RELATÓRIO FINAL DE PROJETO DE PESQUISA – EDITAIS PROPPI e CÂMPUS

IDENTIFICAÇÃO DO EDITAL	Nº: 4	Ano: 2024
	Título: PROPPI – Fluxo Contínuo	
	Edital da PROPPI	(X)
	Edital de câmpus	( )

1. IDENTIFICAÇÃO DO PROJETO	
Título do projeto	Benchmarking de APIs em Diferentes Ambientes de Programação - Sistema Financeiro de Débitos e Créditos
Coordenador(a)	Leonardo Bravo Estácio
Discentes pesquisadores Bolsista(s) e/ou voluntários	1 – Fabricio Iaronka de Souza 2 – Vinícius Souza Corbellini
Câmpus	Lages

2. RELATÓRIO TÉCNICO

2.1. Resumo do projeto

Implementar uma API RESTful nas linguagens Rust, JavaScript, Java e Python, seguindo especificação funcional e arquitetura padronizada. Utilizar duas instâncias da API, balanceador de carga com algoritmo round-robin e banco PostgreSQL compartilhado, assegurando condições operacionais uniformes. Aplicar restrições de recursos, com limites idênticos de processamento e memória. Executar testes de carga com alto volume de requisições concorrentes, repetindo os experimentos para garantir validade estatística. Mensurar latência, taxa de erros, consistência dos dados e consumo de CPU e memória. Comparar os resultados, avaliando desempenho, eficiência e estabilidade sob alta concorrência. Gerar evidências empíricas e estatísticas para apoiar decisões na escolha de linguagens e tecnologias voltadas ao desenvolvimento de APIs escaláveis, eficientes e resilientes, contribuindo com a engenharia de software e com práticas de benchmarking de sistemas.

2.2. Resultados alcançados

As metas previstas foram cumpridas com a entrega de quatro implementações funcionalmente equivalentes da API (Java/GraalVM/Quarkus, JavaScript/Node/Fastify, Python/Uvicorn/FastAPI e Rust/Tokio/Axum), executadas em duas instâncias balanceadas (app1/app2) atrás de Nginx e sobre um SGBD relacional mantido fixo. O ambiente foi padronizado com limites idênticos de recursos, manifestos de versão e procedimentos de *health check/readiness*, assegurando controle experimental e reprodutibilidade.

As campanhas de teste foram conduzidas em modelo aberto, com fases de aquecimento, regime permanente e resfriamento. Foram coletadas séries temporais de latências (p50, p95, p99), *throughput*, taxa de erros e uso de recursos, acompanhadas de verificações de consistência lógica (idempotência). A análise estatística empregou *bootstrap* para intervalos de confiança e testes não

paramétricos para comparar variantes sob a mesma carga, permitindo identificar faixas estáveis de operação, pontos de saturação e o impacto das estratégias de acesso a dados (pooling, *prepared statements* e formato de transporte) sobre as caudas de latência.

Como produtos, o projeto disponibilizou implementações de referência, *compose* e manifestos para reexecução, scripts de automação e coleta, além de *datasets* e sumários organizados para análise. A síntese dos achados resultou em diretrizes técnicas voltadas à redução da variabilidade em p95/p99, ao aumento da previsibilidade em produção e ao ajuste de parâmetros operacionais (admissão, *timeouts* e *backpressure*).

A participação dos bolsistas ocorreu em todas as etapas: preparação do ambiente, instrumentação com Gatling, execução controlada dos experimentos, consolidação das séries e aplicação dos métodos estatísticos. Esse envolvimento gerou contribuições formativas diretas em engenharia de desempenho de software, integração contínua entre ensino, pesquisa e extensão e produção de materiais técnico-científicos orientados por evidências.

A organização dos dados seguiu procedimento padronizado. Cada execução foi tratada como conjunto independente e segmentada em blocos numerados (Run 1…N). Em cada bloco, consolidaram-se as cinco categorias de requisições (All Requests, validações, extratos, créditos, débitos) e calculou-se um “p99 médio”, definido como a média aritmética dos p99 dessas categorias. Com base nesse critério, selecionaram-se os Top-5 blocos de cada execução, isto é, os cinco intervalos com maior p99 médio, representando caudas mais lentas de forma consistente.

A partir desses Top-5, para cada operação e para cada métrica (p95, p99 e média), calcularam-se dois valores: (i) a média da métrica nos cinco blocos selecionados (altura da barra) e (ii) o desvio-padrão correspondente (barra de erro), refletindo a variabilidade entre os blocos críticos. Em seguida, elaboraram-se gráficos de barras agrupadas por operação, com uma cor por execução, grade no eixo Y e rótulos descritivos para favorecer a comparação.

