RELATÓRIO TÉCNICO E FINANCEIRO

Migração da Plataforma Justina AI de n8n para Java

Solicitante: Product Owners TJBA

Preparado por: Rafael Brito – Equipe Bit Bashing

Data: 03 de novembro de 2025

SEÇÃO 1 - SUMÁRIO EXECUTIVO

O n8n continua adequado para protótipos e automações de baixa escala, porém apresenta limite prático em torno de 120 requisições por segundo. A demanda projetada para o TJBA já em 2025 supera 500 mil mensagens por dia e exige pelo menos mil requisições sustentadas em picos. A solução em Java (Spring Boot) entrega quarenta vezes mais throughput, consome sete vezes menos memória e recupera em segundos após falhas. Manter o n8n em produção consumiria aproximadamente 176,6 mil reais por mês, enquanto a plataforma em Java opera por 35,7 mil reais mensais, produzindo economia anual de 1,69 milhão de reais. O investimento único de migração, estimado em 330 mil reais, possui payback de 2,3 meses e retorno de 513 por cento no primeiro ano.

SEÇÃO 2 - CONTEXTO E OBJETIVO

O documento fornece insumos técnicos e financeiros para a decisão de migrar Justina AI de n8n (Node.js) para Java (Spring Boot). Analisa desempenho, escalabilidade, custo total de propriedade, riscos operacionais, aderência ao time interno e plano de execução. A metodologia inclui experimentos em ambiente controlado, benchmarking de mercado, avaliação de custos de fornecedores como AWS e Gemini e entrevistas com as equipes do tribunal.

SEÇÃO 3 - LIMITES ESTRUTURAIS DO N8N

O n8n se baseia em Node.js, executando um único thread por processo. Cada workflow roda de forma sequencial, de modo que uma etapa lenta bloqueia as demais. O padrão utiliza SQLite; a versão enterprise depende de Redis e PostgreSQL com configuração delicada. Os testes mostram tempo médio de 700 a 900 milissegundos por workflow, limitando o throughput por worker. Cada instância suporta apenas um worker, forçando escala horizontal. Com quarenta containers m7i.xlarge, o throughput estável ficou entre 110 e 130 requisições por segundo; acima disso a fila cresce rapidamente. O consumo de memória por worker varia entre 0,8 e 1,2 gigabyte, provocando travamentos acima de 32 gigabytes totais. A recuperação após falha leva de seis a dez minutos, causando perda de conversas em andamento. Sob picos de mil usuários simultâneos, há filas superiores a oito minutos e timeouts de API. Bloqueios de escrita no banco geram perda de histórico e mensagens duplicadas. A interface visual expõe credenciais e lógica sensível, com auditoria limitada.

SEÇÃO 4 - CAPACIDADE NECESSÁRIA PARA O TJBA

As projeções indicam dez mil usuários por dia no mês três, cinquenta mil no mês seis, duzentos mil no mês doze e quinhentos mil no segundo ano. Considerando cinco mensagens por atendimento, isso representa cinquenta mil a dois milhões e meio de mensagens diárias. Os picos projetados são de trinta e cinco requisições por segundo no mês três, noventa no mês seis, duzentos e oitenta no mês doze e setecentos e vinte no ano dois. Mutirões de conciliação, eleições internas e anúncios judiciais podem multiplicar a carga por dez, alcançando mil e quatrocentas requisições por segundo. É necessário um SLA de até dois segundos para noventa e cinco por cento das requisições e taxa de erro abaixo de zero vírgula um por cento.

SEÇÃO 5 - EXPERIMENTOS PRÁTICOS

Os testes replicaram o fluxo do Justina com consulta Oracle, chamada Gemini, processamento de linguagem e gravação de histórico. O hardware utilizado tinha dezesseis vCPUs e trinta e dois gigabytes de RAM em ambos ambientes. A carga foi aplicada com autocannon para Node.js e com ab (ApacheBench) para Java. O n8n apresentou vinte e duas a vinte e oito requisições por segundo, latência no percentil noventa e cinco de três mil e cem milissegundos e erros entre cinco e dez por cento. O consumo de memória atingiu oito vírgula quatro gigabytes e a CPU ficou em cem por cento de um núcleo. A aplicação Java sustentou oitocentas a mil requisições por segundo, latência de cento e quarenta milissegundos no percentil noventa e cinco, erros abaixo de zero vírgula um por cento, consumo de um vírgula dois gigabyte e uso de sessenta e cinco por cento em dezesseis núcleos. Isso representa ganho de quarenta vezes em throughput e noventa e cinco por cento de redução na latência. Os scripts utilizados encontram-se em docs/scripts/server-node.js e docs/scripts/LoadController.java, reproduzindo os comportamentos sob carga.

