Simples soluções, grandes resultados: melhorando legados

Trilha Dados e Estratégia

Márcio Rogério Nizzola

Márcio R. Nizzola

Tech Lead na CI&T, com foco em .NET.

Formado em Análise de Sistemas e MBA em Gestão de Projetos.

Professor na Etec de Itu desde 2008, atuando nos cursos técnicos da área de tecnologia, ministrando disciplinas de programação, bancos de dados e projetos de TCC.

Desenvolvendo software desde 1992.

Membro fundador da comunidade Itu Developers.

Esta palestra é um compilado de casos comuns, boas práticas e qualquer semelhança com o ambiente, pessoas ou casos é mera coincidência!

Nizzola

O QUE É UM LEGADO ?

Todo software assim que é colocado em produção torna-se um legado!

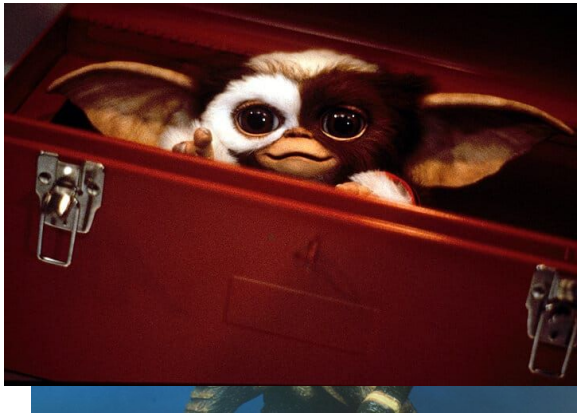
Ele traz consigo a herança arquitetural de sua época e equipe que o desenvolveu.

Mudar qualquer coisa é difícil e impacta muitas vezes em limitações técnicas.

O custo destas manutenções são sempre Impactados pelas características arquiteturais e técnicas da aplicação.

GO-LIVE DO SOFTWARE

DURANTE O CICLO DE VIDA



Problemas comuns em Legados

Segurança

Sistemas antigos são mais vulneráveis

Performance

Limitações técnicas do legado impedem o uso de cloud e escalabilidade

Custos

A estrutura legada na maioria das vezes tem custo maior.

Pessoas

Manutenção do time e novas contratações mais difíceis.

Motivação

Manter a equipe motivada diante das limitações é sempre um desafio.

Evolução

Nem tudo que se quer desenvolver é suportado

Espera-se que o mercado global de serviços de modernização de aplicativos cresça de US $ 11,4 bilhões em 2020 para US $ 24,8 bilhões em 2025 a uma taxa composta de crescimento anual (CAGR) de 16,8% durante o período de previsão.

"Application Modernization Services Market", MarketsandMarkets, julho de 2020

COMO SE FAZIA SOFTWARE NO PASSADO ?
POR QUE DEVEMOS MUDAR !

1

Aplicações Monolíticas

Tínhamos aplicações monolíticas, sem separação entre front-end e back-end, com regras espalhadas no código.

2

Bancos de Dados pesados

Era muito comum ter apenas conceito de bancos de dados SQL based, obrigando a ter altos custos de licenciamento e suporte à bancos gigantescos.

3

Ausência de testes

Código com alto acoplamento, regras de negócio não testáveis e possibilidade de quebrar a aplicação toda a cada nova implementação.

4

Falta de monitoramento

A falta de monitoramento das aplicações impede a ação rápida das equipes de suporte, problemas somente são detectados quando alguém reclama.

5

Infra-estrutura local

A maioria dos legados foram feitos baseados na idéia de ter um servidor local onde nele era implantada a solução, dificultando a escalabilidade e redundância.

6

Sem suporte Mobile

Aplicações desenvolvidas há mais de 10 anos sequer pensavam na concentração maior de utilização de aplicações mobile.

CI&T

DIFÍCIL DECISÃO:
CRIAR NOVAS APLICAÇÕES OU MELHORÁ-LAS?