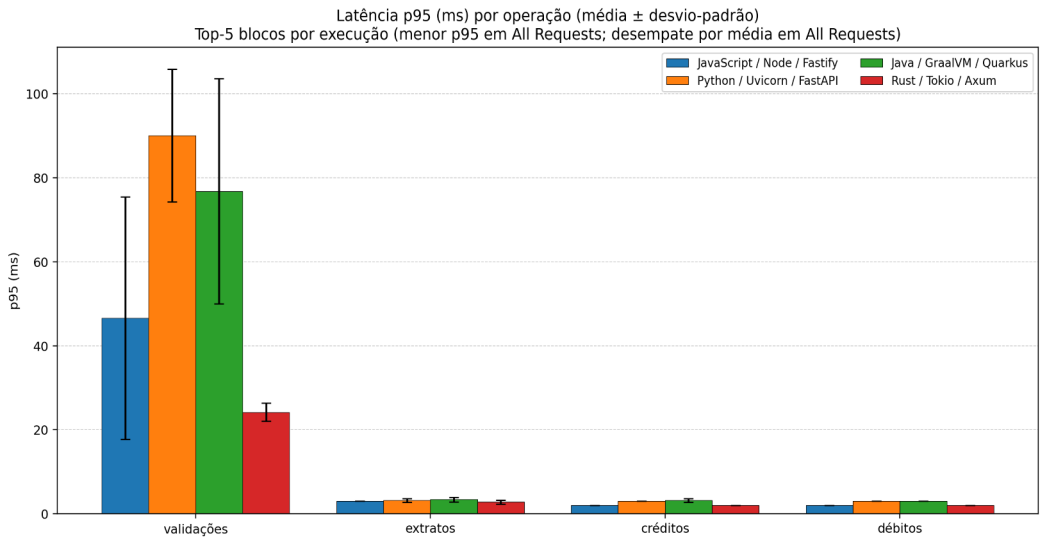


Figura 1: Gráfico p95

A Figura 1 compara a latência p95 (ms) por operação entre as execuções, restrito aos Top-5 blocos definidos pelo p99 médio. Cada grupo do eixo X corresponde a uma operação; cada barra colorida, a uma execução. A altura indica o p95 médio nos cinco blocos críticos; a barra de erro mostra o

desvio-padrão. O recorte evidencia o "pior cenário consistente" de p95, permitindo avaliar estabilidade e degradação por operação.

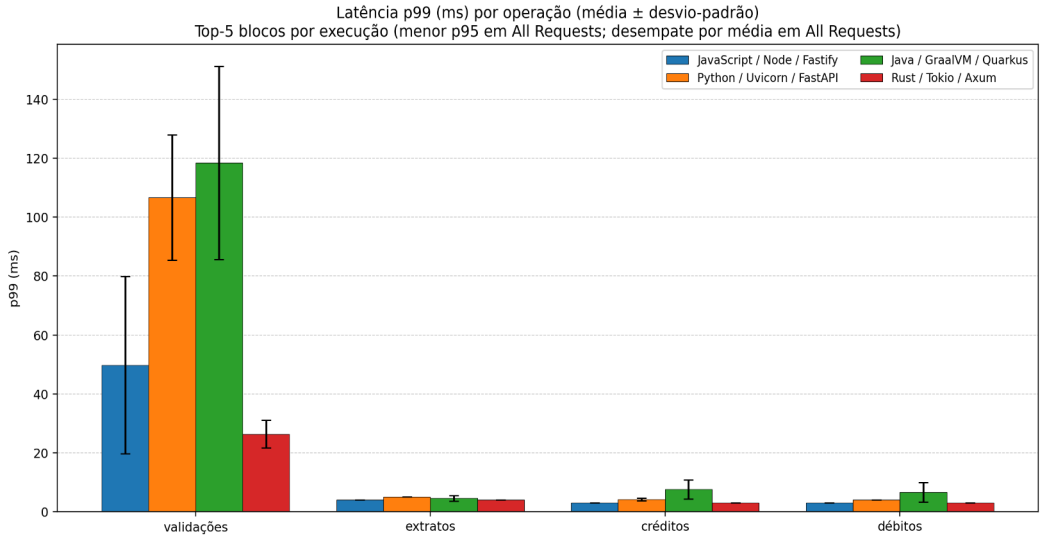


Figura 2: Gráfico p99

A Figura 2 aplica a mesma lógica ao p99. Por capturar a cauda, o p99 está diretamente alinhado ao critério de seleção dos Top-5, realçando diferenças de cauda entre execuções e operações. Barras mais altas indicam caudas mais lentas; barras de erro maiores sinalizam maior instabilidade entre blocos críticos.

