Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Instituto de Gestão e Tecnologia da Informação**

**Relatório do Projeto Aplicado**

Proposta de arquitetura para melhoria na criação e processamento de relatórios

Lauan Borges Guermandi

Orientador(a): Ricardo Brito Alves

09/03/2022

****

**LAUAN BORGES GUERMANDI**

**INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO**RELATÓRIO DO PROJETO APLICADO

PROPOSTA DE ARQUITETURA PARA MELHORIA NA CRIAÇÃO E PROCESSAMENTO DE RELATÓRIOS

Relatório de Projeto Aplicado desenvolvido para fins de conclusão do curso Arquitetura de software e soluções.  
  
Orientador (a): Ricardo Brito Alves

**Pederneiras - SP  
09/03/2022**

**Sumário**

[1. CANVAS do Projeto Aplicado 5](#_heading=h.2xcytpi)

[1.1 Desafio 6](#_heading=h.3as4poj)

[1.1.1 Análise de Contexto 6](#_heading=h.1pxezwc)

[1.1.2 Personas 7](#_heading=h.2r0uhxc)

[1.1.3 Benefícios e Justificativas 8](#_heading=h.1664s55)

[1.1.4 Hipóteses 9](#_heading=h.3q5sasy)

[1.2 Solução 10](#_heading=h.25b2l0r)

[1.2.1 Objetivo SMART 10](#_heading=h.kgcv8k)

[1.2.2 Premissas e Restrições 11](#_heading=h.34g0dwd)

[1.2.3 Backlog de Produto 12](#_heading=h.1jlao46)

[2. Área de Experimentação 13](#_heading=h.43ky6rz)

[2.1 Sprint 1 15](#_heading=h.2iq8gzs)

[2.1.1 Solução 15](#_heading=h.xvir7l)

[● Evidência do planejamento: 15](#_heading=h.3hv69ve)

[● Evidência da execução de cada requisito: 15](#_heading=h.1x0gk37)

[●](#_heading=h.4h042r0) Evidência dos resultados: 15

[2.1.2 Lições aprendidas 15](#_heading=h.2w5ecyt)

[2.2 Sprint 2 16](#_heading=h.1baon6m)

[2.2.1 Solução 16](#_heading=h.3vac5uf)

[● Evidência do planejamento: 16](#_heading=h.2afmg28)

[● Evidência da execução de cada requisito: 16](#_heading=h.pkwqa1)

[●](#_heading=h.39kk8xu) Evidência dos resultados: 16

[2.2.2 Lições aprendidas 16](#_heading=h.1opuj5n)

[2.3 Sprint 3 17](#_heading=h.48pi1tg)

[2.3.1 Solução 17](#_heading=h.2nusc19)

[● Evidência do planejamento: 17](#_heading=h.1302m92)

[● Evidência da execução de cada requisito: 17](#_heading=h.3mzq4wv)

[●](#_heading=h.iq2k9uh7fyi7) Evidência dos resultados: 17

[2.3.2 Lições aprendidas 17](#_heading=h.2250f4o)

[3. Considerações Finais 18](#_heading=h.haapch)

[3.1 Resultados Finais 18](#_heading=h.319y80a)

[3.2 Contribuições 18](#_heading=h.1gf8i83)

[3.3 Próximos passos 18](#_heading=h.40ew0vw)

## 1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado.

Interface gráfica do usuário

Descrição gerada automaticamente

## Desafio

### 1.1.1 Análise de Contexto

A criação de relatórios para a análise estratégica que facilita a tomada de decisão na hora de avaliar o rumo que um negócio irá tomar, é um pré-requisito para o sucesso.

O desafio deste projeto é garantir que grandes quantidades de dados sejam extraídas de um sistema, sem que ele seja impactado negativamente.

