

3 DESENVOLVIMENTO ANDROID

O Android é uma plataforma de código aberto, baseado no kernel do Linux, teve seu desenvolvimento iniciado em 2003 pela Android Inc e em 2005 foi adquirida pela Google. Em 2007 foi criado uma associação de empresas de software, a Open Handset Alliance, cuja missão é desenvolver uma plataforma completa, aberta e gratuita para dispositivos móveis. Ainda no ano de 2007 ocorreu o lançamento da versão beta do primeiro SDK para Android. Desde que os fabricantes do Android liberaram o Kit de desenvolvimento de software (SDK), facilitou o surgimento de novos aplicativos, já que basicamente qualquer empresa ou pessoa poderia desenvolver para esse sistema, ganhando muitos adeptos. (MONTEIRO, 2013)

Para o desenvolvimento, normalmente utiliza-se o Android Studio, uma IDE oficial criada pela empresa JetBrains, nos aplicativos deste artigo foi utilizado a versão 3.6.3.

3.1 LINGUAGEM JAVA

Desde o início, o desenvolvimento nativo utiliza o Java como linguagem oficial, que já é uma linguagem que está no mercado há muito tempo e bastante disseminada, não somente para a plataforma Android. No aplicativo escrito em Java foi utilizado a versão 8, ela está no Android desde 2017, porém algumas funcionalidades do Java 8 para o Android, dependem da versão mínima da SDK do Android.

3.2 LINGUAGEM KOTLIN

A empresa JetBrains criou uma nova linguagem moderna chamada Kotlin, em 2011, para que seja produtiva e simples de aprender Em 2017 o time Android anunciou oficialmente suporte ao Kotlin tornando-a uma das linguagens oficiais para o desenvolvimento na plataforma android sdk. Em 2019, a Google realiza mais um grande passo, o movimento “Kotlin-first”, basicamente tornando a linguagem Kotlin como preferencial para desenvolvimento do Android. No aplicativo deste artigo escrito em Kotlin foi utilizada a versão 1.3.0

3.3 OUTRAS LINGUAGENS E FRAMEWORKS

No desenvolvimento Android também é possível utilizar outros frameworks e linguagens não nativas, mas que cumprem seu propósito. Dentre esses frameworks estão: Apache Cordova, React Native, que utilizam JavaScript para o desenvolvimento; o Flutter que utiliza Dart e o Xamarin, que utiliza o C#. Por não serem linguagens nativas oficiais, não foram objetos de estudo deste artigo, mas são utilizadas para o desenvolvimento de aplicativos android.

4 ESTUDO DE CASO

Para este trabalho, foi desenvolvido um aplicativo conceito nas duas linguagens, para que seja observado as diferenças e características de ambas as linguagens para o desenvolvimento.

4.1 APLICATIVO: BARREIRAS SANITÁRIAS

O aplicativo Barreiras Sanitárias, tem como objetivo fazer o cadastro de pessoas que passam pelo local da instalação da barreira sanitária, além do registro, a pessoa é convidada a responder um questionário referente aos sintomas da COVID-19. No aplicativo também é possível fazer o cadastro de localidades das barreiras sanitárias, com nome, localização e endereço, além de ter a pesquisa de já cadastrados, através do documento da pessoa, funcionalidade que permite responder novamente ao questionário.

Com as funcionalidades criadas, o aplicativo possui os componentes e funcionalidades básicas para um aplicativo Android, como uso de formulários de cadastro, listas interativas, pesquisa e transição de dados entre telas.

4.2 SINTAXE

Na programação, cada linguagem normalmente têm sua sintaxe diferente uma das outras, neste artigo será destacado as diferenças na declaração de variáveis, construção de métodos ou funções, instrução *Switch/Case*, uso de funções lambda, parâmetros opcionais, construção de classe modelo e implementação do padrão Singleton, pois durante o desenvolvimento dos aplicativos essas diferenças foram significativas conforme as análises a seguir.

As variáveis são constantemente utilizadas no desenvolvimento para armazenar vários tipos de dados na execução de um algoritmo. Nas linguagens utilizadas é possível perceber que a declaração são bem diferente conforme a Figura 1.

Figura 1. Exemplos de variáveis entre as linguagens.