Pontos negativos para criar tudo novo

- Alto custo
- Tempo de desenvolvimento
- Complexidade da tarefa
- Foco do time irá atrapalhar resolução de problemas urgentes

Alternativa

- Resolver problemas perceptíveis na aplicação atual, dando uma sobrevida ao legado
- Pagar as maiores dívidas técnicas e assim facilitar a correção de bugs

Impactos

- Redução de cobranças na resolução de bugs.
- Percepção dos usuários em melhorias de performance, principalmente com aumento da demanda
- Diminuição de abertura de chamados para deixar equipe livre para criar novas aplicações

Não tem como melhorar o que já temos para suportar o tempo de desenvolvimento de novas versões das aplicações? Evitando perder tempo com bugs?

Perguntas comuns feitas pela gerência de TI

Quais são os maiores problemas da aplicação atual que causarão stress enquanto se faz as novas ?

Pergunta feita pela equipe de desenvolvimento

Por que demora tanto para resolver um bug na versão atual?

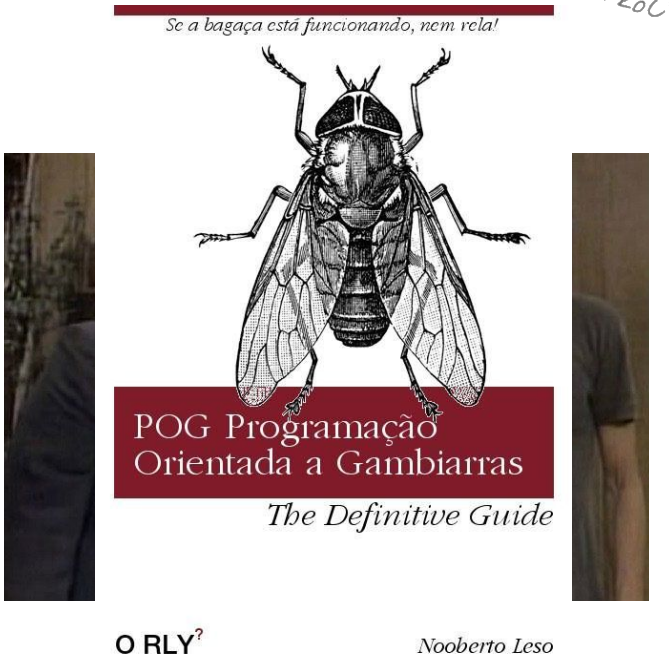
Perguntas comuns feitas pela gerência de TI

DÍVIDAS TÉCNICAS

- Pagar as dívidas que tem o maior custo (juros mais altos). Exemplo, aquelas que para desenvolver novas features, obrigam a ter novos contornos, consumindo sempre mais tempo de desenvolvimento e aumentando a dívida

- Corrigir aquelas em que por meio de métricas são as causadoras da maioria dos problemas reportados

- Aquelas que impedem a evolução técnica de alguma forma (conteinerização, escalabilidade, uso de bancos não relacionais, obrigatoriedade de uso de bibliotecas obsoletas)



POR QUE É TÃO DIFICIL RESOLVER DÍVIDAS TÉCNICAS ?

Elas não são visíveis pelos usuários.
Suas correções na maioria das vezes não impactam em mudanças perceptíveis.
Quem paga a conta não vê valor nisso (apesar de estar gastando bem mais em cada intervenção no código).
Sempre há outra prioridade.

Como convencer a priorizar ?

- métricas de chamados abertos.
- métricas de entregas de features.
- métricas de qualidade do código.

CI&T

“

Para cada bug corrigido, dois novos problemas são introduzidos.

”

CI&T

POR ONDE COMEÇAR ?

PROPOSTA DE PLANO DE AÇÃO

01 › **02** › 03

Propor nova arquitetura

Melhorar Legado

Desenvolver novas aplicações

SABEMOS ONDE QUEREMOS CHEGAR

Quebra de monolito em pequenos serviços

Reescrita de código em pequenos serviços e aos poucos ir segregando as aplicações em pequenos serviços. (Strangle Pattern)

TEMPO ESTIMADO: MAIS DE UM ANO COM A EQUIPE ATUAL

Divisão da aplicação em domínios distintos

A equipe de desenvolvimento pode focar em cada domínio do sistema separadamente e assim entregar as melhorias em partes, sem precisar ter conhecimento do todo..