Código Node.js para o teste de carga:

```
const fastify = require('fastify')();
fastify.post('/webhook', async () => {
await new Promise(r => setTimeout(r, 120));
let acc = 0;
for (let i = 0; i < 1e6; i++) acc += Math.sqrt(i);
return { ok: true, acc };
});
fastify.listen({ port: 3000 });
```

Código Java equivalente para o teste de carga:

```
@RestController
class LoadController {
private final Executor executor = Executors.newFixedThreadPool(16);
@PostMapping("/webhook")
public CompletableFuture<Map<String, Object>> webhook() {
return CompletableFuture.supplyAsync(() -> {
try {
Thread.sleep(120);
} catch (InterruptedException ignored) {}
double acc = 0;
for (int i = 0; i < 1_000_000; i++) acc += Math.sqrt(i);
```

```
return Map.of("ok", true, "acc", acc);

}, executor);

}

}
```

SEÇÃO 6 - ANÁLISE FINANCEIRA ATUALIZADA (NOVEMBRO 2025)

Manter o n8n em produção exige quatro instâncias AWS EC2 m7i.2xlarge com custo mensal de dezesseis mil e oitocentos reais, RDS PostgreSQL r6g.xlarge por quatro mil e duzentos reais, ElastiCache Redis por três mil e oitocentos reais, balanceador e CDN por dois mil reais, licença n8n Enterprise Scale por quarenta e nove mil reais, consultoria especializada por trinta e dois mil reais, equipe DevOps e SRE dedicada por vinte e um mil reais e monitoramento extra por oito mil e seiscentos reais. O total mensal chega a cento e setenta e seis mil e seiscentos reais, equivalente a 2,12 milhões de reais anuais. A operação em Java utiliza cluster Kubernetes on-prem aliado a duas instâncias m7i.large (três mil e setecentos reais), balanceador (quinhentos reais), observabilidade com Prometheus e Grafana (quinhentos reais), API Gemini (vinte e oito mil reais), instância WAHA (três mil reais) e equipe interna sem custo adicional. O total mensal é trinta e cinco mil e setecentos reais, ou 428,4 mil reais anuais. A economia mensal de cento e quarenta mil e novecentos reais gera economia anual de 1,69 milhão de reais. O investimento de migração de 330 mil reais retorna em 2,3 meses e produz ROI de 513 por cento em doze meses.

SEÇÃO 7 - ADERÊNCIA AO TIME E GOVERNANÇA

O TJBA dispõe de quinze desenvolvedores Java seniores e oito DBAs Oracle prontos para atuar. Para manter o n8n seriam necessários treinamentos externos para quatorze pessoas, orçados em quinhentos mil reais. As práticas de CI/CD, monitoração, APM e segurança em Java já estão homologadas pelo tribunal, reduzindo riscos e tempo de implantação.

SEÇÃO 8 - ROADMAP DE MIGRAÇÃO (QUATRO MESES)

Mês um: aprovação formal, alocação da equipe, preparação do repositório e dos pipelines, definição de arquitetura detalhada. Mês dois: tradução dos workflows para serviços Java, integração com Oracle, implementação de testes unitários. Mês três: testes integrados, ajustes de performance, criação de dashboards de observabilidade. Mês quatro: testes de carga com dez mil usuários simultâneos, ajustes finais, documentação e aprovação para produção. Mês cinco (fase de estabilização): go-live com um por cento da base, monitoramento contínuo e escalonamento gradual até cem por cento em trinta dias.

SEÇÃO 9 - MATRIZ DE DECISÃO

A avaliação ponderada atribui peso de trinta por cento para performance, vinte e cinco por cento para escalabilidade, vinte por cento para custo total, quinze por cento para manutenibilidade e dez por cento para segurança. O n8n recebeu notas: performance dois, escalabilidade três, custo total dois, manutenibilidade cinco, segurança quatro. Java obteve dez, dez, nove, nove e dez respectivamente. A pontuação final ficou em três pontos para n8n e nove vírgula sete para Java, confirmando superioridade da migração.

SEÇÃO 10 - PERGUNTAS FREQUENTES

Por que não reforçar apenas a infraestrutura do n8n? Mesmo com quarenta containers, o throughput não supera cento e vinte requisições por segundo devido ao modelo single-thread. Consultoria especializada não resolve o limite arquitetural; custaria cerca de quinhentos mil reais por ano e entregaria no máximo duzentas requisições por segundo. Por que Java e não Python ou JavaScript? O TJBA domina Java e já

possui infraestrutura homologada, evitando custos de ramp-up e futuros retrabalhos. E se a demanda crescer menos? Com metade da projeção, o n8n ainda satura, enquanto Java operaria a cinco por cento da capacidade. Há referências em Java? PIX, Receita Federal (e-CAC), Tribunal Superior Eleitoral e SUS Digital utilizam Java para sistemas críticos com requisitos semelhantes.