Fonte: elaborado pelo autor

No Java é necessário declarar o tipo da variável (ex.: String, int, Long, Cidade), atribuir um valor a variável se necessário e caso seja uma variável imutável, usar a palavra reservada *final*. Já no Koltin, é possível declarar ou omitir o tipo de variável, assim o tipo é inferido pela atribuição de um valor, mas é necessário inicializar a variável. Também na declaração devemos usar a palavra reservada *val* (para valores imutáveis) ou *var* (para variáveis mutáveis).

Em Kotlin os tipos comuns são não nulos por padrão, não é possível atribuir *null* na variável idade do exemplo da Figura 1, evitando os erros comuns de *NullPointerException*, já a variável cidade possui um sinal de interrogação ao lado do

tipo, nesse caso indica que essa variável é *nullable,* ou seja, é possível atribuir valor nulo para ela.

Normalmente quando um algoritmo acessa uma propriedade de um objeto nulo resulta numa exceção como o *NullPointerException* no caso do Java. O Kotlin possui um mecanismo, para acessar uma propriedade que pode ser nula, de uma forma segura sem que ocorra a exceção. Na Figura 2 é possível ver que no código Java é necessário fazer a verificação primeiro se o objeto usuario é nulo para depois comparar uma propriedade dele, já no Kotlin basta usar a interrogação depois do objeto, dessa forma caso este objeto seja nulo, ele não ocorrerá a exceção.

Figura 2. Exemplos de métodos entre as linguagens.

Fonte: elaborado pelo autor

Outra funcionalidade importante do Kotlin observada na Figura 2, é a *Extension Functions*, no exemplo foi criado a extensão maiorQue no objeto *Long*, utilizado na propriedade *id*. Com essa extensão permite que todo *Long* tenha essa função, facilitando a comparação do número deixando o código mais limpo e claro, ao invés de ficar implementando a regra toda vez ou a criação de uma classe *Utils*. Até o momento no Java não possibilita a criação de *Extension Functions*.

Ainda sobre a Figura 2, podemos observar que para criar um método no Java, é necessário primeiro especificar o retorno, mesmo que não haja (no caso usar a palavra reservada *void*), depois o nome do método e os parâmetros (no padrão já mencionado). Já a declaração no Kotlin é necessário o uso da palavra reservada *fun*, depois o nome da função, os parâmetros (que podem ser opcionais, isso será mencionado mais à

frente) e o tipo de retorno, sendo que se a função não tiver retorno, não é necessário declarar.

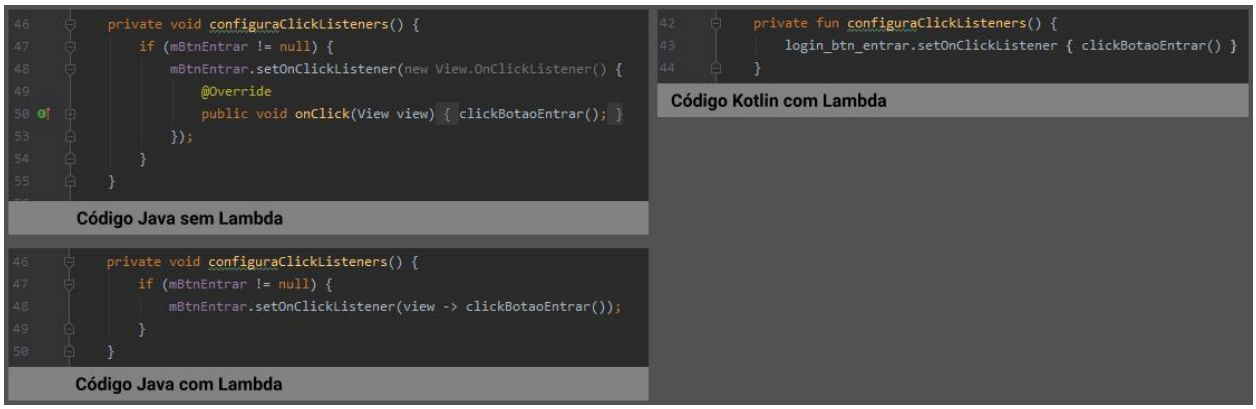
Figura 3. Comparação de instrução switch entre as linguagens.

Fonte: elaborado pelo autor.

