

Visão Geral:

A empresa **não pratica DevOps** hoje. O que existe é uma separação clássica Dev x Ops, com handoff manual, deploys frágeis e testes tardios em produção.

Os dados comprovam isso:

- 2 dias entre código pronto e deploy → gargalo organizacional, não técnico
- 80% de sucesso em deploy manual → inaceitável para produção
- 2 incidentes por semana → efeito direto de falta de automação e testes
- MTTR de 4h → ausência de observabilidade e resposta estruturada

1. C – CALMS

Processo escolhido: Entrega de código até deploy em produção

Processo atual

1. Os Desenvolvedores “terminam” o código, não há uma definição clara de pronto.
2. Geram um pacote manualmente.
3. Envia para Operações.
4. Operações faz deploy manual.
5. Testes manuais **em produção**.
6. Monitoramento manual de logs.

Pontos críticos

- **Transferência de responsabilidade:** “agora é problema do Ops”
- **Ops testando código que não escreveu**
- **Testes tardios** → bugs descobertos quando o impacto já é alto
- **Conhecimento concentrado:** Delphi legado depende de 1 pessoa.

Oportunidades de melhoria

- Eliminar handoffs manuais
- Trazer testes para antes do deploy
- Padronizar deploy
- Compartilhar responsabilidade por estabilidade (Dev + Ops)

Risco ignorado pela empresa: o sistema legado em Delphi é um *single point of failure humano*.

2. A – CALMS

Solução proposta: Pipeline CI/CD padronizado

O que automatizar

1. **Build automático**
2. **Testes automatizados**
3. **Deploy automatizado**
4. **Rollback automático**
5. **Monitoramento e alertas**

Arquitetura proposta

- **CI/CD:** GitHub Actions, GitLab CI ou Azure DevOps
- **Builds:**
 - Java: Maven/Gradle
 - C#: .net CLI
 - JavaScript: npm/yarn
- **Testes:**
 - Unitários obrigatórios
 - Integração para e-commerce
- **Deploy:**
 - Blue-Green ou Canary
- **Infra:**
 - Docker
 - Infra como Código (Terraform ou equivalente)

Plano de implementação

Erro comum: tentar implantar DevOps “por decreto”.

Fases:

Fase 1 – Piloto (E-commerce)

- Sistema novo, menos dependências
- Pipeline simples
- Deploy automatizado em staging

Fase 2 – Produção controlada

- Deploy automatizado com aprovação manual
- Métricas visíveis

Fase 3 – Legado

- Pipeline mínimo
- Testes básicos
- Documentação do sistema Delphi

Risco: tentar automatizar o legado sem antes documentá-lo vai gerar caos. Primeiro conhecimento, depois automação.

3. M e S – CALMS

Métricas relevantes

Usaremos **DORA Metrics**, porque são comprovadas:

1. **Lead Time** (commit → produção)
2. **Deployment Frequency**
3. **Change Failure Rate**
4. **MTTR**

Meta (6 meses)

- Lead Time: 2 dias → **30 minutos**
- Deploy success: 80% → **>95%**
- Incidentes: 2/semana → **<1/mês**
- MTTR: 4h → **<30 min**

Compartilhamento de conhecimento

Ações práticas

- Post-mortem sem “caça às bruxas”
- Tech talks internas mensais
- Documentação viva (README, runbooks)
- Pair Dev-Ops em incidentes

4. As Três Maneiras do DevOps

Primeira Maneira – Acelerar o Fluxo

Problemas atuais:

- Deploy manual
- Testes tardios

- Dependência de Ops

Ações:

- Pipeline automatizado
- “You build it, you run it” (parcial)
- Ambientes padronizados

Resultado esperado: fluxo contínuo, menos interrupções.

Segunda Maneira – Ampliar o Feedback

Problemas atuais:

- Feedback só após erro em produção
- Logs manuais

Ações:

- Monitoramento com métricas e alertas
- Logs centralizados
- Feedback automático no PR (falha de teste bloqueia merge)

Resultado: erro pequeno detectado cedo.

Terceira Maneira – Experimentar e Aprender

Problemas atuais:

- Medo de deploy
- Falha vista como culpa

Ações:

- Deploys pequenos e frequentes
- Feature flags
- Ambiente de experimentação
- Blameless post-mortems