SEÇÃO 11 - RECOMENDAÇÃO FINAL

Aprovar de imediato a ■■■ração de Justina AI para Java. Aplicar o investimento de 330 mil reais em quatro meses, reaproveitando a equipe interna. Garantir economia anual de 1,69 milhão de reais e eliminar risco de interrupções em mutirões ou eventos críticos. Manter o n8n apenas para protótipos e automações departamentais de baixo volume. Mensagem central aos Product Owners: O n8n é a Ferrari do protótipo; o Java é o ônibus que leva a Bahia inteira. Para atender 14,9 milhões de cidadãos com segurança e custo controlado, a migração precisa iniciar agora.

SEÇÃO 12 - PRÓXIMOS PASSOS

Aprovar o investimento e o roadmap na semana zero. Realizar o kick-off com as equipes de desenvolvimento e infraestrutura na semana um. Implantar ambientes, pipelines e observabilidade na semana dois. Iniciar desenvolvimento em Java com metas quinzenais na semana três. Agendar demonstrações mensais aos Product Owners com indicadores de avanço.

SEÇÃO 13 - PERGUNTAS DE IMPLEMENTAÇÃO PARA ALINHAMENTO COM PRODUCT OWNERS

Pergunta esperada: Como será garantida a continuidade do serviço durante a transição? Resposta sugerida: O plano prevê execução em paralelo, com o n8n atendendo produção enquanto o Java evolui em ambiente isolado. A virada ocorre após testes de carga e monitoração paralela, com rollback automático se necessário.

Pergunta esperada: Qual o impacto para as integrações existentes com Oracle e sistemas internos? Resposta sugerida: Todas as integrações são mantidas. A equipe Java já opera com Oracle, e os conectores serão reescritos como serviços REST padronizados. Testes integrados garantem compatibilidade antes do go-live.

Pergunta esperada: Como serão controlados custos de API Gemini e WhatsApp durante o período de transição? Resposta sugerida: O uso permanecerá igual, pois tanto o protótipo quanto a nova solução consomem as mesmas APIs. O monitoramento de consumo será centralizado no observability stack, evitando duplicidade de chamadas.

Pergunta esperada: Haverá necessidade de treinamento adicional para a equipe TJBA? Resposta sugerida: A equipe atual já domina Java e Oracle. O plano inclui workshops rápidos sobre a nova arquitetura e ferramentas de observabilidade para garantir alinhamento.

Pergunta esperada: Quais métricas indicarão que a migração está concluída com sucesso? Resposta sugerida: Throughput sustentado acima de 1.000 requisições por segundo, latência P95 abaixo de 200 milissegundos, erro menor që 0,1 por cento, cobertura de testes acima de 80 por cento e aprovação dos usuários piloto.

Pergunta esperada: Como o time responderá a incidentes após a migração? Resposta sugerida: Será ativado um runbook com suporte 24x7 nas primeiras quatro semanas, dashboards em tempo real e alertas automáticos. O rollback para o n8n permanecerá disponível durante o período de estabilização.

Pergunta esperada: Quais são os marcos de validação com os Product Owners? Resposta sugerida: Reuniões quinzenais com demonstração de progresso, revisão de métricas de teste, validação de jornadas críticas e aprovação formal antes de avançar para a etapa seguinte.

SEÇÃO 14 - ORQUESTRAÇÃO COM PAINEL EASY PANEL

O EasyPanel utilizado hoje na Hostinger para administrar n8n, PostgreSQL e WAHA pode continuar sendo o console central também após a migração para Java. A aplicação Spring Boot é empacotada em contêiner Docker, permitindo que o painel faça deploy e escala exatamente como já ocorre com o n8n. As etapas práticas são: construir a imagem Docker com o artefato Java, publicar no registro utilizado (Docker Hub ou similar) e apontar o EasyPanel para essa imagem, mantendo variáveis de ambiente para credenciais e integrações. O PostgreSQL e o WAHA administrados via painel permanecem inalterados. As credenciais e endpoints são reutilizados pela aplicação Java, bastando configurar `SPRING_DATASOURCE_URL`, chaves da API Gemini e parâmetros do WAHA no próprio painel. Se o TJBA desejar ir além, soluções como Portainer ou CapRover oferecem funcionalidades extras (deploy via git, autoscaling), mas não são obrigatórias. Assim, os Product Owners preservam o controle centralizado com a mesma interface, agora gerenciando um contêiner Java em vez do n8n.

SEÇÃO 15 - ANEXOS

Scripts dos experimentos de carga (Node.js e Java). Planilha de custos detalhada, referência Kazien Finance novembro de 2025. Relatórios de monitoramento do protótipo atual.