O uso da instrução switch, se dá para evitar a complexidade da sequência de vários *ifs* e *elses* de modo que o código fique mais limpo. Na Figura 3, foi utilizado na classe DAOFactory o *switch* com possíveis casos para o enum TabelasDataBase, para verificar qual instância do *DAO* retornará. A principal diferença, é que o Kotlin usa a instrução *when*, apresenta uma sintaxe mais simples e com menos código, conforme apresentado na Figura 3. Como implementado o *enum* é um objeto não nulo e não há necessidade de aplicar um retorno *default*, uma vez que a IDE entende que todas as possibilidades do objeto não nulo foram explícitas nesta instrução. No Java é necessário informar o valor *default* e garantir que o enum do parâmetro não seja nulo.

As funções Lambdas já estão presentes em várias linguagens de programação, para agilizar o desenvolvimento e manter o código limpo, uma vez que: A grande vantagem de funções lambda é diminuir a quantidade de código necessária para a escrita de algumas funções (DEVMEDIA, 2020).

Figura 4. Comparação de funções Lambda.

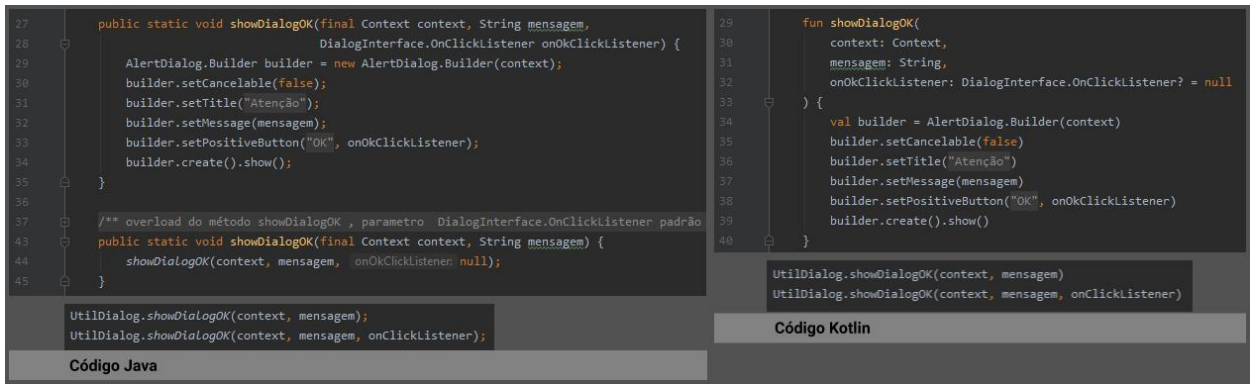


Fonte: elaborado pelo autor.

No Java as funções Lambdas só foram introduzidas a partir da versão 8 e a diferença na implementação pode ser percebida na Figura 4, com menos código e mais limpo, desde o início o Kotlin possui suporte a esse tipo de função.

Outra funcionalidade interessante nas linguagens de programação, são os parâmetros opcionais nos métodos, deixando o código mais limpo e prático, com valores padrões, caso não sejam passados. No exemplo utilizado na Figura 5, podemos ver a construção de uma caixa de diálogo, passando um evento de click padrão ou um evento que possa ser customizado pelo desenvolvedor ao utilizar esse método.

Figura 5. Comparação das classes Modelo entre as linguagens.



Fonte: elaborado pelo autor.

Ao observar no código Java, temos dois métodos com parâmetros diferentes que fazem a mesma coisa, visto que o segundo invoca o primeiro, foi utilizado a sobrecarga de métodos, permitindo que dois ou mais métodos tenham o mesmo nome com parâmetros diferentes, para que a chamada com o evento de *click* padrão seja feita, sem que a todo momento seja passado o valor padrão para o método.

No Kotlin é possível criar funções com valores padrões, desde que esses parâmetros sejam os últimos, conforme mostrado podemos invocar o método passando ou não, o último parâmetro, caso não seja passado, será atribuído o valor padrão definido automaticamente, eliminando a necessidade de uma sobrecarga.

Referente a construção do layout no Android, é comum usar a estrutura de XML disponível para a plataforma. Nos dois projetos foi utilizado esse método, que para as linguagens não é diferente e logo não houve impacto neste estudo comparativo. Existe em desenvolvimento, um kit de ferramentas que promete simplificar e acelerar o desenvolvimento de interfaces, chamado de Jetpack Compose para trabalhar com o Kotlin, no momento que este artigo foi escrito, esse kit está em fase *Preview* para desenvolvedores, por este motivo não foi objeto de estudo deste artigo.

Existem outras diferenças entre as sintaxes das classes mas no desenvolvimento dos aplicativos não foram utilizadas.