Evolução tecnológica

Utilizar novos conceitos como conteinerização, bancos não relacionais, serverless, frameworks atualizados, padrões de projeto e TDD.

CI&T

Perguntas comuns feitas pela gerência de TI

SITUAÇÃO DA EQUIPE DE DESENVOLVIMENTO

“

Que tal melhorar a aplicação atual para suportar o tempo de desenvolvimento da nova e assim trabalharmos tranquilamente ?

”

Proposta da equipe de desenvolvimento.

AÇÕES INICIAIS: INFRAESTRUTURA

META: GARANTIR A ESTABILIDADE DA APLICAÇÃO

Monitorar aplicações

Com a introdução do monitoramento passamos a saber na hora quando falhas graves ocorrem ou perdas de performance.

Implantar CI

Poder realizar deploys com segurança e rapidez também dá maior confiança na liberação de features e hotfixes.

Gerenciar Jobs

Controlar de forma centralizada jobs em execução e monitorar as suas falhas. Podendo ativar e desativá-los quando necessário.

Plano de SCM

Uma política clara e estabelecida com o time garante que não haja perdas inconsistências no código.

“

Com estas mudanças pudemos conferir maior confiabilidade à manutenção do legado e termos métricas para orientar pontos de melhoria.

”

Case 1 - Performance em auto complete

Solução: implementação de cache

Situação problema: Usuários reclamando da lentidão no autocomplete da solução.

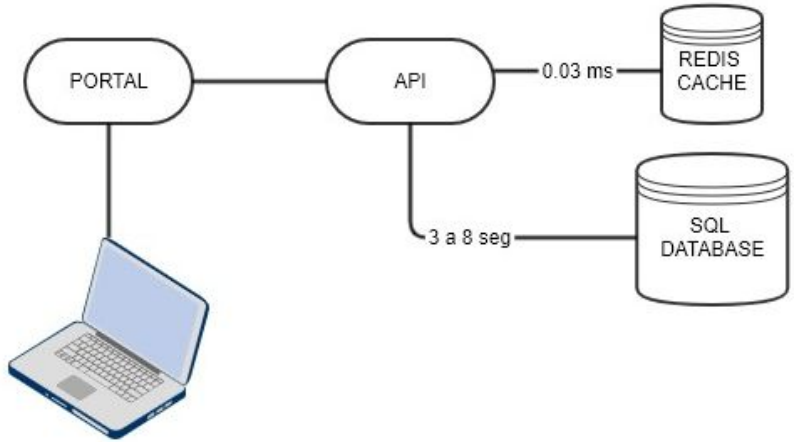
Diagnóstico: para formação do conteúdo, há acesso à um banco de dados SQL com centenas de milhares de localidades, onde consultas podem levar de 3 a 8 segundos dependendo da combinação textual.

Case 1 - Performance em auto complete

Solução: implementação de cache

Métricas: A mesma consulta é disparada inúmeras vezes no dia com as mesmas palavras chave.

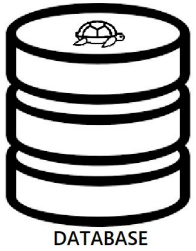
Proposta de melhoria: Através do uso de cache, salvar resultados obtidos para as próximas consultas, deixando a lentidão apenas para o primeiro usuário a solicitar a mesma palavra chave.



