revista MASTER ensino, pesquisa e extensão

Vol. 11. | Núm. 21 | Ano 2026

revistamaster.imepac.edu.br

IMEPAC

Comparação de Desempenho entre Kotlin e Java em Aplicações Back-End

Performance comparison between Kotlin and Java in Back-end applications

DOI: 10.47224/revistamaster.v11i21.660

Alfredo José Tadeu de Oliveira.

Bruno Antonio Joanucci.

Lucas Moreira Silva.

Walter Rodrigues de Andrade.

e-mail: lucas.moreira@aluno.imepac.edu.br

Resumo

No cenário atual do desenvolvimento back-end, Java é a linguagem dominante, enquanto Kotlin, inicialmente focado no Android, se sobressai por sua clareza, proteção contra nulidades e total compatibilidade com a JVM, no entanto seu desempenho não foi suficientemente estudado. O objetivo principal desta pesquisa é comparar a performance e a usabilidade de aplicações REST criadas em Java e Kotlin, além de analisar a curva de aprendizado e o efeito na manutenção de sistemas já existentes. A metodologia adotada consiste na implementação de APIs idênticas em ambas as linguagens as quais são submetidas a testes de carga, avaliando o tempo de resposta e a utilização da CPU, com cada experimento sendo repetido para garantir a confiabilidade.

Palavras-chave: linguagem de programação; benchmark; análise de performance;Kotlin;Java

Abstract

In the current back-end development landscape, Java remains the dominant language. Meanwhile, Kotlin, initially focused on Android, has emerged as a strong contender, notable for its conciseness, null-safety features, and full JVM compatibility. However, its performance in server-side applications has not yet been sufficiently investigated. The main objective of this research is to compare the performance and usability of REST APIs developed in Java and Kotlin. Additionally, it analyzes the learning curve and the impact on the maintenance of existing systems. To achieve this, identical APIs were implemented in both languages and subjected to load tests, evaluating response time and CPU utilization. Each experiment was repeated to ensure the reliability of the results.

Keywords: programming language; benchmark; performance analysis; Java; Kotlin

1 INTRODUÇÃO

No desenvolvimento de aplicações back-end, a escolha da linguagem de programação representa um fator importante que pode influenciar diretamente a eficiência, manutenção, recursos e escalabilidade das aplicações. Dentre as linguagens amplamente utilizadas para o desenvolvimento back-end, o Java, um pilar do mundo da programação há décadas, continua a servir como uma linguagem fundamental em aplicações de nível empresarial, devido à sua versatilidade, grande comunidade ativa e grande diversidade de bibliotecas e frameworks.(Reddy et al., 2022)

Nos últimos anos, no entanto, o Kotlin tem ganhado notoriedade, inicialmente impulsionado pelo suporte oficial do Google para o desenvolvimento Android. Com recursos como concisão, segurança contra nulidade e interoperabilidade total com Java, o Kotlin também tem despertado o interesse da comunidade para aplicações em servidores.

Apesar do volume de estudos comparativos entre Java e Kotlin no contexto de aplicações móveis, há poucas investigações empíricas voltadas especificamente para aplicações back-end REST. Muitos estudos existentes concentram-se em cenários de testes sintéticos como em (Glukhov & Mullayanov, 2020), que comparou as linguagens utilizando testes do Computer Language Benchmarks Game (CLBG), como Fasta e N-body. Embora valiosos, esses testes não refletem necessariamente a complexidade e as características de um serviço web em produção. Essa lacuna justifica a condução de testes experimentais que permitam comparar diretamente as implementações Java e Kotlin neste cenário.

Diante deste contexto, este artigo busca suprir essa lacuna, realizando uma análise comparativa do desempenho de APIs REST implementadas em Java e Kotlin utilizando o framework Spring Boot. A avaliação é centrada nas métricas de tempo de resposta e consumo de CPU, que são fundamentais para aplicações que exigem alto desempenho e escalabilidade.

