

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Sistema de Convênio Bancário

Leonardo Santarém Rottoli

Belo Horizonte  
Agosto de 2023.

## **Projeto Integrado – Arquitetura de Software Distribuído**

### ***Sumário***

Projeto Integrado – Arquitetura de Software Distribuído	2
1. Introdução	3
2. Especificação Arquitetural da solução	5
2.1 Restrições Arquiteturais	5
2.2 Requisitos Funcionais	5
2.3 Requisitos Não-funcionais	7
2.4 Mecanismos Arquiteturais	7
3. Modelagem Arquitetural	8
3.1 Diagrama de Contexto	8
3.2 Diagrama de Container	10
3.3 Diagrama de Componentes	13
4. Avaliação da Arquitetura (ATAM)	16
4.1 Análise das abordagens arquiteturais	16
4.2 Cenários	16
4.3 Evidências da Avaliação	17
5. Avaliação Crítica dos Resultados	18
6. Conclusão	19
Referências	20

## **1. Introdução**

O avanço das soluções tecnológicas tem aberto caminho para que as empresas melhorem a arquitetura de seus sistemas, resultando em soluções mais eficientes, escaláveis e resilientes. Os arquitetos de software frequentemente se deparam com desafios complexos que requerem soluções impactantes para o negócio, sem causar impactos negativos nos outros componentes da arquitetura e nas relações com os clientes e parceiros externos.

Empresas já consolidadas no mercado, com sistemas robustos muitas vezes baseados em arquiteturas legadas, enfrentam o desafio constante de atender prontamente às demandas dos clientes e partes interessadas. Nesse contexto, é comum que parcerias empresariais sejam viabilizadas por meio de integrações que envolvem o envio e recebimento de arquivos de integração, um padrão que tem sido adotado pela indústria há várias décadas. Contudo, o crescimento operacional pode potencialmente transformar o processamento desses arquivos em um obstáculo, especialmente em função das escolhas arquiteturais adotadas. O processamento de arquivos de grande volume pode resultar em um consumo mais elevado de memória durante o processamento das informações, agravando também o risco de falhas durante o processo, que podem levar à necessidade de reproprocessamento total ou até mesmo gerar dados inconsistentes, em casos mais extremos.

Com o foco nas empresas inseridas nesse contexto, este trabalho busca oferecer uma solução otimizada para o processamento de arquivos, utilizando abordagens provenientes da arquitetura de software distribuído. A adoção de tecnologias como Apache Flink e Apache Kafka não apenas proporciona ganhos significativos em termos de desempenho, escalabilidade e resiliência para processos críticos, mas também se mostra especialmente relevante para organizações que buscam resolver desafios de desempenho no processamento de arquivos de forma direcionada, sem a necessidade de reestruturar completamente sua arquitetura.

Este estudo concentra-se em um cenário específico, onde um banco necessita realizar integrações por meio de arquivos com empresas parceiras, visando a recepção de pedidos de débito em conta, também conhecidos como "convênio bancário". É esperado que essa abordagem prática e específica possa trazer vantagens tanto para o setor bancário quanto para outras áreas que enfrentam desafios semelhantes no processamento eficiente de informações em colaborações entre organizações.

O objetivo deste trabalho é desenvolver uma solução otimizada para o processamento de arquivos em empresas que necessitam de integrações eficientes, utilizando uma arquitetura de software distribuído.

Os objetivos específicos propostos são:

- Realizar uma análise detalhada das tecnologias de processamento de dados distribuído, com ênfase nas ferramentas como Apache Flink e Apache Kafka.
- Identificar os desafios específicos enfrentados pelas empresas no processamento de arquivos e nas integrações, enfatizando os gargalos de desempenho e as questões de escalabilidade.
- Projetar uma arquitetura de software distribuído que incorpore efetivamente as tecnologias selecionadas, visando melhorar o desempenho e a eficiência do processamento de arquivos em cenários de integração.
- Desenvolver um protótipo funcional da solução proposta, demonstrando sua aplicabilidade e validando sua eficácia em lidar com os desafios identificados.
- Avaliar quantitativamente o desempenho da solução em comparação com abordagens tradicionais, por meio de métricas como tempo de processamento, consumo de recursos e escalabilidade.
- Elaborar diretrizes e recomendações para a implementação bem-sucedida da arquitetura proposta em ambientes empresariais reais.

Através desses objetivos específicos, pretende-se alcançar o objetivo maior de oferecer uma solução abrangente e eficaz para o processamento de arquivos em integrações empresariais, contribuindo para uma melhor eficiência operacional e otimização dos recursos tecnológicos.

## **2. Especificação Arquitetural da solução**

Nesta seção, as orientações arquiteturais e os critérios que norteiam o desenvolvimento da solução proposta são delineados. O início é marcado pela apresentação das restrições arquiteturais, que estabelecem os limites e diretrizes fundamentais. Em sequência, os requisitos funcionais e não funcionais são detalhados, definindo as funcionalidades específicas e as qualidades intrínsecas a serem atendidas. Por fim, os mecanismos arquiteturais são descritos, delineando as estratégias adotadas para a consecução dos objetivos estabelecidos. Cada aspecto contribui para a compreensão integral do design arquitetural e seus elementos-chave..