Temos uma cenários onde existe um sistema com uma arquitetura distribuída, orientada a serviços, que utiliza vários bancos de dados relacionais SQL Server, similar ao seguinte diagrama:

Diagrama

Descrição gerada automaticamente

O sistema é utilizado para que clientes recebam as dívidas de seus próprios clientes. O nosso cliente cadastra um cliente dele, e a dívida que este cliente tem com ele, então o cliente dele utiliza nosso sistema para realizar o pagamento desta dívida.

Neste sistema há uma funcionalidade de gerar relatórios, onde grandes quantidades dados são extraídos diretamente dos bancos de dados do ambiente de produção, e muitas vezes falham.

Esta proposta de arquitetura tem como objetivo, diminuir os impactos negativos do processamento de relatórios, na aplicação.

Visando uma melhor compreensão do cenário e o problema apresentado para este projeto aplicado, iremos a seguir apresentar a matriz CSD.

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Com a utilização dessa ferramenta, podemos verificar que além dos relatórios falharem, também geram um impacto negativo na aplicação, gerando lentidão no sistema, e erros em outras funcionalidades, devido a ***locks*** nos bancos de dados.

Minha proposta para melhorar este processo é replicar o banco de dados e atualizá-lo em tempo real, utilizando uma estratégia de CDC(Change Data Capture). Assim o banco de dados de produção e a aplicação em si, não serão impactados pelo processamento de um relatório.

Para sintetizar as observações feitas, vamos utilizar o framework POEMS.

Interface gráfica do usuário

Descrição gerada automaticamente

### 1.1.2 Personas

As personas envolvidas são os clientes, que usam o sistema para cadastrar dívidas, e seus clientes que utilizam o sistema para pagar suas dívidas.

Para melhor entendimento, vamos utilizar a ferramenta Mapa de empatia.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MAPA DE EMPATIA: CLIENTE** | | | | | |
| **Quem** | **Fazer** | **Vê** | **Diz** | **Faz** | **Ouve** |
| Cliente | Obter relatórios para otimizar sua estratégia de venda. | Oportunidade de receber dívidas de seus próprios clientes. | Eu preciso de relatórios para melhorar o processo de tomada de decisão. | Busca inovação. | É difícil tomar uma decisão sem dados para avaliar. |
| PENSA/SENTE | | | | | |
| DORES | | | GANHOS | | |
| Problemas ao gerar relatórios do sistema. | | | Maior rendimento. | | |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MAPA DE EMPATIA: CLIENTE DE CLIENTE** | | | | | |
| **Quem** | **Fazer** | **Vê** | **Diz** | **Faz** | **Ouve** |
| Cliente dos Clientes | Busca comodidade. | Facilidade no uso de tecnologia. | Quero facilidade nos processos. | Busca conforto. | Tecnologia é bem-vinda. |
| PENSA/SENTE | | | | | |
| DORES | | | GANHOS | | |
| Problemas ao pagar dívidas. | | | Facilidade no pagamento das suas dívidas. | | |
|  |

### 1.1.3 Benefícios e Justificativas

Quando um relatório está sendo processado, o sistema fica instável e lento devido à alta quantidade de dados, e o tempo em que demoram para ser extraídos, pois os dados são consultados diretamente no base de dados de produção. Isso interfere em outras várias operações no sistema, como também no fluxo de pagamento, diminuindo o rendimento do sistema.

A intenção deste projeto é criar uma arquitetura capaz de suportar a extração de grandes massas de dados, sem que seja afetada a aplicação, assim evitando lentidão e erros internos.

Com a solução aplicada, a utilização da aplicação trará uma melhor experiência aos usuários, e maior credibilidade ao sistema.