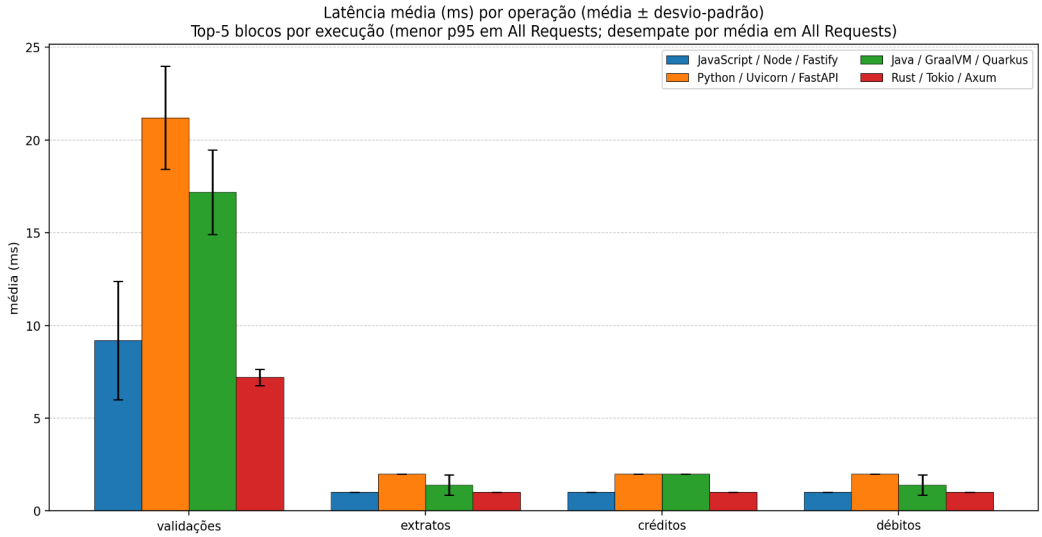


Figura 3: Gráfico de latência média

A Figura 3 compara a latência média (ms) por operação, também apenas sobre os Top-5. Como a média é menos sensível a eventos raros do que o p99, o contraste entre ambas permite distinguir efeitos concentrados na cauda (piora no p99 sem elevação proporcional da média) de degradações mais globais do tempo de resposta (piora concomitante de média e p99).

2.2.1 Resultados não alcançados ou parcialmente alcançados e suas justificativas

A variante em Go, prevista no planejamento, não foi implementada. Priorizou-se a maturidade e a comparabilidade das quatro stacks efetivamente utilizadas (Java, JavaScript/Node, Python, Rust), evitando ampliar o escopo sem ganho proporcional de evidência e preservando cronograma e controle experimental.

Alguns ensaios foram parcialmente executados: (i) avaliação exaustiva de formatos de transporte binário/"row mode", devido a suporte desigual entre *drivers*; (ii) matriz completa de injeção de falhas (latência, perda, *jitter*) e *backoff*, substituída por testes exploratórios. As decisões visaram manter isonomia entre variantes e estabilidade estatística das rodadas em regime permanente. As conclusões centrais (p95/p99, *throughput*, erros) permanecem válidas e as lacunas mapeiam extensões para trabalhos futuros.

2.3. Produção técnica e bibliográfica gerada com o projeto

Foi publicada a versão em resumo do estudo e realizada a divulgação científica em evento institucional. O trabalho intitulado *Benchmarking de APIs em Diferentes Ambientes de Programação* foi apresentado na VII Mostra "3 Minutos de Ciência e Tecnologia" do IFSC (formato vídeo e pôster), conforme certificado emitido pela Coordenação de Pesquisa e Inovação, que atesta autoria de Fabrício Iaronka de Souza, Vinícius Souza Corbellini e Leonardo Bravo Estácio.

A produção técnica associada incluiu a apresentação oral com slides para a sessão "3 Minutos de Ciência e Tecnologia", que sintetiza objetivos, metodologia, métricas e conclusões do *benchmark* comparativo entre as implementações avaliadas; este material compôs o suporte à comunicação dos resultados junto ao público do evento.