Resultados: percepção imediata dos usuários
Tempo utilizado na solução: 1 dia

Case 2 - Consultas SQL com Serialização XML

Solução: Reescrita das consultas



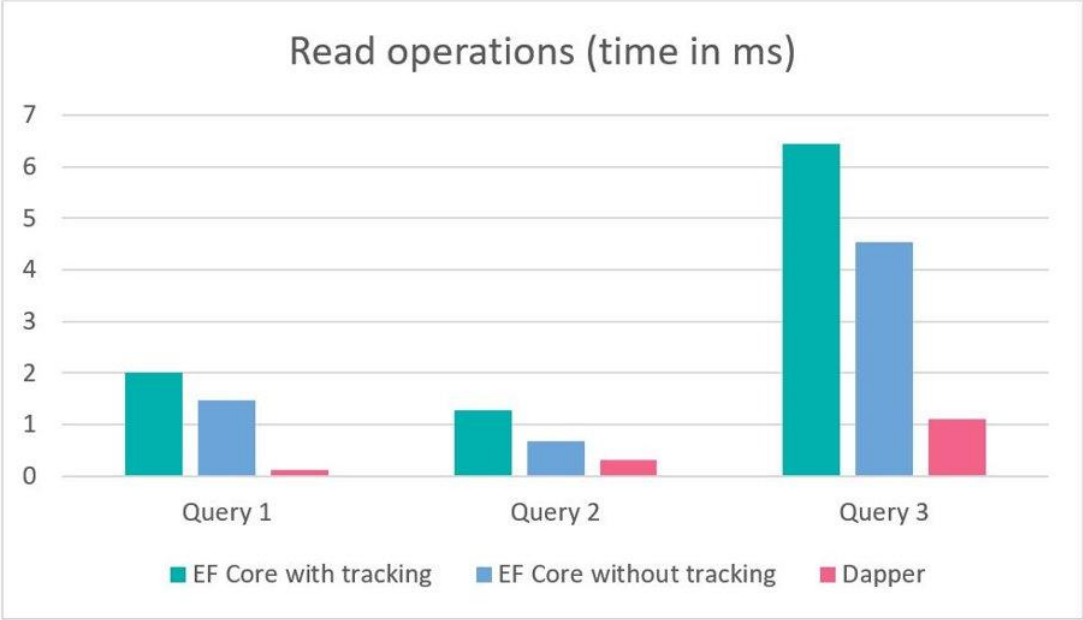
Situação problema: Lentidão nas consultas ao banco de dado

Diagnóstico: Identificamos a transformação dos resultados em XML dentro das consultas e a sua deserialização dentro da aplicação, causando:

- Maior lentidão na execução das consultas no banco de dados.
- Maior consumo de banda na transferência de dados entre servidor de banco e a aplicação.
- Consumo de memória e processamento na aplicação ao alocar espaço para os dados em XML e depois na criação de coleções de objetos com os dados deserializados.
- Procedures mal escritas, com múltiplos JOINS, algumas com 3000 linhas convertidas para apenas 700 linhas.

Case 2 - Consultas SQL com Serialização XML

Solução: Reescrita de consultas SQL com implementação do Dapper no C#.



Resultados:

- Percepção de melhora no desempenho imediata.
- Redução do uso de infra estrutura de banco de dados (menos custos).
- Redução no consumo de memória da aplicação.
- Redução de consumo de banda na comunicação entre servidor e banco de dados.

Case 3 - Lentidão nas consultas

Solução: expurgo de dados antigos

Situação problema: Lentidão nas consultas ao banco de dados.

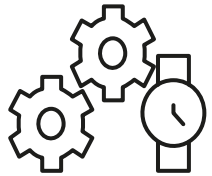
Diagnóstico: Identificamos dados legados, com informações de mais de 10 anos, armazenadas sem a menor necessidade.

Solução: Arquivamento de informações legadas.

Resultados: Além do aumento da performance, o consumo de memória do servidor de banco que antes era um servidor de 256 Gb, hoje opera com apenas 32 Gb.

Case 4 - Lentidão em horários pré-definidos

Solução: ajuste de jobs e aplicações



Situação problema: Existência de inúmeros jobs (Windows Service e Console Application) espalhados pelos servidores. Suspeita de lentidão em determinados horários ser causada por eles.