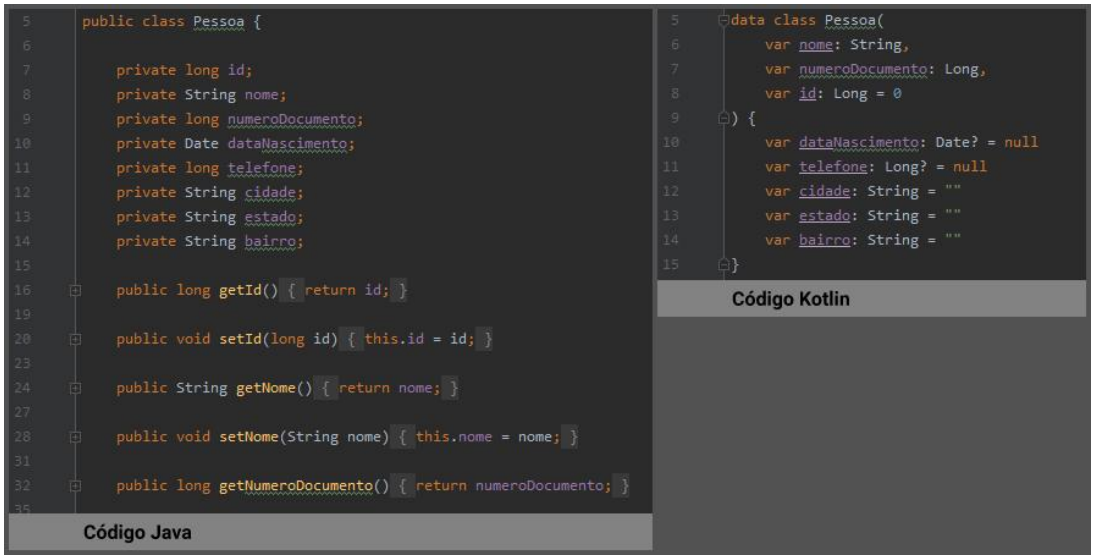
4.3 CLASSES MODELO

Um dos padrões utilizados na criação de sistemas para auxiliar a persistência de dados é o padrão Data Access Object conhecido como DAO (GUERRA, 2012), separando a lógica da aplicação da lógica de persistência, dessa forma mantendo o código mais limpo e possibilitando facilmente a alteração das classes responsáveis para conexão e manipulação da base de dados, caso necessário. Neste projeto foi utilizado este padrão, criando classes para representar o modelo de dados e suas propriedades, classes de acesso ao banco de dados e as interfaces que definem as operações a serem executadas.

Em ambas linguagens não houve problemas na implementação, além das diferenças básicas das linguagens, a construção das classes modelo tiveram diferenças,

assim como na implementação do padrão de projeto *Singleton*, nas classes *DAO*. A Figura 6 mostra o código comparativo do modelo entre as linguagens.

Figura 6. Comparação das classes Modelo entre as linguagens.



Fonte: elaborado pelo autor.

No projeto escrito em Java as classes modelos, foram construídas como objetos POJO (Plain Old Java Object), com atributos privados expostos por getters e setters. Por mais que as IDE's gerem esses métodos automaticamente, dependendo da quantidade de atributos, a manutenção e refatoração podem ser confusas, pela quantidade de código gerado. Vale destacar que na Figura 6 não foram exibidos todos os getters e setters gerados, a quantidade de linha geradas, contando com espaços foram de 52 linhas.

O Kotlin possui uma definição de classe específica para representação dos dados de forma estruturada, chamada de *Data Class*, além de permitir acesso direto às propriedades da classe, sem a utilização dos getters e setters, o compilador gera automaticamente o código das funções: *equals()*, *hashCode(), toString()* e *copy()*, de todas as propriedades declaradas no construtor primário, facilitando a manutenção e refatoração. Além de poucas linhas, o Kotlin conseguiu agregar mais funcionalidades e deixar o código da classe modelo mais limpo.

4.4 SINGLETON

No desenvolvimento dos aplicativos foi utilizado o padrão de projeto *Singleton*, principalmente nas classes de implementação *DAO* (*Data access object*), para evitar possíveis problemas de várias conexões com o banco de dados, visto que a intenção deste padrão é: “Garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso para a mesma.” (GAMMA, 2008, p.130)

A forma de implementação clássica deste padrão no Java, é deixar um único construtor da classe como privado, assim somente a própria classe tem acesso a ele de forma controlada e disponibiliza um método público para retorno da instância dele, garantindo somente uma instância da classe.