2 METODOLOGIA

A metodologia adotada neste trabalho baseia-se em uma abordagem quantitativa e comparativa , com o objetivo de avaliar o desempenho de operações básicas de persistência em um sistema desenvolvido na linguagem Java e Kotlin. Ambas as aplicações serão APIs REST simples, contendo a mesma estrutura, funcionalidades e lógica de programação, diferenciando-se apenas pela linguagem utilizada. Essa abordagem permite uma comparação justa entre as duas tecnologias, mantendo todos os demais fatores constantes.

As aplicações serão submetidas a testes de desempenho, nos quais serão executadas requisições simuladas em uma máquina. Durante os testes, serão coletadas duas métricas principais: o tempo de resposta das requisições e o uso de CPU durante a execução da aplicação.

Para garantir a confiabilidade dos dados, cada teste será repetido várias vezes. Dessa forma, será possível identificar possíveis variações anormais e descartar resultados fora do padrão causados por eventos inesperados. Os dados obtidos serão registrados e analisados comparativamente, com foco em identificar as vantagens e desvantagens de cada linguagem no contexto de aplicações back-end.

2.1 AMBIENTES DE TESTE E MATERIAIS UTILIZADOS

Todos os experimentos foram executados em um ambiente de hardware e software controlado para garantir a consistência e a reprodutibilidade dos resultados. Ambientes de teste utilizados:

Processador: 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz.

Memória RAM: 16,0 GB (utilizável: 15,7 GB).

Armazenamento: SSD SATA 1TB.

Software e Ecossistema:

Sistema Operacional: Windows 11 24H2.

Runtime: OpenJDK 24 com JVM 21.

Aplicações: Duas APIs REST funcionalmente idênticas, uma desenvolvida em Java e outra em Kotlin.

Framework: Spring Boot 3.5.0.

Banco de Dados: MySQL.

Ferramenta de Teste: Postman (para o envio automatizado de requisições HTTP).

2.2 PROTOCOLO EXPERIMENTAL

O protocolo de testes foi aplicado de forma idêntica nas duas aplicações, de forma a reduzir a influência de fatores externos e garantir a consistência dos resultados. Foi estabelecido um número total de [número de execuções] execuções para cada um dos testes descritos abaixo.

Teste de Inicialização: O tempo de startup foi medido a partir do início do processo da aplicação até o momento em que a aplicação sinalizava em seus logs que estava pronta para receber requisições.

Teste de Persistência: Foram enviadas requisições HTTP do tipo POST para endpoints responsáveis por receber e persistir no banco de dados os objetos `vagão` e `encoste`.

Teste Computacional: Foram acionados endpoints que executavam dois algoritmos computacionalmente intensivos: o cálculo de Fatorial por recursão e o Crivo de Eratóstenes.

2.3 COLETA DE DADOS E MÉTRICAS

A coleta de dados para os testes de desempenho foi realizada de forma automatizada. Os testes foram executados por meio de requisições HTTP utilizando a ferramenta Postman. Para a medição do uso de CPU e tempo de execução, empregaram-se os recursos do Spring Boot Actuator, bem como as classes ThreadMXBean e ManagementFactory, ambas da biblioteca padrão do Java. As métricas específicas coletadas em cada execução foram:

Tempo de Inicialização: Medido em segundos (s).

Teste de Resposta da Requisição: Tempo total da transação, medido em segundos (s) pela ferramenta Postman.

Uso de CPU: Percentual de utilização do processador atribuído ao processo da aplicação durante a execução do teste.

3 RESULTADOS E DISCUSSÃO

Esta seção apresenta e analisa os resultados obtidos a partir dos testes realizados nas APIs REST desenvolvidas em Java e Kotlin, com foco em cinco categorias: tempo de inicialização, requisições POST de persistência para dois tipos de objetos, cálculo de fatorial recursivo e execução do Crivo de Eratóstenes. As métricas comparadas foram o tempo de execução e o uso de CPU por operação.

3.1 TEMPO DE INICIALIZAÇÃO

Usando as mesmas condições de ambiente de execução para ambas as aplicações, foi medido o tempo de inicialização. O tempo médio de inicialização da aplicação Java foi de 25,67 segundos, enquanto a aplicação Kotlin apresentou média de 30,78 segundos. Isso representa uma vantagem de aproximadamente 16,6\% em favor da aplicação Java.