Para melhor entendimento, vamos utilizar o CANVAS de proposta de valor.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente com confiança média

Para encontrar pontos de melhorias e oportunidades de inovação para a realização desse projeto, vamos utilizar a ferramenta Blueprint.

|  |  |
| --- | --- |
| BLUEPRINT | |
| OBJETIVOS | Obter um relatório |
| ATIVIDADES | Acessar o sistema |
| QUESTÕES | Ação simples |
| BARREIRAS | Erro ao gerar relatório |
| AÇÕES DE CLIENTE | Quer obter um relatório |
| FUNCIONALIDADE | Maior garantia de sucesso de execução |
| INTERAÇÃO | Utiliza um navegador |
| MENSAGEM | Um arquivo de relatório |
| ONDE OCORRE | Navegador |
| TAREFAS APARENTES | Solucionar erro ao gerar relatório |
| TAREFAS ESCONDIDAS | Reduzir sobrecarga no banco de dados |
| PROCESSOS DE SUPORTE | Comunicar ao suporte técnico |
| SAÍDA DESEJÁVEL | Gerar um relatório com sucesso |

### 1.1.4 Hipóteses

Com relação as hipóteses levantadas para este projeto aplicado observamos que:

|  |  |
| --- | --- |
| OBSERVAÇÃO | HIPÓTESE |
| O cliente espera poder tirar relatórios em qualquer horário no sistema. | Especula que o sistema pode gerar relatórios, sem ocorrer problemas. |

Após um brainstorm, foram levantadas algumas ideias:

|  |  |
| --- | --- |
| CENÁRIOS | |
| C1 | COMPLEXIDADE NA EXECUÇÃO DO PROJETO |
| C2 | URGÊNCIA NA EXECUÇÃO DO PROJETO |
| C3 | INVESTIMENTO NECESSÁRIO A EXECUÇÃO DO PROJETO |
| C4 | BENEFÍCIOS ESPERADOS AO FINAL NO PROJETO |
| C5 | NIVEL DE SATISFAÇÃO DA ALTA ADMINISTRAÇÃO |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ESCALA | BENEFÍCIOS | ABRANGÊNCIA | SATISFAÇÃO | INVESTIMENTO | CLIENTES | OPERACIONAL |
| 5 | VALOR IMEDIATO PARA O MODELO DE NEGÓCIO | TOTAL | TOTAL | NENHUM | NENHUM IMPACTO | DIFICULDADE: MUITO FÁCIL |
| 4 | SIGNIFICATIVO PARA O MODELO DE NEGÓCIO | GRANDE | GRANDE | BAIXO | BAIXO IMPACTO | DIFICULDADE: FÁCIL |
| 3 | RAZÁVEL PARA O MODEO DE NEGÓCIO | RAZOÁVEL | RAZOÁVEL | MÉDIO | MÉDIO IMPACTO | DIFICULDADE: MÉDIO |
| 2 | POUCO PARA O MODELO DE NEGÓCIO | PEQUENA | PEQUENA | GRANDE | GRANDE IMPACTO | DIFICULDADE: GRANDE |
| 1 | BAIXO PARA O MODELO DE NEGÓCIO | BAIXA | BAIXA | ELEVADO | ELEVADO IMPACTO | DIFICULDADE: ELEVADO |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IDEIAS | C1 | C2 | C3 | C4 | C5 | TOTAL |
| Criar uma base de dados secundária e atualizá-la em tempo real | 5 | 4 | 4 | 4 | 5 | 22 |
| Trabalhar a transformação de dados em uma pipeline | 5 | 5 | 5 | 5 | 5 | 25 |
| Utilizar uma estratégia multicloud | 4 | 5 | 4 | 5 | 5 | 23 |

## Solução

Para diminuir a carga de consultas na base de dados de produção que o processamento de um relatório traz, a solução escolhida foi criar uma base de dados secundária, e utilizar o recurso de CDC (Change Data Capture) para mantê-la atualizada. E uma API que será utilizada para gerenciar uma fila de processamento de relatórios utilizando o Kafka.