Com o Kotlin é possível criar um Singleton de forma mais fácil, basta utilizar a estrutura *object declaration*, ao invés de criar uma classe, utiliza-se a palavra reservada *object*, que o compilador irá gerar automaticamente um construtor privado. Nessa classe *object* ainda é possível ter propriedades, funções e métodos de inicialização.

Figura 7. Comparação das classes Modelo entre as linguagens.

Fonte: elaborado pelo autor.

Como podemos observar na Figura 7, para criar um *Singleton* com Kotlin, basta utilizar a estrutura *object*, utilizando apenas uma palavra, reduzindo tempo codificando, até prevenindo possíveis erros de digitação na implementação desse padrão de projeto.

4.5 PRODUTIVIDADE

Durante o desenvolvimento, foi utilizado o mesmo tipo de arquitetura e padrão para ambos aplicativos, para que não haja diferenças de implementação na plataforma, a não ser as próprias características das linguagens. A forma de desenvolvimento para a plataforma Android não teve alterações na implementação, foi mais uma questão da linguagem em si onde alguns pontos observados no desenvolvimento com as linguagens interferem na agilidade e praticidade da construção dos aplicativos.

Pode parecer simples mas alguns detalhes opcionais nas linguagens podem fazer diferença quando se escreve várias linhas de código. Por exemplo a obrigatoriedade de colocar um tipo de retorno em métodos que não tem retorno, poderia ser omitido, já que não há necessidade de informar que retorna vazio. No Kotlin, o uso do ponto e vírgula para delimitar as instruções é opcional, foi observado que se não utilizar essa pontuação não fez diferença em relação a legibilidade do código. Já no Java, seu uso é obrigatório, pois caso não utilize o ponto e vírgula ocorre um erro na compilação do projeto.

Escrevemos métodos o tempo todo durante a programação, e soluções aparentemente simples como parâmetros opcionais com valores padrão agilizam e simplificam muito o dia a dia do programador, poupando as vezes de ter que fazer várias sobrecargas para tornar alguns parâmetros opcionais mantendo maior legibilidade do código. As funções Lambdas do Kotlin e Java 8 também se mostraram excelentes para a produtividade quando se conhece bem, conforme mostrado na Figura 4 reduz o código e deixa mais limpo, porém a primeira vista pode causar uma certa estranheza.

Não ter que ficar fazendo verificações e validações o tempo todo de objetos nulos, poupa muito tempo de desenvolvimento, os objetos não nulos são bem mais produtivos e seguros de trabalhar, a possibilidade das chamadas seguras com esses objetos também evitam erros desnecessários e não se preocupar a princípio com o famoso *NullPointerException*.

Muitos projetos utilizam a estrutura de classe modelo, conforme visto neste artigo, o uso do *Data Class* do Kotlin foi extremamente produtivo, a redução de código gerado foi grande e facilitaria muito qualquer tipo de manutenção nesses tipos de classes, o

máximo de atributos que uma classe teve no artigo foi de doze, porém existem sistemas mais complexos com muito mais atributos, definitivamente faria muita diferença.

Naturalmente o profissional será mais produtivo na linguagem que ele terá mais domínio, mas no momento que o conhecimento é adquirido a produtividade vai depender mais das possibilidades da linguagem e a IDE utilizada. Se uma linguagem faz com que você escreva menos código com mais clareza, mais consistente e a IDE proporciona um bom suporte para isso, certamente esta será mais produtiva.

4.6 COMPILAÇÃO E DISTRIBUIÇÃO DO APLICATIVO

O foco principal do artigo é a análise durante o desenvolvimento referente às linguagens propostas, mas durante o desenvolvimento não foi percebido diferenças significativas entre o tempo de compilação nos projetos de cada linguagem, sendo que no Java a média foi de 557 milissegundos no Kotlin 597 milissegundos conforme Tabela 1, sendo que foi analisado alguns tempos de *build* dos projetos para a geração do arquivo APK (*Android Application Pack*), que é o arquivo de instalação do aplicativo.

TABELA 1 - Tempo de build do APK (milisegundos)

Linguagem	build 1	build 2	build 3	build 4	build 5
Java	615	546	643	541	441
Kotlin	1000	565	470	452	502

Fonte: elaborado pelo autor.