Como produto bibliográfico e de divulgação, foi elaborado e exibido pôster científico do trabalho, contendo introdução, objetivo, metodologia padronizada em ambiente Docker com instâncias balanceadas e PostgreSQL compartilhado, resultados comparativos (latência média, p95/p99, *throughput*) e conclusões. Este pôster acompanha este relatório como comprovante de divulgação dos resultados.

2.4. Conclusões do projeto e possibilidades futuras

O projeto alcançou o objetivo central de avaliar, em ambiente controlado e reprodutível com SGBD fixo, o impacto da linguagem/runtime e da camada de acesso a dados no desempenho de APIs de pagamento funcionalmente equivalentes. As campanhas foram conduzidas com aquecimento, janela em regime permanente e resfriamento, sob limites homogêneos de recursos (incluindo orçamento total de memória dos contêineres ≤ 350 MB). Foram obtidas séries consistentes de p95, p99, throughput e taxa de erros, permitindo comparações válidas entre as variantes implementadas (Java, JavaScript/Node, Python e Rust) com análise estatística não paramétrica e intervalos de confiança por bootstrap.

Os resultados indicam que conexões persistentes, uso de prepared statements e calibração do pool reduzem latências de cauda e variabilidade sob pressão. Verificou-se, também, que escolhas de formato de transporte e de serialização/decodificação podem deslocar significativamente p95/p99, a depender do cliente e do runtime, ainda que o SGBD e o mix de operações permaneçam constantes. Esses achados foram consolidados em protocolo reexecutável, scripts de automação e diretrizes técnicas aplicáveis a ambientes de produção.

O Gatling foi utilizado como gerador de carga principal. Os cenários reproduziram o mix de operações e as fases de execução, com perfis de injeção calibrados (rampas e patamares estáveis de

usuários virtuais, com pacing) para garantir medição em regime permanente e reprodutibilidade entre execuções. A instrumentação e a coleta foram integradas ao pipeline de experimentos, assegurando rastreabilidade dos resultados e controle de variáveis.

Alguns itens foram ajustados sem comprometer as conclusões: a variante em Go não foi incluída por priorização de maturidade e comparabilidade entre as quatro stacks restantes; a matriz completa de perturbações (latência, perda, jitter e backoff) foi reduzida para preservar isonomia entre variantes e estabilidade estatística. As contribuições permanecem alinhadas à proposta, oferecendo evidência empírica sobre caudas de latência, protocolo de medição e recomendações de engenharia.

Como continuidade, recomenda-se: (i) incluir a variante em Go e ampliar o espectro de drivers e formatos de transporte mantendo o SGBD controlado; (ii) aprofundar a análise de efeitos de GC/JIT e políticas de agendamento por runtime; (iii) estender a injeção de falhas com estratégias de backoff e controle de admissão; (iv) replicar o protocolo em arquiteturas e hardwares distintos (incluindo nuvem); e (v) padronizar suites no Gatling para comparação sistemática de perfis de carga (rampas, estágios constantes e explosões controladas), fortalecendo a generalização dos resultados.

2.5. Atividades desenvolvidas pelos pesquisadores discentes bolsistas e/ou voluntários

As atividades dos discentes abrangeram todo o ciclo experimental. No planejamento, participaram da definição do protocolo, das variáveis de controle e dos critérios de reprodutibilidade (manifestos de versão, *health check/readiness*, pinagem de CPU e cotas de memória via *cgroups*). Na preparação do ambiente, atuaram na conteinerização dos serviços, no balanceamento de instâncias da aplicação e na fixação do SGBD (mesmo motor, versão e parâmetros), garantindo comparabilidade entre as variantes (Java/GraalVM/Quarkus, JavaScript/Node/Fastify, Python/Uvicorn/FastAPI e Rust/Tokio/Axum).

Na etapa de instrumentação e execução, os discentes implementaram os *endpoints* equivalentes com idempotência por chave, configuraram os cenários do Gatling (rampas e patamares estáveis de usuários virtuais, *pacing* e janelas de aquecimento/regime permanente), e conduziram as campanhas sob limites homogêneos de recursos (orçamento total de memória dos contêineres ≤ 350 MB). Também organizaram as séries de latência (p50, p95, p99), *throughput* e erros em estruturas padronizadas, viabilizando auditoria e repetição das rodadas.

Na análise, consolidaram conjuntos de dados, efetuaram estimativas de incerteza por *bootstrap* e comparações não paramétricas entre variantes, interpretando os resultados à luz de efeitos de *pooling*, *prepared statements*, formato de transporte e estratégias de serialização/decodificação. A síntese final resultou em diretrizes técnicas e em materiais de divulgação (resumo expandido, pôster e apresentação), fortalecendo a comunicação científica.