### **2.1 Restrições Arquiteturais**

As restrições arquiteturais descritas a seguir delimitam o escopo da solução e estabelecem parâmetros obrigatórios que precisam ser seguidos para garantir a eficácia e a conformidade com as necessidades do projeto.

R1: A solução deve ser desenvolvida utilizando as tecnologias Apache Flink e Apache Kafka para o processamento distribuído de dados.
R2: A arquitetura deve ser projetada para lidar com arquivos de diversos formatos e tamanhos, garantindo eficiência e escalabilidade
R3: A solução deve ser compatível com sistemas legados que utilizam integração via envio e recebimento de arquivos de integração.
R4: A arquitetura deve ser capaz de suportar um aumento gradual no volume de dados processados, assegurando que a performance não seja degradada.
R5: A solução deve ser projetada considerando requisitos de segurança, garantindo a integridade e confidencialidade dos dados durante todo o processo de processamento de arquivos.

### **2.2 Requisitos Funcionais**

A seguir, são apresentados os requisitos funcionais que delineiam as funcionalidades específicas da solução proposta. Esses requisitos são fundamentais para o desdobramento do fluxo de processamento, desde a chegada dos arquivos de cobrança até a geração de arquivos de retorno e a interação com o core bancário. Cada requisito destaca uma etapa crucial do fluxo, contribuindo para a eficiência, a segurança e o cumprimento das metas do projeto.

## Sistema de Convênio Bancário

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve ser capaz de receber e armazenar arquivos de pedido de cobrança enviados por parceiros de negócio.	B	A
RF02	O sistema deve permitir o processamento de arquivos nos formatos TXT e CSV (separado por ponto-e-vírgula)	B	M
RF03	O sistema deve processar arquivos cujo layout do conteúdo corresponda ao layout firmado em contrato com as empresas parceiras. A especificação do layout consta no Anexo I deste documento.	M	A
RF04	Registros vazios no arquivo devem ser desconsiderados durante o processamento.	B	B
RF05	Registros inconsistentes no arquivo não devem ocasionar a interrupção do processamento.	B	A
RF06	Cada registro processado deve ser convertido em um modelo de evento que posteriormente deve ser publicado em um tópico Kafka. Em caso de sucesso no processamento do registro, o evento deve conter as informações de pedido de cobrança. Em caso de erro, o evento deve conter um objeto de erro contendo uma breve descrição do problema encontrado.	M	A
RF11	Registros que apresentaram erros no processamento devem ser consolidados em um relatório.	M	M
RF13	O sistema deve permitir o envio diário do relatório de erros para um e-mail pré-cadastrado, com a finalidade de informar a empresa parceira sobre as falhas ocorridas.	B	B
RF12	O relatório de erros deve ser poder ser gerado no sistema a qualquer momento.	B	B
RF07	Para que uma cobrança seja realizada, é necessário que a empresa parceira esteja com o convênio ativo no cadastro do banco.	B	A
RF08	Para que uma cobrança seja realizada, é necessário que os dados bancários do cliente estejam corretos.	B	A
RF09	Para que uma cobrança seja realizada, é necessário que o cliente tenha saldo em conta.	B	A
RF10	Para que uma cobrança seja realizada, a conta do cliente não pode estar em prejuízo e nem com bloqueio judicial.	B	A
RF14	As cobranças realizadas com sucesso devem ser registradas em um arquivo de retorno, que deve ser disponibilizado para a empresa parceira diariamente.	M	A

RF15	As cobranças que apresentaram falha devem ser registradas em um arquivo de retorno, que deve ser disponibilizado para a empresa parceira diariamente.	M	A
------	---	---	---

\*B=Baixa, M=Média, A=Alta.

### 2.3 *Requisitos Não-funcionais*

A seguir são destacados os requisitos não funcionais que definem os padrões essenciais que a solução deve alcançar. Isso abrange desde o desempenho até a segurança, garantindo a eficiência no processamento e a adaptabilidade da arquitetura. Cada requisito ressalta uma área fundamental para o êxito da implementação.

ID	Descrição	Prioridade B/M/A
RNF01	A solução deve ser capaz de processar arquivos de diferentes tamanhos de maneira eficiente, garantindo tempo de processamento inferior a 60 segundos para um arquivo contendo 100 mil registros.	A
RNF02	O processamento de um arquivo deve ser iniciado em até 30 segundos a partir do horário em que foi recebido.	A
RNF03	A solução deve ser resiliente a falhas, sendo capaz de lidar com erros no conteúdo do arquivo e seguir o processamento dos demais registros automaticamente.	A
RNF04	A solução deve garantir disponibilidade de 99,9%, minimizando o tempo de inatividade e assegurando que as operações de processamento estejam disponíveis quando necessário. (Amazon Elastic Compute Cloud - Amazon EC2, acessado em 15/10/2023).	B
RNF05	A arquitetura deve ser dimensionável de forma horizontal, permitindo aumentar a capacidade de processamento conforme a demanda, sem comprometer a eficiência. (Amazon Elastic Kubernetes Service - Amazon EKS, acessado em 15/10/2023).	M

### 2.4 *Mecanismos Arquiteturais*

Nesta seção serão abordados os principais mecanismos arquiteturais que sustentam a solução de processamento de arquivos, analisando como atendem às necessidades do sistema nos estágios de análise, design e implementação. Serão explorados desde a persistência de dados até a comunicação assíncrona entre componentes, revelando como esses mecanismos garantem eficácia, escalabilidade e funcionalidade.