O APK gerado pela linguagem Java foi cerca de 1 *megabyte* menor do que gerado pelo Kotlin, ficando na média de 3 *megabytes* o tamanho.

5 QUESTIONÁRIO

Considerando que o objetivo é a comparação entre as duas linguagens foi elaborado um questionário com 10 perguntas, enviadas online para diversas comunidades de desenvolvedores no período 25/05/2020 à 10/06/2020 e obteve 72 respostas, com o foco sobre a opinião dos desenvolvedores sobre as linguagens Java e

Kotlin. As respostas do tipo “Outra”, refere-se a outras linguagens que não foram foco e objeto desse estudo, por isso não foram especificadas.

5.1 VOCÊ COMEÇOU A DESENVOLVER APPS ANDROID POR QUAL LINGUAGEM?

Java: 65 (90,3%) Kotlin: 3 (4,2%) Outra: 4 (5,6%)

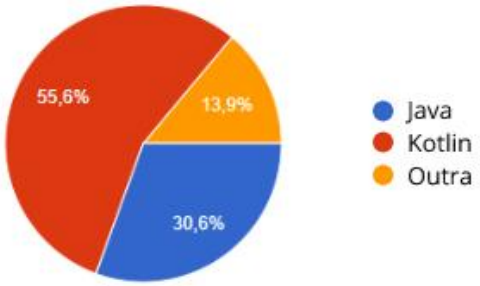
Muito comum os desenvolvedores terem iniciados pelo Java, pois é a linguagem nativa para desenvolvimento, desde o início do Android, o que pode demonstrar que o perfil da maioria, são desenvolvedores que estão há mais tempo na área. Lembrando que o Kotlin é uma linguagem recente e foi oficializado no ano de 2017.

5.2 QUAL LINGUAGEM VOCÊ CONSIDERA MAIS FÁCIL DE APRENDER?

Java: 22 (30,6%) Kotlin: 40 (55,6%) Outra: 10 (13,9%)

Mesmo as pessoas tendo iniciado no Java (90,3% segundo esta pesquisa) e estar no mercado de trabalho há mais tempo, não só para o Android, o Kotlin apareceu como a linguagem mais fácil de aprender. Mas o fato do número considerável de 30,6% dos desenvolvedores acharem o Java mais fácil de aprender pode ter uma relação com o início dos estudos, onde tenham aprendido ou trabalhado com Java primeiramente.

Figura 8. Linguagens mais fáceis de aprender.



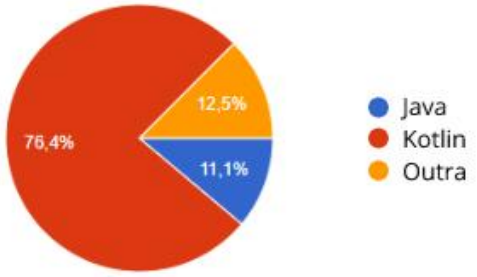
Fonte: elaborado pelo autor.

5.3 QUAL LINGUAGEM VOCÊ CONSIDERA MAIS PRÁTICA DE DESENVOLVER?

Java: 8 (11,1%) Kotlin: 55 (76,4%) Outra: 9 (12,5%)

Motivos pela praticidade do desenvolvimento do Kotlin, podemos citar os recursos modernos da linguagem, estruturas mais simples e códigos mais limpos. Interessante aqui comparar com a questão 2, que cerca de 54% dos participantes que consideram Java fácil de aprender, consideram o Kotlin mais prático para desenvolver.

Figura 9. Linguagens mais práticas para desenvolver.



Fonte: elaborado pelo autor.

5.4 NOS SEUS PROJETOS PESSOAIS QUAL LINGUAGEM VOCÊ ESTÁ USANDO?

Java: 12 (16,7%) **Kotlin: 49 (68,1%)** **Outra: 11 (15,7%)**

Por mais que 90,3% dos desenvolvedores tenham aprendido inicialmente Java, eles começaram a migrar para outras linguagens, restando somente 12,7% com essa linguagem nos projetos pessoais. Grande parte indo para o Kotlin e parcela indo para outras linguagens, mostra uma tendência de crescimento do Kotlin nos projetos.

5.5 QUAL NOTA VOCÊ ATRIBUI PARA O DESENVOLVIMENTO ANDROID UTILIZANDO JAVA?

Nota 1: 2 (2,9%) **Nota 2: 5 (7,1%)** **Nota 3: 22(31,4%)**

Nota 4: 25(35,7%) **Nota 5: 16(22,9%)**

As notas variam de 1 até no máximo 5.