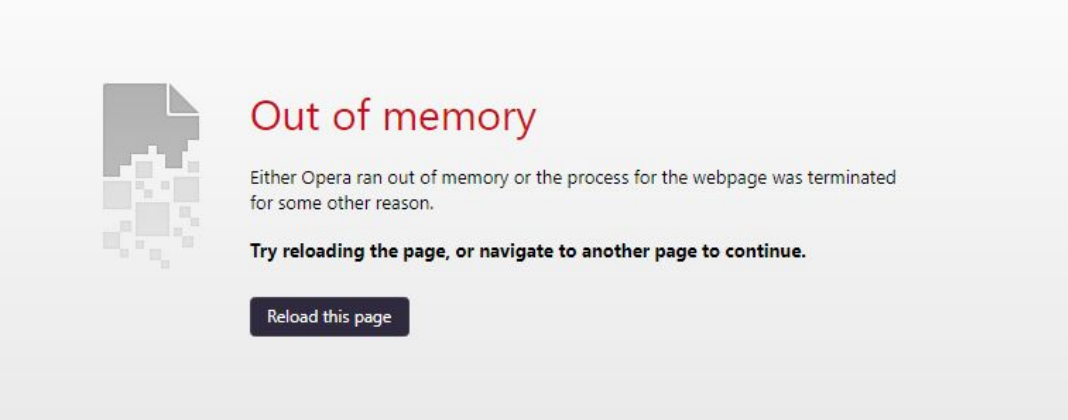
Diagnóstico: Havia jobs em execução como Windows Services, e uma outra parte sendo orquestrada e executada via Jenkins.

Solução: Conversão de todos os Jobs para Console Application e inserção na orquestração do Jenkins para melhor gerenciamento.

Resultados: Controle sobre horários de execução dos jobs e centralização de administração em interface amigável e controlada. Melhoria nos momentos em que o site tornava-se indisponível devido à concorrência no acesso à dados.

Case 5 - Quebras em aplicações

Situação problema: Em algumas situações o sistema exibia no front-end mensagem de falhas na aplicação.

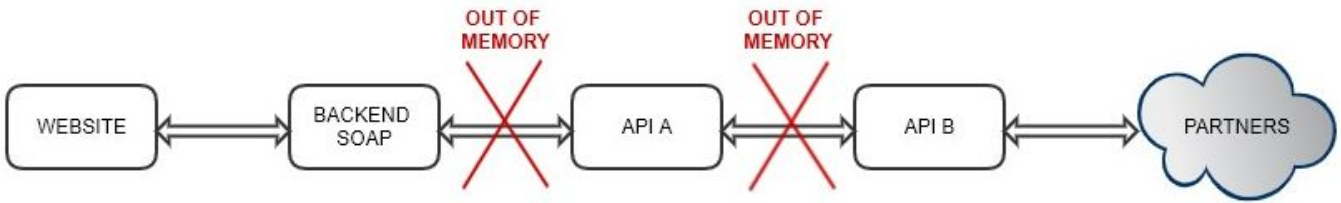


Observação: O uso da ferramenta de monitoramento foi essencial nessa descoberta, dado que os logs não eram 100% funcionais.

Case 5 - Quebras em aplicações

Solução 1: remoção de compactação entre APIS

Diagnóstico: Foi necessário revisar todo o fluxo da operação em que ocorria o problema para identificar o ponto de quebra.
Em dadas circunstâncias, a quantidade de informações trocada entre API´S era tão grande, que a descompactação dos pacotes trocados ocasionava quebra da aplicação por excessivo consumo de memória.