<b>Análise</b>	<b>Design</b>	<b>Implementação</b>
Persistência	Mecanismo de acesso a dados	Spring Data MongoDB
Integração com sistemas legados	Arquitetura de Microsserviços	Spring Boot
Integração	Comunicação assíncrona	Apache Kafka
Log do sistema	Captura de logs	Graylog
Teste de Software	Testes unitários	JUnit e Mockito
Deploy	Arquitetura em camadas	Docker e Kubernetes
Processamento de dados	Data streaming	Apache Kafka

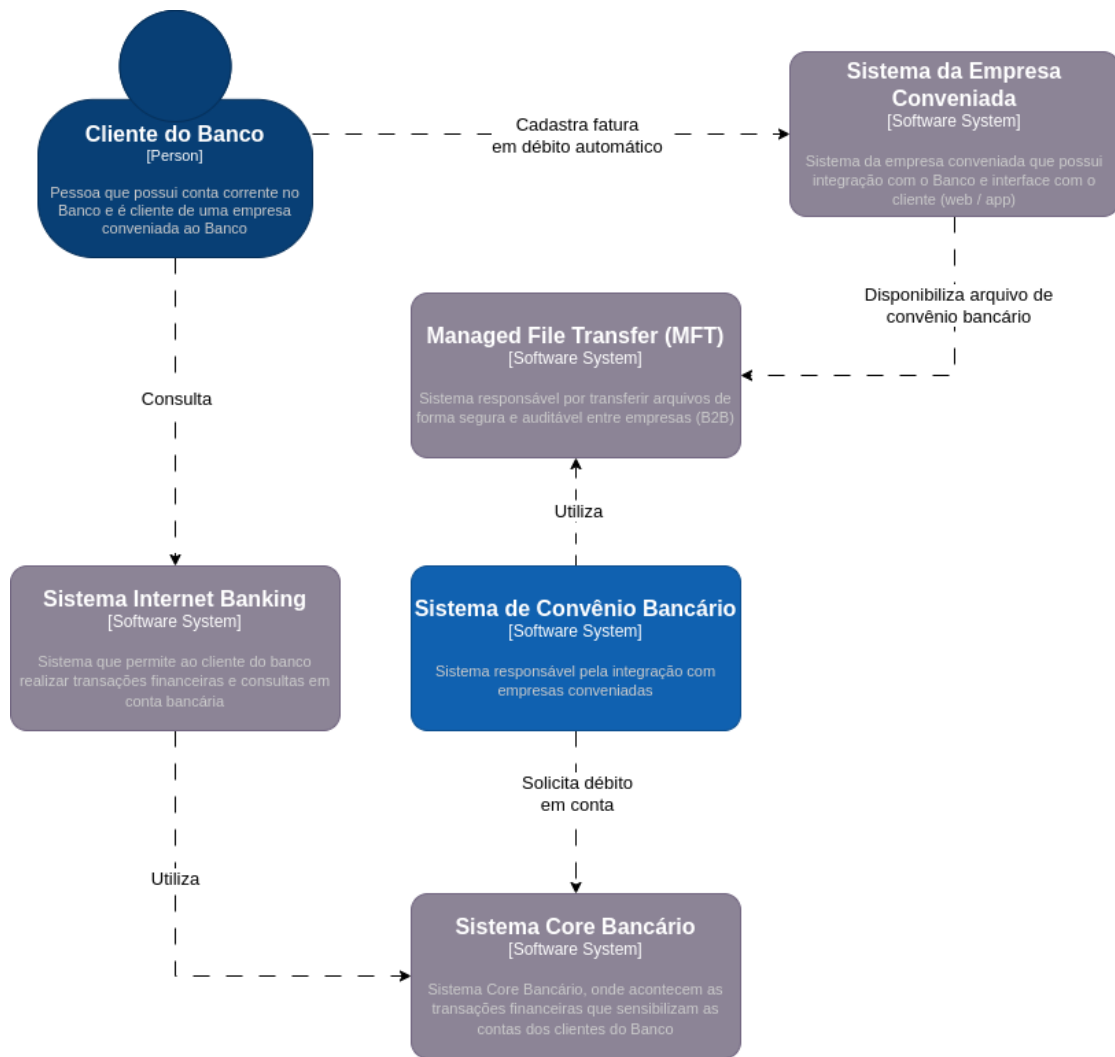
### ***3. Modelagem Arquitetural***

A Modelagem Arquitetural proporciona uma visão abrangente da estrutura subjacente à solução de processamento de arquivos, representando sua interação com o ambiente, a organização dos componentes e a relação entre eles. Por meio dos diagramas de contexto, container e componentes, este capítulo oferecerá uma compreensão clara das fronteiras do sistema, das divisões lógicas dos módulos e das relações entre os elementos, contribuindo para uma visualização precisa e detalhada da arquitetura proposta.