Essa diferença pode estar relacionada ao tempo de carregamento adicional introduzido pelas bibliotecas Kotlin na inicialização da JVM, que, embora otimizadas, ainda representam uma camada extra.

3.2 OPERAÇÕES DE PERSISTÊNCIA (POST)

Foram realizadas requisições POST para salvar dois tipos de objeto em um banco de dados MySQL. O objeto 1 consiste de uma classe com 2 Enums, 3 Longs (sendo 1 deles o atributo usado como identificador no banco de dados) e 1 String entre seus atributos. O objeto 2 possui como atributos apenas 1 Long que é seu id e 1 List de tamanho variado de objeto 1. Em cada requisição, foram enviados uma grande quantidade de objetos para serem salvos, de forma que em uma requisição de objeto 1 foram salvos 2000 objetos e na requisição do objeto 2 foram salvos 600 objetos.

Durante o envio de requisições POST para persistência do primeiro objeto, a aplicação Java apresentou média de 5,86 s de tempo de resposta com uso de CPU de 12\%. A versão Kotlin, por outro lado, teve tempo médio de 8,51 s e uso de CPU de 14,73%. Isso representa uma diferença de desempenho de 31,18% a favor de Java no tempo de resposta e 18,61% de economia de CPU.

Resultados semelhantes foram observados na persistência do segundo objeto: tempo médio de 6,18 s para Java contra 8,00 s para Kotlin, com uso de CPU de 6,27% e 6,08%, respectivamente. Embora o tempo de resposta tenha sido novamente inferior ao da versão Java (22,79% mais rápido), o uso de CPU foi levemente superior (3% a mais que Kotlin).

Esses resultados sugerem que, apesar da eficiência sintática do Kotlin, o bytecode gerado pode impactar negativamente a performance em operações de persistência, possivelmente por diferenças no comportamento do compilador ou no Just-In-Time da JVM;

3.3 TESTE COMPUTACIONAL: FATORIAL RECURSIVO

O endpoint que calcula fatorial recursivamente apresentou resultados mais próximos entre as linguagens. Java executou a operação em média 0,319s, contra 0,391s da versão Kotlin — um ganho de 18,1% para Java. O uso de CPU foi 9,01% (Java) e 9,78% (Kotlin), uma diferença de apenas 7,8%, mas ainda favorável a Java.

A semelhança nos resultados pode indicar que, em tarefas puramente algorítmicas com poucas operações de I/O ou uso de infraestrutura externa, ambas as linguagens aproveitam de forma semelhante os recursos da JVM.

3.4 TESTE COMPUTACIONAL: CRIVO DE ERATÓSTENES

No teste do Crivo de Eratóstenes, a diferença foi mais acentuada. Java concluiu em média 2,65 s, enquanto Kotlin precisou de 3,32 s — uma diferença de 20,3%. No consumo de CPU, Java obteve 20,2% contra 29,8% de Kotlin, totalizando uma diferença de mais de 32% no uso da CPU.

Esses dados reforçam a hipótese de que operações intensivas, mesmo em contextos controlados, ainda podem sofrer variações consideráveis em Kotlin devido a questões de otimização de bytecode e comportamento do compilador JIT.

3.5 ANÁLISE GERAL

A Tabela 1 resume os resultados médios e o percentual de vantagem do Java sobre o Kotlin:

Tabela 1 – Desempenho comparativo de Java e Kotlin em inicialização, POST de objetos e algoritmos, em ARAGUARI(MG), entre 1 e 5 de junho de 2025.