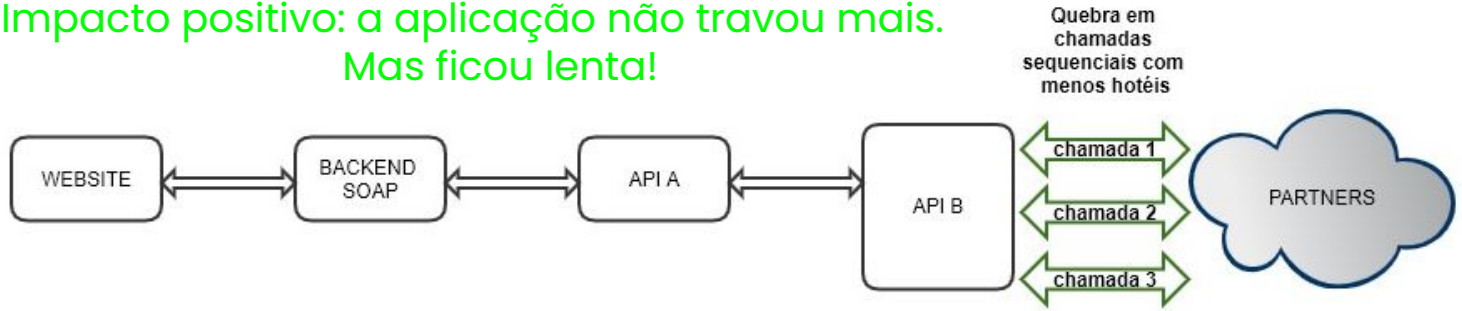
Solução parte 1: remover a compactação, dado que as aplicações estavam no mesmo datacenter e o tempo de transferência não tinha grande impacto.

Case 5 - Quebras em aplicações

Solução 2: quebra em menores chamadas

Diagnóstico: Ainda assim, alguns casos continuaram a acontecer quando eram acionadas consultas de hotéis em cidades maiores.
O tamanho do objeto Json que continha múltiplos hotéis chegava em alguns casos à 2 milhões de caracteres, estourando a capacidade de deserialização da biblioteca utilizada.
Solução: quebrar a chamada limitando o número de hotéis e assim receber objetos menores e evitar o limite de tamanho.

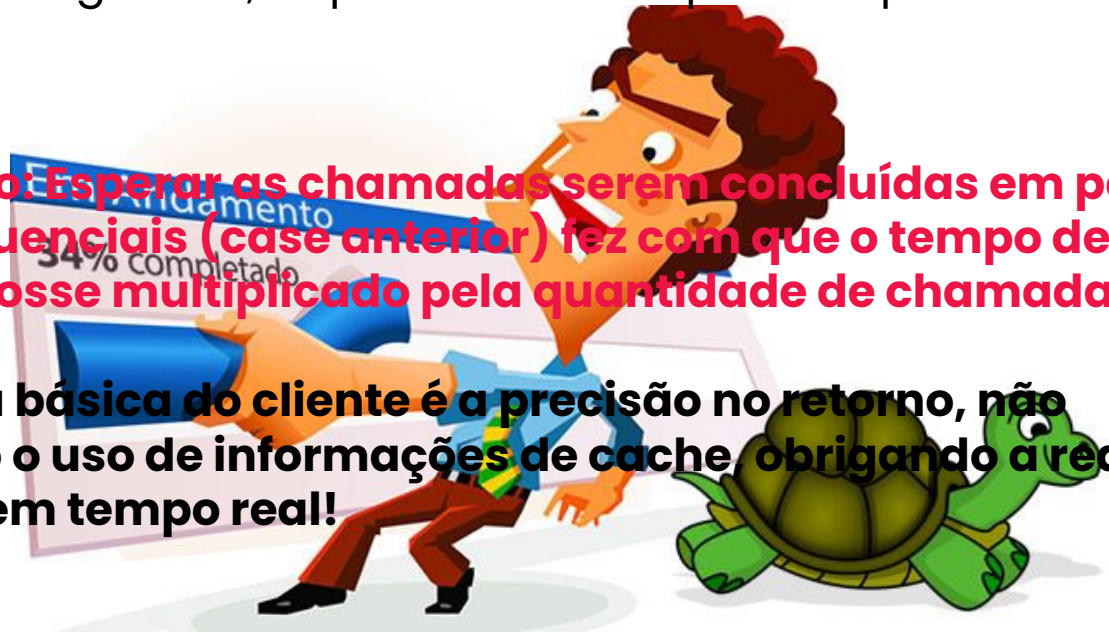
Impacto negativo: a aplicação funciona mas o desempenho é péssimo.
Impacto positivo: a aplicação não travou mais.
Mas ficou lenta!