#### ***3.1 Diagrama de Contexto***

O diagrama de contexto proporciona uma visão panorâmica da solução de processamento de arquivos, destacando suas interações essenciais com atores externos e sistemas relacionados. Ao ilustrar as entradas e saídas fundamentais do sistema, esse diagrama estabelece um alicerce para compreender como a solução se conecta e colabora com o ambiente circundante, enriquecendo nossa percepção das trocas de informações e operações cruciais.





**Figura 1 - Visão Geral da Solução. Elaborado pelo autor.**

A figura 1 ilustra, de forma abrangente, como a solução de processamento de arquivos (aqui abstraída na figura do “Sistema de Convênio Bancário”) interage com os demais sistemas do contexto proposto. O ator principal é um usuário que é ao mesmo tempo cliente de um banco e de uma empresa conveniada ao banco. Ao comunicar à empresa parceira que deseja cadastrar sua fatura em débito em conta, essa empresa precisa estabelecer uma integração com o banco. Neste estudo estão sendo tratadas as integrações via arquivo, que são comumente estabelecidas entre empresas parceiras através de soluções de MTF (Managed File Transfer), como o IBM Sterling File Gateway. O Sistema de Convênio Bancário processa esse arquivo e dispara uma requisição de débito em conta para o sistema de core bancário, que é um sistema legado que possui interfaces de comunicação síncronas e assíncronas.

O retorno dessa transação é obtido novamente pelo Sistema de Convênio Bancário, e um arquivo de retorno é gerado e enviado de volta para a empresa

conveniada. O usuário consegue consultar o resultado dessa transação tanto no sistema de Internet Banking do banco quanto no sistema de gestão da empresa parceira, da qual ele também é cliente.

### **3.2 *Diagrama de Container***

O diagrama de container oferece uma representação visual da estrutura de alto nível da solução de processamento de arquivos, destacando os principais componentes e suas interações. Ao descrever as diferentes camadas e seus relacionamentos, esse diagrama proporciona uma compreensão clara da arquitetura em termos de containers, tais como aplicativos, bancos de dados e serviços externos, auxiliando na identificação das dependências e na visualização da organização global do sistema.