O desenvolvimento com Java, teve a média de 3,68 com maiores votos nas notas 3 e 4, mas outros votos espalhados. Por mais que o Java já esteja a mais tempo no mercado, a pouca praticidade na linguagem, a média dessa nota pode ser um reflexo da questão 3, por não ser tão prática para o desenvolvimento.

5.6 QUAL NOTA VOCÊ ATRIBUI PARA O DESENVOLVIMENTO ANDROID UTILIZANDO KOTLIN?

Nota 1: 4 (5,6%) **Nota 2: 0 (0%)** **Nota 3: 2 (2,8%)**

Nota 4: 18 (25,4%) **Nota 5: 47 (66,2%)**

As notas variam de 1 até no máximo 5.

O desenvolvimento com Kotlin teve a média de 4,46 com maiores votos nas notas 4 e 5, o que indica que é uma boa pontuação. Com a praticidade de

desenvolvimento e facilidade de aprendizado, pode ser o reflexo para a nota média dessa pergunta.

5.7 DESDE QUANDO ESTÁ UTILIZANDO KOTLIN EM SEUS PROJETOS ANDROID?

Não utilizo: 21 (29,2%) 2016: 3 (4,2%) 2017: 12 (16,7%)
2018: 15 (20,8%) 2019: 17 (23,6%) 2020: 4 (5,6 %)

Cerca de 30% das pessoas estão utilizando Java ou outra linguagem para projetos próprios, os outros 70% utilizam Kotlin. Podemos perceber um crescente uso da nova linguagem, considerando que a mesma só foi oficializada em 2017 e já em 2019 onde teve o maior crescimento, foi realizado o movimento *"Kotlin-first"*.

5.8 ATUALMENTE NA SUA EMPRESA ESTÁ USANDO QUAL LINGUAGEM NO DESENVOLVIMENTO DE APPS ANDROID?

Java: 21 (29,2%) Kotlin: 42 (58,3%) Outra: 9 (12,5%)

Podemos perceber que mesmo sendo uma linguagem recente, o Kotlin já está bem vivo no mercado de trabalho, até mesmo pelo fato observado nas outras questões, indicando como sendo uma linguagem prática e fácil de aprender, que afeta a produtividade dos desenvolvedores.

5.9 EXISTEM PLANOS DE MUDANÇA DA LINGUAGEM DOS APP ANDROID DA SUA EMPRESA, PARA ALGUMA LINGUAGEM?

Java: 0 (0%) Kotlin: 14 (19,4%) Outra: 8 (11,1%)
Sem planejamento: 50 (69,4%)

Podemos perceber que 69,4% dos participantes responderam que sua empresa não têm planos de alteração da linguagem atual. Mas o que chama mais atenção é os 0% de votos para Java, ou seja, nenhuma das intenções registradas de planejamento é para a linguagem.

5.10 NAS ULTIMAS VAGAS PARA DESENVOLVEDOR ANDROID QUAIS LINGUAGENS VOCÊ VIU QUE ERA REQUISITO?

Java: 53 (73,2%) **Kotlin: 55 (77,5%)** **Outra: 24 (33,8%)**

Essa questão poderia selecionar mais de uma resposta.

Em muitos casos o Java e Kotlin eram selecionadas em conjunto, o que indica que nas vagas de emprego, procuram profissionais que têm conhecimento principalmente nas linguagens Java e Kotlin. Segundo a questão 9, não há planos para mudar os aplicativos para Java e essa procura no mercado de trabalho Android, provavelmente é para dar continuidade nas aplicações já iniciadas em Java ou para fazer a migração de linguagem.

6 CONSIDERAÇÕES FINAIS

Como resultado final do presente trabalho, foram desenvolvidos dois aplicativos com a mesma estrutura e funcionalidades, porém com linguagens diferentes, para análise do desenvolvimento e produtividade das linguagens oficiais e nativas para plataforma Android: Java e Kotlin. Além da análise quanto à programação, um questionário com opiniões de diversos profissionais de várias comunidades para reforçar o estudo sobre as linguagens propostas.

O presente trabalho objetivou apresentar diferenças das linguagens Java e Kotlin, através da implementação de um aplicativo conceito para cada umas das linguagens. Com os aplicativos contendo funcionalidades básicas de implementação, porém comuns, podemos observar diferenças significativas em vários pontos durante a programação, onde na maioria refere-se a manter um código menos verboso, limpo e mais consistente.