Case 6 - Lentidão em respostas

Solução: revisão de consultas à parceiros

Situação problema: Agora que todas as buscas funcionavam, algumas eram realizadas com tempo de resposta em torno de 1 minuto e 40 segundos, o que é muito tempo de espera do usuário.



Diagnóstico: Esperar as chamadas serem concluídas em pequenos blocos sequenciais (case anterior) fez com que o tempo de execução fosse multiplicado pela quantidade de chamadas.

A premissa básica do cliente é a precisão no retorno, não permitindo o uso de informações de cache, obrigando a realizar consultas em tempo real!

Case 06 - Lentidão em respostas

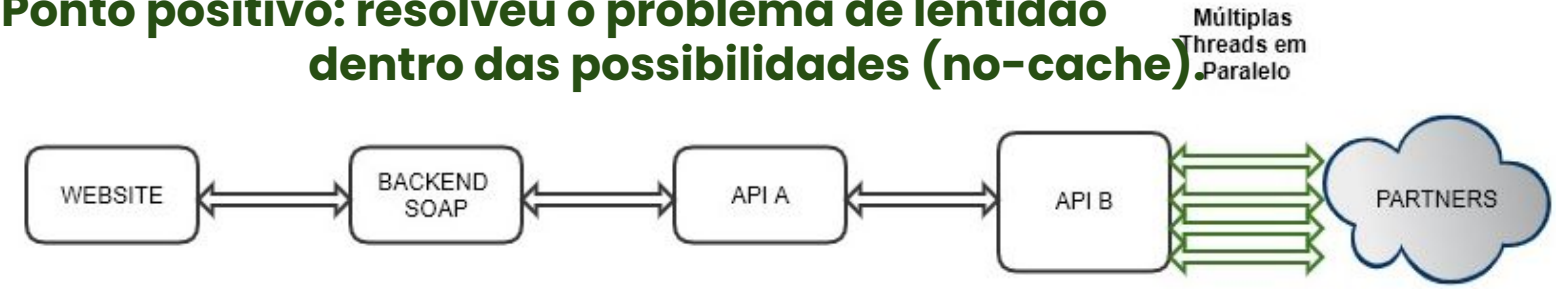
Solução: revisão de consultas à parceiros

Solução: alteração da API com a **implementação de chamadas assíncronas paralelas**

Resultados: redução do tempo de execução !
Chamadas de 1:40 min reduzidas para 30 a 40 segundos.

Ponto negativo: alto consumo de processamento.

Ponto positivo: resolveu o problema de lentidão dentro das possibilidades (no-cache).



Case 7 - Lentidão no carregamento Web

Solução: eliminação de tratamentos com javascript

Problema: Após o backend retornar dados para a montagem da tela, o navegador ainda ficava realizando processamentos de scripts em background, não liberando o usuário para interagir com o site até a finalização.

Diagnóstico: Foram encontrados scripts rodando após o postback para modificar elementos do site, alterando valores, aplicando regras de negócio sobre alguns elementos, e até fazendo ofuscação de objetos para que não fosse utilizado recurso de copiar-colar.

Solução: Remoção de regras de negócio presentes em código javascript, e introdução do tratamento destas regras no backend, fazendo com que os dados já sejam introduzidos no retorno tratados.
Resultado: dependendo da tela 10 a 20 segundos de redução.

CI&T

Case 8 - Escalabilidade

Solução: preparar a aplicação

Situação problema: Aplicações legadas possuem vários empecilhos para que se haja mais de um servidor atendendo as requisições.

Diagnóstico: Encontramos os seguintes itens impeditivos:

- Sessões em memória na aplicação.
- Cache presente na aplicação.
- Arquivos de imagens dentro do próprio server (\images \content)

Case 9 - Escalabilidade

Solução: preparar a aplicação

Solução: Criar meios para que as aplicações possam ter um ponto único para manter estas informações possibilitando então a colocação de um load balancer.