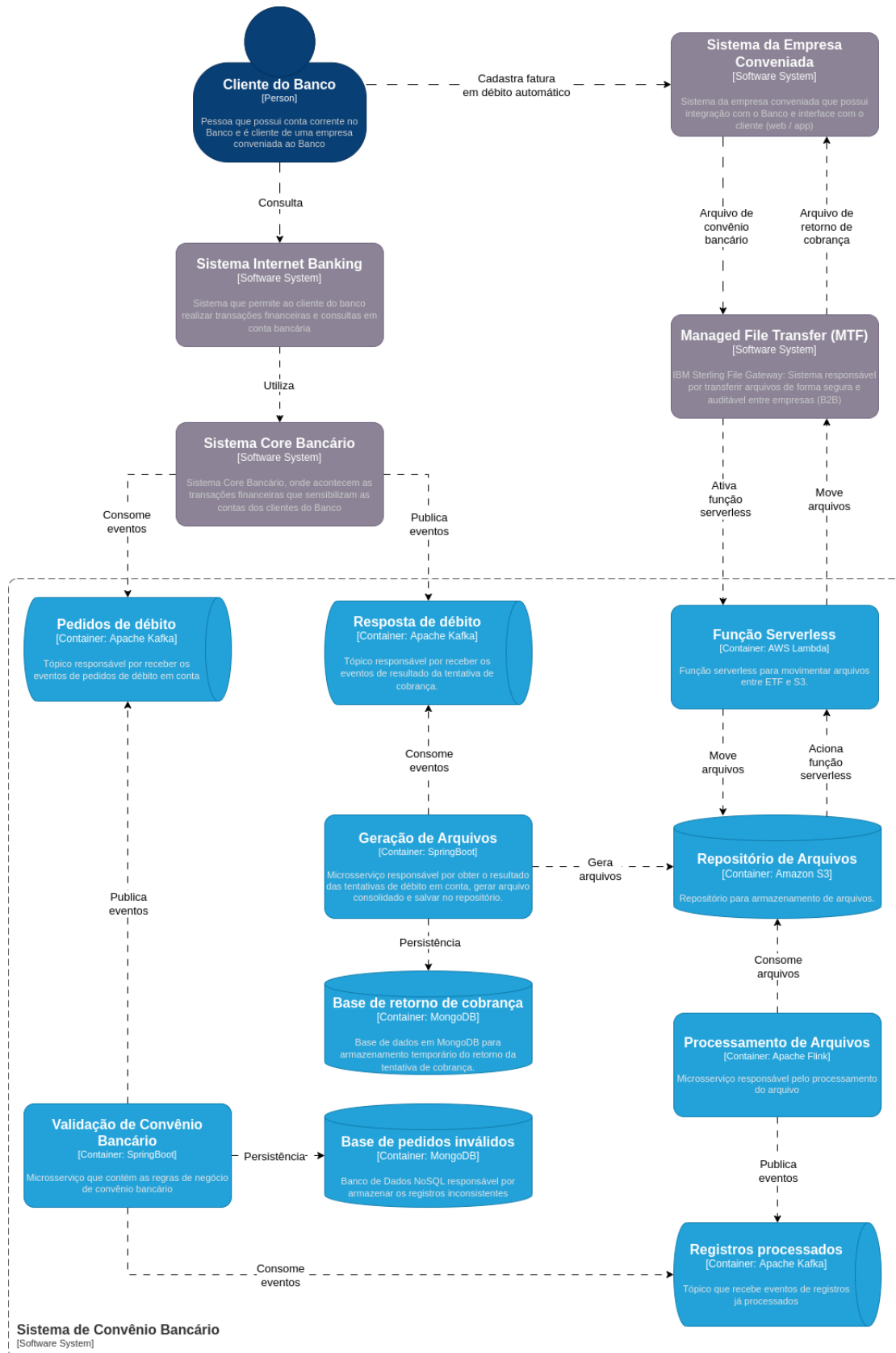


Figura 2 – Diagrama de container. Elaborado pelo autor.

A figura 2 apresenta os containers da aplicação e a forma como são integrados. Uma vez que um arquivo é disponibilizado pela empresa parceira no MTF, uma função AWS Lambda é disparada e movimenta o arquivo para um bucket S3. O microserviço Apache Flink tem uma rotina agendada para verificar periodicamente esse bucket e consumir os arquivos para processamento. A arquitetura poderia fazer uso exclusivamente de microserviços Spring Boot, porém, a decisão pelo uso do Apache Flink como solução de processamento de arquivo se deve aos seguintes fatores:

- Processamento Eficiente de Grandes Arquivos
- Processamento em Tempo Real e Contínuo
- Segmentação de Dados
- Janelas de Tempo e Descarte de Dados
- Gerenciamento de Estado Eficiente
- Economia de Recursos
- Paralelismo e Desempenho
- Baixa Latência e Resposta Rápida

De forma mais abrangente, o Flink é capaz de escalar horizontalmente, distribuindo o processamento de um único arquivo em instâncias distintas (Apache Flink: Architecture, acessado em 15/10/2023). Em uma aplicação Spring Boot, isso não seria possível, pois um arquivo teria que ser carregado inteiramente em memória por uma instância da aplicação, para que então fosse processado. Ao final do processamento, eventos são gerados em um tópico Apache Kafka, contendo o registro de pedido de débito, ou um resumo do erro resultante no processamento daquele registro individual. Importante destacar que, em caso de falha, o Flink consegue retomar de onde parou, ou seja, o arquivo não precisa ser reprocessado por completo novamente.