Segue o diagrama referente a solução:

Diagrama

Descrição gerada automaticamente

### 

### 1.2.1 Objetivo SMART

Com relação a execução dos objetivos deste projeto, espera-se alcançar os seguintes resultados expostos na tabela SMART abaixo:

|  |  |
| --- | --- |
| S (Specific) | Melhorar o processamento de relatórios |
| M (Mensurable) | Gerar relatórios sem que haja erros |
| A (Attainable) | Criação de base de dados isolada para o processamento dos relatórios |
| R (Relevant) | Estabilidade da aplicação |
| T (Time based) | Aumento da confiança dos clientes dentro de um mês |

### 1.2.2 Premissas e Restrições

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RISCO | PROBABILIDADE | IMPACTO | AÇÃO A SER TOMADA | DESCRIÇÃO |
| Relatórios ficarem desatualizados por muito tempo | BAIXO | BAIXO | Criar uma fila de atualização em tempo real. | Os dados não podem ficar desatualizados por mais que 5 minutos |

### 1.2.3 Backlog de Produto

O backlog será gerenciado utilizando a ferramenta Trello.

A **Sprint 0** será onde apresentamos o problema de solução referente ao projeto.

Na **Sprint 1** iremos trabalhar a replicação dos dados utilizando o recurso de CDC, com a ferramenta Debezium.

Na **Sprint 2** vamos criar a pipeline para transformas nossos dados, para facilitar o processamento dos nossos relatórios.

Na **Sprint 3** vamos analisar o impacto das melhorias, e corrigir possíveis problemas de performance.

Na **conclusão final** vamos realizar a análise conclusiva do projeto, possíveis revisões para o relatório final.

Interface gráfica do usuário

Descrição gerada automaticamente

# 2. Área de Experimentação

## 2.1 Sprint 1

### 2.1.1 Solução

#### Evidência do planejamento

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

#### Evidência da execução de cada requisito:

* **Criar ambiente de desenvolvimento com SQL Server 2017 + Kafka + Debezium**

O ambiente foi criado utilizando containers Docker, através de um arquivo docker-compose.yml.

Texto

Descrição gerada automaticamente

#### **Evidência dos resultados:**

Texto

Descrição gerada automaticamente

* **Criar script para adicionar o conector do SQL Server no Kafka Connect**

Foi Criado um script bash para adicionar o conector no Kafka Connect

Texto

Descrição gerada automaticamente

#### **Evidência dos resultados:**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* **Criar script SQL para ativar o Change Data Capture no servidor do SQL Server 2017**

Foi Criado um script T-SQL para configuração do CDC(Change Data Capture) em todas as tabelas do banco de dados.

Texto

Descrição gerada automaticamente

#### **Evidência dos resultados:**

Texto

Descrição gerada automaticamente

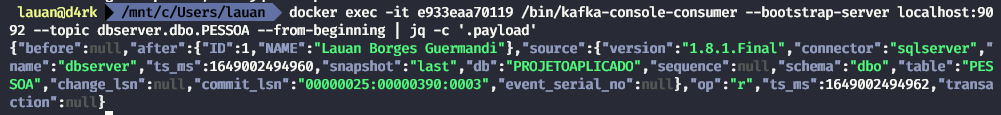
* **Criar teste de streaming de dados**

Para testar a se as alterações dos dados estão sendo adicionados a um tópico no Kafka de forma correta, foi utilizado o kafka-console-consumer. Ao inserir dados na tabela pessoa, os dados foram enviados ao tópico no Kafka.

#### **Evidência dos resultados:**

Tela de celular com aplicativo aberto

Descrição gerada automaticamente



### 2.1.2 Lições aprendidas

Após a execução das atividades previstas na Sprint 1, tivemos algumas lições aprendidas:

* O intervalo entre a inserção dos dados e o envio da mensagem ao Kafka é muito baixo.
* Para cada tabela o Debezium gera um tópico no Kafka.
* Podemos utilizar de vários consumidores.