- Implementando sessões em um Cache Redis
- Utilizando um storage com mapeamento dentro de cada server para que as pastas \images e \content referenciem sempre os mesmos locais.
- Remoção de cache da aplicação de dados mutáveis para um cache único no Redis.

Resultados: Garantir a execução em mais de um servidor até que se possa implementar serviços escaláveis e resilientes em clusters kubernetes.

CI&T

Case 9 - Regras de negócio complexas

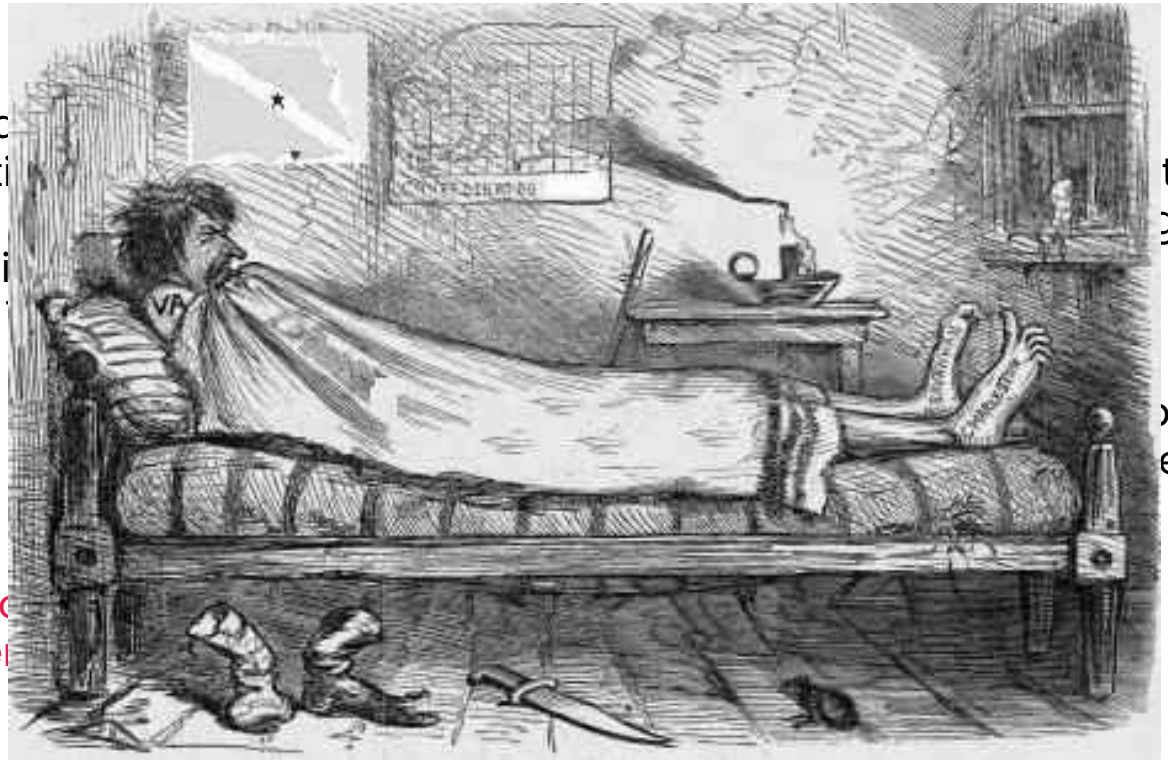
Problema: falta de testes de unidade.

Problema
Diagnósti
portanto
que não i
Solução:

tável,
garantir

ráticas
empre

Resultado
implemer



CI&T

E ASSIM PODEMOS DEIXAR O CAMINHO LIVRE PARA EVOLUIR

DÚVIDAS ?

CONTATO

marcio@nizzola.com.br
www.linkedin.com/in/nizzola

linktr.ee/NIZZOLA

E AMANHÃ TEM MAIS...

ciandt.com

THANK YOU