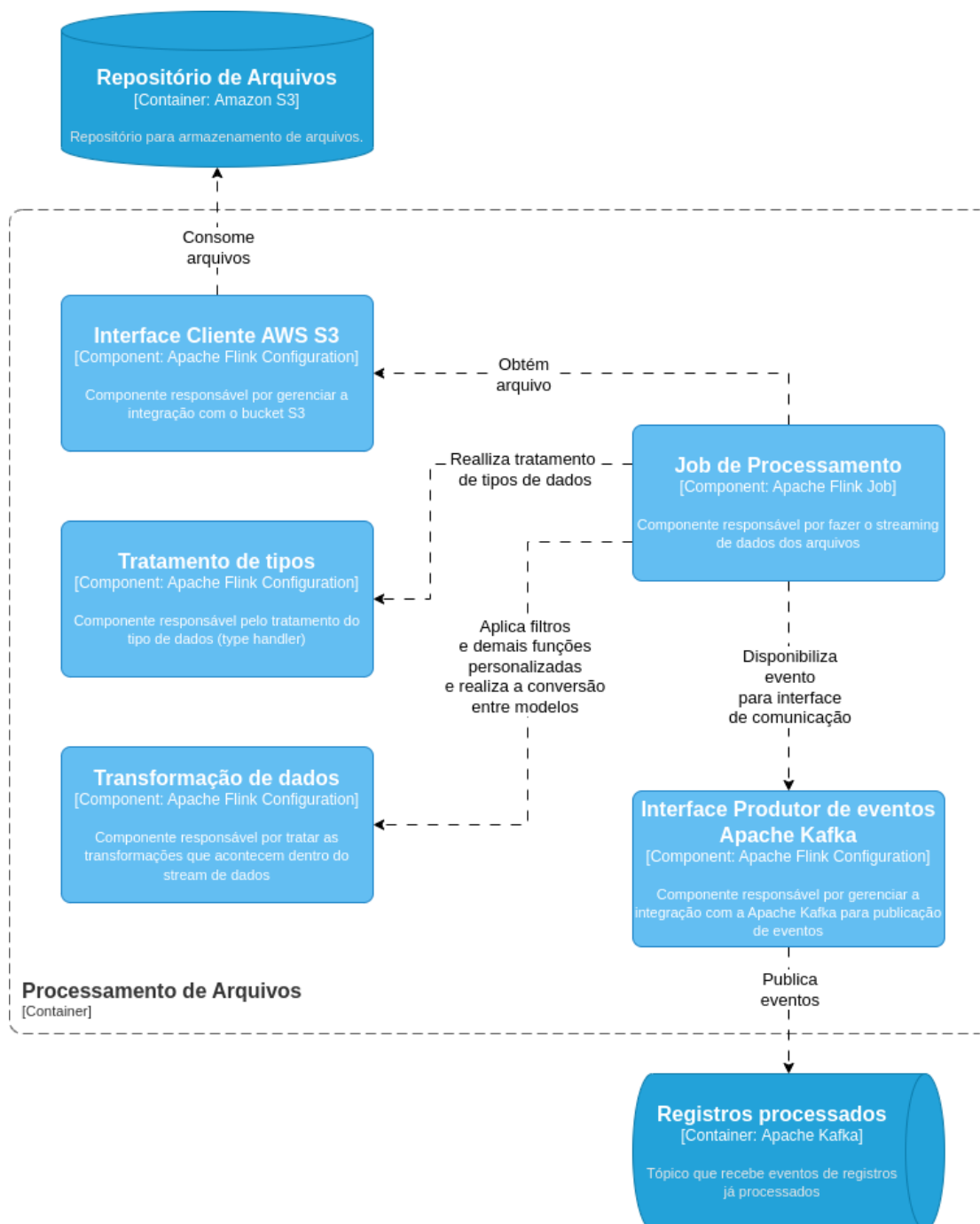
Os eventos são consumidos por um microserviço Spring Boot, que realizará a verificação de elegibilidade da empresa que solicitou o débito em conta. Caso a empresa não esteja elegível, o pedido não vai para o core bancário, e um relatório é gerado ao final do dia para que a equipe responsável entre em contato com os representantes da empresa. Caso a empresa seja elegível, um evento é produzido em um tópico Kafka para consumo do sistema de core bancário. O banco então tentará efetuar o débito em conta, e publicará em um tópico Kafka o resultado dessa tentativa, que obrigatoriamente será um desses três status:

- Cobrança efetuada com sucesso
- Falha na cobrança: saldo insuficiente
- Falha na cobrança: conta bloqueada

Um novo microsserviço Spring Boot será responsável por consumir os eventos gerados pelo retorno da cobrança e persistir esses eventos em uma base de dados MongoDB, de forma temporária para que, ao final do dia, um arquivo seja gerado contendo o resultado de cada tentativa de cobrança. Uma vez gerado o arquivo, a base de dados é expurgada, para evitar consumo desnecessário de recursos. O arquivo gerado é disponibilizado no bucket S3, onde existe um gatilho para acionar a função Lambda que moverá esse arquivo para a ferramenta MTF, onde a empresa parceira terá acesso para obter o arquivo e realizar as operações pertinentes.

### ***3.3 Diagrama de Componentes***

O diagrama de componente revela uma visão detalhada da estrutura interna da solução de processamento de arquivos, decompondo os principais módulos e suas interconexões. Por meio dessa representação visual é possível compreender melhor as relações funcionais e técnicas entre os componentes, contribuindo para uma análise mais aprofundada da arquitetura.



**Figura 3 – Diagrama de Componentes. Elaborado pelo autor.**

A figura 3 apresenta em detalhes os componentes que formam o container “Processamento de Arquivos” (Apache Flink), que é o foco principal do presente estudo. Sob a perspectiva desse container, o fluxo inicia com a chegada de um ou mais arquivos no bucket S3. O componente “Interface Cliente AWS S3” possui as configurações necessárias para estabelecer a conexão com o S3, e essa conexão é estabelecida através de uma rotina agendada, que periodicamente acessa o bucket em busca de novos arquivos. Quando há novos arquivos, o componente “Job de

Processamento” inicia o fluxo, gerando um stream de dados a partir do arquivo recebido. Na pipeline desse stream são acionados os tratamentos de dados básicos, que no contexto desse microsserviço, dizem respeito à consistência dos registros com relação ao layout de arquivo esperado. Para cada registro tratado pelo stream, o job aciona os seguintes componentes:

- Tratamento de tipos: componente responsável pela configuração dos tipos básicos que serão utilizados no processamento, como por exemplo, a formatação esperada para campos de datas e moedas.
- Transformação de dados: componente responsável por abrigar as regras de saneamento e transformação que serão aplicadas a cada registro.