## 2.2 Sprint 2

### 2.2.1 Solução

#### Evidência do planejamento:

#### Interface gráfica do usuário, Texto, Aplicativo Descrição gerada automaticamente

#### Evidência da execução de cada requisito:

* + **Adicionar serviço do PostgreSQL no ambiente de desenvolvimento**

Foi adicionado o a configuração do PostgreSQL no docker-compose.yml.

**Evidência dos resultados:**

Serviço do PostgreSQL executando.

#### Texto Descrição gerada automaticamente

* **Criar script para configurar o PostgreSQL Sink Connector no Kafka Connect**

Foi Criado um script bash para adicionar o conector PostgreSQL Sink no Kafka Connect.

Texto

Descrição gerada automaticamente

**Evidência dos resultados:**

Texto

Descrição gerada automaticamente

* **Testar replicação de dados com a estrutura real de banco de dados**

Inserir dados nas tabelas no banco SQL Server e verificar se os dados foram replicados no MongoDB.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Evidência dos resultados:**

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente com confiança média

* **Estruturar aplicação de relatório**

Criação da estrutura inicial da aplicação para gerar relatórios.

Tela de computador com fundo preto

Descrição gerada automaticamente

**Evidência dos resultados:**

O código pode ser visualizado no seguinte [link](https://github.com/LauanGuermandi/Projeto-Aplicado/tree/master/src).

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

### 2.2.2 Lições aprendidas

Após a execução das atividades previstas na Sprint 2, tivemos algumas lições aprendidas:

* É necessário um conector de sincronização para cada tabela do banco de dados.

## 2.3 Sprint 3

### 2.3.1 Solução

#### Evidência do planejamento:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

#### Evidência da execução de cada requisito:

* + **Adicionar *producer* para a fila de processamento de relatório**

Foi criada uma estrutura de classes para produzir as mensagens para o kafka utilizando o pattern Adapter.

Texto

Descrição gerada automaticamente

**Evidência dos resultados:**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* **Adicionar *consumer* para o tópico de processamento de relatório**

Foi crida uma estrutura de classes para consumir mensagens utilizando o pattern Adapter.

Texto

Descrição gerada automaticamente

**Evidência dos resultados:**

Este é o log da mensagem consumida do tópico do kafka.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* + **Criar funcionalidade para gerar relatórios no formato CSV a partir da mensagem consumida do kafka**

A partir dos dados consumido no banco de dados é criado um arquivo CSV, com os filtros da mensagem consumida do kafka.

Texto

Descrição gerada automaticamente

**Evidência dos resultados:**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

### 2.3.2 Lições aprendidas

Após a execução das atividades previstas na Sprint 2, tivemos algumas lições aprendidas:

* Utilizar processamento em segundo plano, é uma boa maneira de reduzir o processamento do servidor.
* Podemos desacoplar a utilização de bibliotecas e componentes utilizando o pattern *Adapter*.
* O pattern *Factory* é muito útil quando você precisa criar objetos dinamicamente.

# 3. Considerações Finais

## 3.1 Resultados

Após a implantação desta arquitetura, a aplicação não é mais impactada durante o processamento de um relatório, já que agora temos um banco isolado, com dados atualizados em tempo real, para que a aplicação de relatório utilize para consulta.

Ao utilizar um fila para processamento dos relatórios, para que ocorram de maneira assíncrona, também diminuímos a necessidade de hardware no servidor da API.

## 3.2 Contribuições

O projeto foi inovador, já que a um bom tempo este problema existia na aplicação, mas todas as soluções apresentadas eram paliativas ou com um custo muito alto.

Visto que esta, é uma solução com serviços *open source*, e por utilizar containers Docker podemos realizar o *deploy* com facilidade em várias plataformas, o custo se torna mais baixo.

## 3.3 Próximos passos

Os próximos passos do projeto são:

* Adaptar a POC para que seja publicada no Kubernetes.
* Criar testes automatizados para a aplicação