Dentre as linguagens estudadas neste artigo, o Kotlin se mostrou mais produtivo no desenvolvimento do aplicativo, possuindo recursos únicos, como classes e estrutura específicas para atender problemas comuns e mais fluidez no escrita do código com recursos opcionais na linguagem. Um problema inicialmente enfrentado, é a estranheza

que a linguagem pode causar, para quem está migrando do Java para o Kotlin, algumas anotações diferentes na declaração de variáveis, funções, a forma como se instancia uma classe, implementa uma interface. Mas logo com a prática você se familiariza com esses detalhes e acabará escrevendo o seu código Java mais parecido com o Kotlin, já que provou ser uma linguagem mais fluida para a codificação.

Com base nas respostas do questionário aplicado, podemos reforçar que o Kotlin é uma linguagem prática e fácil de aprender, os desenvolvedores realmente estão utilizando em seus projetos pessoais e já é realidade no mercado de trabalho, mesmo que para uma linguagem recente.

Este trabalho não explorou todas as funcionalidades possíveis na criação de aplicativos com ambas as linguagens, diversas outras sintaxes podem ser aplicadas de acordo com as necessidades. Uma vez que as linguagens de programação e a plataforma Android sofrem atualizações com melhorias, o que pode proporcionar um futuro artigo com novas versões do ambiente de programação para a plataforma.

REFERÊNCIAS

GAMMA, R. R. J. et al. **Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos**. bookamn 2008

GUERRA, EDUARDO. **Designer Pattern com Java**, 2012. Casa do Código

MONTEIRO, BOSCO. **Google Android. Crie Aplicações Para Celulares e Tablets**. Casa do Código 2013

Usar recursos da linguagem Java 8 **ANDROID DEVELOPERS**, 2020. Disponível em: <https://developer.android.com/studio/write/java8-support#supported_features> Acesso em: 10 de junho de 2020.

Using Kotlin for Android Development **KOTLIN**, 2020. Disponível em: <https://kotlinlang.org/docs/reference/android-overview.html> Acesso em: 5 de junho de 2020.

Jetpack Compose **ANDROID DEVELOPER**, 2020. Disponível em: <https://developer.android.com/jetpack/compose> Acesso em: 5 de junho de 2020.

Como usar funções lambda em Java **DevMedia**, 2020. Disponível em: <https://www.devmedia.com.br/como-usar-funcoes-lambda-em-java/32826> Acesso em: 5 de junho de 2020.

Android Announces Support for Kotlin **Android Developers Blog**, 2017 Disponível em: <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html> Acessado em: 10 de junho de 2020.

Android's commitment to Kotlin **Android Developers Blog**, 2019 Disponível em: <https://android-developers.googleblog.com/2019/12/androids-commitment-to-kotlin.html> Acessado em: 10 de junho de 2020.

Java 8 Language Features Support Update **Android Developers Blog**, 2017 Disponível em: <https://android-developers.googleblog.com/2017/04/java-8-language-features-support-update.html> Acessado em: 10 de junho de 2020.

Alterações de comportamento do Android 5.0 **Android Developers**, 2020 Disponível em: <https://developer.android.com/about/versions/android-5.0-changes> acessado em: 24 de junho de 2020.

Why JetBrains needs Kotlin **Kotlin Blog**, 2011 Disponível em: <https://blog.jetbrains.com/kotlin/2011/08/why-jetbrains-needs-kotlin/> acessado em: 24 de junho de 2020.

Hello World **Kotlin Blog**, 2011 Disponível em: <https://blog.jetbrains.com/kotlin/2011/07/hello-world-2/> acessado em: 24 de junho de 2020.

Coronavírus: Sobre a doença **Ministério da Saúde**, 2020 Disponível em: <https://coronavirus.saude.gov.br/sobre-a-doenca> Acesso em: 22 de junho de 2020.

Diferença entre os patterns PO, POJO, BO, DTO e VO **DevMedia**, 2020 Disponível em <https://www.devmedia.com.br/diferenca-entre-os-patterns-po-pojo-bo-dto-e-vo/28162> acessado em: 24 de junho de 2020.

Código fonte das aplicações em Java e Kotlin **app-java-x-kotlin**, 2020 Disponível em <https://github.com/fbvictorhugo/app-java-x-kotlin/releases/tag/tcc> acessado em: 4 de julho de 2020.