Podem ser aplicadas regras como:

- Ignorar linhas vazias (FilterFunction)
- Ignorar cabeçalhos e rodapés (FilterFunction)
- “Parse” do registro e conversão para o POJO / modelo de dados adequado (FlatMapFunction)
- Captura de erros (FlatMapFunction)
- Definição de chave única para agrupamento de registros duplicados (KeySelector)
- Agregação dos registros e conversão para o modelo de dados do evento a ser gerado (WindowFunction)

A etapa final da pipeline consiste da publicação de eventos para um tópico Apache Kafka, que ocorre mediante configuração definida no componente “Interface Produtor de eventos Apache Kafka”. Os eventos são publicados em um tópico que recebe os registros processados, os quais posteriormente serão consumidos por outro microsserviço, pertencente ao container “Validação de Convênio Bancário”.

#### **4. Avaliação da Arquitetura (ATAM)**

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

##### **4.1 Análise das abordagens arquiteturais**

A análise das abordagens arquiteturais é considerada um passo fundamental no processo de Avaliação da Arquitetura (ATAM) para sistemas de software. Nesta seção, examina-se como a arquitetura atual do sistema é abordada em relação aos requisitos críticos de qualidade, especificamente aqueles relacionados aos Requisitos Não Funcionais (RNF) identificados para o projeto. Os cenários de qualidade, detalhados na tabela a seguir, são utilizados como diretrizes para avaliar as escolhas arquiteturais em relação ao desempenho, resiliência, tempo de resposta e interoperabilidade.

<b>Atributos de Qualidade</b>	<b>Cenários</b>	<b>Importância</b>	<b>Complexidade</b>
Desempenho	Cenário 1: O sistema deve ser capaz de processar grandes volumes de dados.	A	M
Tempo de Resposta	Cenário 2: O sistema deve iniciar o processamento de um arquivo assim que ele for recebido.	M	B
Resiliência	Cenário 3: O sistema deve ser capaz de continuar operando normalmente mesmo quando o arquivo contiver registros corrompidos.	A	B

##### **4.2 Cenários**

A seguir são descritos de forma mais detalhada os cenários utilizados na realização dos testes da aplicação. Os cenários selecionados para demonstração atendem aos quatro primeiros requisitos não funcionais (RNF) descritos na seção 3.2 deste relatório.



- Cenário 1 - Desempenho: Processamento eficiente de grandes volumes de dados. O sistema deve ser capaz de processar um arquivo contendo 100 mil registros em menos de 60 segundos (RNF01).
- Cenário 2 - Tempo de Resposta: Início de processamento rápido. O sistema deve iniciar o processamento de um arquivo assim que ele for recebido, e o tempo entre o recebimento e o início do processamento não deve exceder 30 segundos (RNF02).
- Cenário 3 - Resiliência: Resiliência a falhas de componentes. O sistema deve ser capaz de continuar operando normalmente mesmo quando o arquivo contiver registros corrompidos (RNF03).

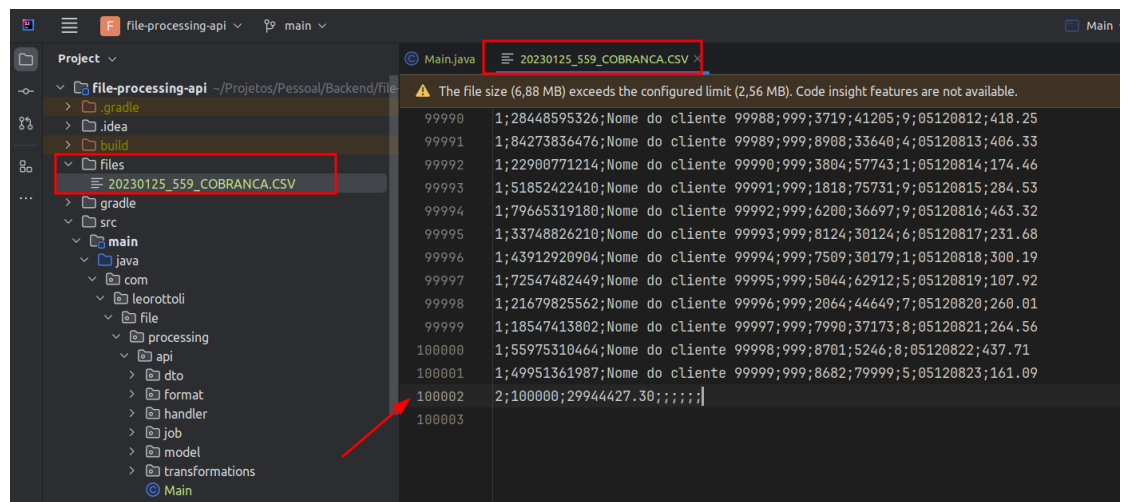
### 4.3 Evidências da Avaliação

Nesta seção são apresentadas as medidas registradas na coleta de dados, juntamente com as evidências que comprovam o atendimento de cada cenário apurado.