Operação	Java (Tempo)	Kotlin (Tempo)	Diferença (%)	Java (CPU)	Kotlin (CPU)	Diferença (%)
Inicialização	25,67 s	30,78 s	16,6%	-	-	-
POST Objeto 1	5,86 s	8,51 s	-31,2%	12,0%	14,7%	-18,6%
POST Objeto 2	6,18 s	8,01 s	-22,8%	6,3%	6,1%	+3,0%
Fatorial recursivo	0,32 s	0,39 s	-18,1%	9,0%	9,8%	-7,8%
Crivo de Eratóstenes	2,65 s	3,32 s	-20,3%	20,2%	29,8%	-32.2%

Fonte: Autores

No panorama geral, Java apresentou vantagem de 21,80% em tempo de execução e 13,89% de economia de CPU, considerando a média dos testes. Isso mostra que, mesmo utilizando o mesmo framework (Spring Boot) e executando em ambientes equivalentes, a linguagem Kotlin ainda apresenta overheads relevantes em cenários de carga.

4 CONCLUSÕES

Este trabalho apresentou uma análise comparativa entre as linguagens Java e Kotlin no desenvolvimento de APIs REST para aplicações back-end, utilizando o mesmo framework (Spring Boot) e arquitetura de software. Por meio da execução de testes experimentais controlados, foi possível medir e comparar métricas-chave como tempo de resposta, uso de CPU e tempo de inicialização.

Os resultados mostraram que, embora Kotlin ofereça vantagens como concisão de código e segurança contra nulidades, a aplicação Java apresentou desempenho superior em todos os testes realizados. Em média, Java foi aproximadamente 21,8% mais rápido e consumiu 13,89% menos CPU nas operações avaliadas. Essa diferença foi particularmente perceptível em tarefas computacionalmente intensivas, como o Crivo de Eratóstenes, e em requisições de persistência de dados, nas quais Kotlin apresentou tempos de resposta significativamente maiores.

As causas dessas diferenças podem estar relacionadas ao bytecode gerado pelos compiladores de cada linguagem e ao comportamento do compilador JIT da JVM, que pode otimizar de forma mais eficiente o código Java, como já sugerido na literatura especializada.

Como contribuição, este estudo oferece dados experimentais objetivos que podem auxiliar desenvolvedores e equipes de engenharia na escolha da linguagem mais adequada para projetos back-end com requisitos de desempenho. Além disso, destaca a importância de não considerar apenas aspectos sintáticos ou de produtividade na escolha de tecnologias, mas também seu impacto direto no consumo de recursos e tempo de execução.

Para trabalhos futuros, sugere-se ampliar os testes com outros tipos de operações, como concorrência, escalabilidade em ambiente distribuído, e consumo de memória, além de investigar a performance com diferentes versões da JVM e explorar o uso de ferramentas de profiling como o JITWatch para análise mais profunda do código nativo gerado.

5 REFERÊNCIAS

FLAUZINO, M. et al. Are you still smelling it? A comparative study between Java and Kotlin language. In: **Proceedings of the VII Brazilian Symposium on Software Components, Architectures, and Reuse**. New York: Association for Computing Machinery, 2018. p. 23-32.

GLUKHOV, D.; MULLAYANOV, B. The Performance Evaluating of Kotlin and Java Implementations. In: **2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)**. New York: IEEE, 2020. p. 1-7.

MONTOYA-GAMEZ, Arnoldo. **Benchmarking Java and Kotlin in Android Runtime Environment**. 2020. Dissertação (Mestrado em Engenharia da Computação) – Iowa State University, Ames, 2020.

NAKAMURA, Hayataka *et al.* Performance Study of Kotlin and Java Programs with Bytecode Analysis. **Journal of Information Processing**, v. 32, p. 380-395, 2024.

PUTRANTO, Bambang Purnomosidi Dwi et al. A Comparative Study of Java and Kotlin for Android Mobile Application Development. In: **2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)**. New York: IEEE, 2020. p. 383-388.

REDDY, Hemasundara *et al*. Optimizing Java applications with advanced functional programming: a comparative analysis of Java, Scala, and Kotlin. **Journal of Informatics Education and Research**, [S. l.], v. 5, n. 2, 2022. DOI: 10.52783/jier.v5i2.2466. Disponível em: https://www.jier.org/index.php/journal/article/view/2466/2031. Acesso em: 25 jun. 2025.