O impacto formativo foi significativo. Os discentes desenvolveram competências em engenharia de desempenho (projeto de experimentos, controle de variáveis, coleta em regime permanente), automação e observabilidade, bem como em análise estatística aplicada. As principais dificuldades envolveram a garantia de determinismo ambiental (sincronização de relógio, variações térmicas, interferência de processos de fundo), a curva de aprendizado das *stacks* e clientes de banco de dados e o tratamento de séries assimétricas com caudas pronunciadas. A superação desses obstáculos reforçou a autonomia técnica, o rigor metodológico e a capacidade de tomada de decisão baseada em evidências, aspectos refletidos na qualidade dos artefatos entregues e na maturidade demonstrada ao longo do projeto.

2.5.1. Ciência do(s) pesquisador(es) discente(s), bolsista(s) e/ou voluntários

Documento assinado digitalmente
gov.br FABRICIO IARONKA DE SOUZA
Data: 05/02/2026 21:35:40-0300
Verifique em https://validar.iti.gov.br

Fabricio Iaronka de Souza
fabricioiaronka1@gmail.com

Documento assinado digitalmente
gov.br VINICIUS SOUZA CORBELLINI
Data: 21/01/2026 11:23:11-0300
Verifique em https://validar.iti.gov.br

Vinícius Souza Corbellini
corbellinivinicius4@gmail.com

3. RELATÓRIO FINANCEIRO

Não se aplica.

3.1 Orçamento do projeto

Sequência	Descrição do item do orçamento	Valor previsto (R$)	Valor executado (R$)	Houve alteração de despesas (Sim/Não)?
-	-	-	-	-
-	-	-	-	-

3.2 Planilha de gastos financeiros

Sequência	Número da NF	Nº do item descrito na planilha do item 3.1 deste relatório	Descrição sucinta dos itens constantes no comprovante fiscal	Valor (R$)
-	-	-	-	-
			Total Geral de AFP utilizado	-

3.3 Valor recebido de Auxílio Financeiro ao Pesquisador (AFP):
Não se aplica.

3.4 Valor de AFP não utilizado e justificativa:
Não se aplica.

4. REGISTROS DO DESENVOLVIMENTO DA PESQUISA

4.1 Registros das atividades do projeto

Os registros das APIs desenvolvidas foram versionadas em:

https://github.com/leobravoe/rust-backend-debitos-creditos-rinha-2024

https://github.com/leobravoe/node-fastify-backend-debitos-creditos-rinha-2024

https://github.com/FabricioIaronka/pythonAPI-rinha-2025

https://github.com/ViniciusCorbellini/rinha-quarkus

4.2 Relação de bens adquiridos, produzidos ou desenvolvidos

Não se aplica.

5. AUTODECLARAÇÕES
Declaro que segui todas as disposições do edital.
Declaro que não contratei, na execução do projeto, familiares, conforme prevê a Lei no 8.958/1994, em especial o § 2º do artigo 3º, referente à vedação de Contratar cônjuge, companheiro ou parente, em linha reta ou colateral, por consanguinidade ou afinidade, até o terceiro grau, de ocupante de cargo de direção das IFES envolvidas no projeto.
Declaro que não contratei, sem licitação, pessoa jurídica que tenha como proprietário sócio ou cotista, seu dirigente, servidor das IFES ou cônjuge, companheiro ou parente em linha reta ou colateral por consanguinidade ou afinidade, até o 3º grau de seu dirigente ou de servidor do IFSC.
Declaro que, nos termos do edital, solicitei o processo de patrimoniamento dos bens permanentes adquiridos com recursos do projeto.
5. ASSINATURA DO(A) COORDENADOR(A) DO PROJETO
gov.br Documento assinado digitalmente LEONARDO BRAVO ESTACIO Data: 12/02/2026 21:20:10-0300 Verifique em https://validar.iti.gov.br _____ Leonardo Bravo Estácio leonardo.bravo@ifsc.edu.br Lages, 31 de janeiro de 2026.

6. CIÊNCIA/APROVAÇÃO DA COORDENADORIA DE PESQUISA DO CÂMPUS
Observações:
gov.br Documento assinado digitalmente JAQUELINE SUAVE Data: 09/02/2026 13:06:45-0300 Verifique em https://validar.iti.gov.br _____ Jaqueline Suave pesquisa.lages@ifsc.edu.br Lages, 31 de janeiro de 2026.