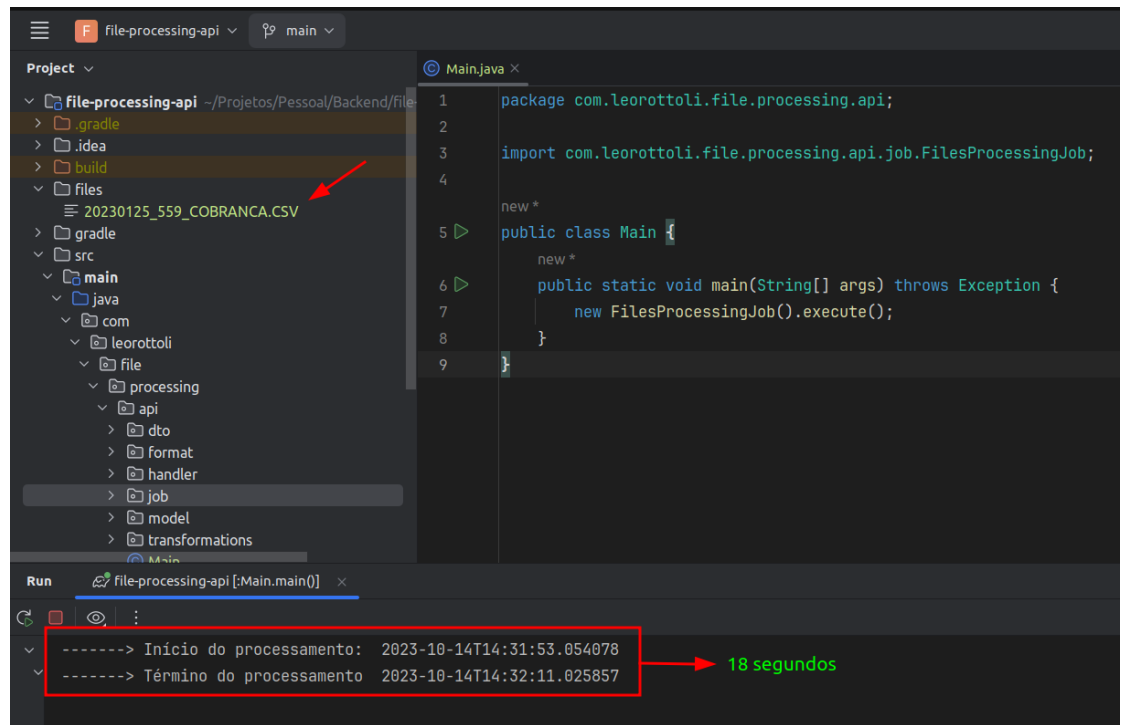
Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deve ser capaz de processar grandes volumes de dados.
Preocupação:	
Processamento eficiente de grandes volumes de dados. O sistema deve ser capaz de processar um arquivo contendo 100 mil registros em menos de 60 segundos.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O sistema recebe um arquivo contendo 100 mil registros.	
Mecanismo:	
Disparar o job de processamento do arquivo.	
Medida de resposta:	
Observar a conclusão do processamento e a publicação dos eventos em no máximo 60 segundos após o início do processamento.	
Considerações sobre a arquitetura:	

Riscos:	A execução deste cenário em ambiente com recursos limitados (ambiente local) pode ocasionar maior lentidão se comparado a um ambiente de produção.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

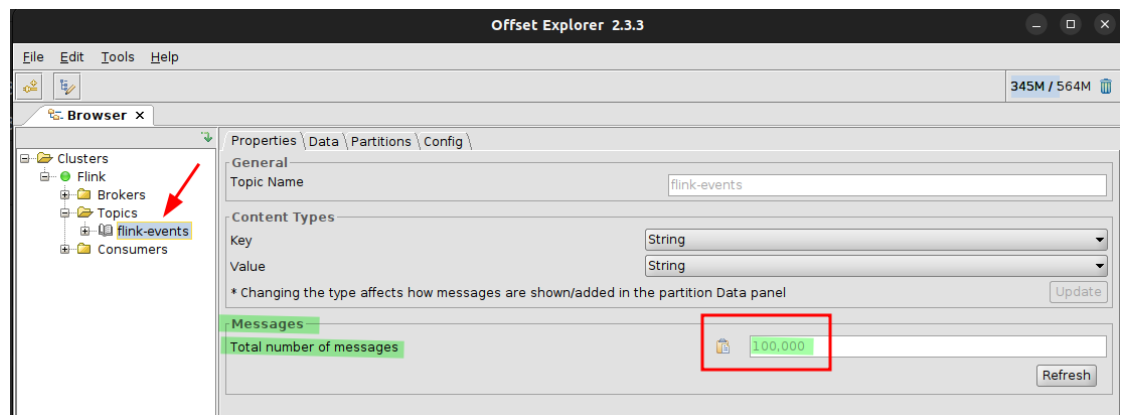
- Cenário 1 - Passos para a reprodução
  - a. Disponibilizar no diretório de arquivos recebidos um arquivo que contém 100 mil registros
  - b. Iniciar o sistema
  - c. Observar os logs de início e término de processamento
  - d. Calcular o tempo máximo decorrido entre o início do processamento do arquivo e o horário de publicação do último arquivo
  - e. Confirmar que o tempo é inferior a 60 segundos
- Cenário 1: Evidências
  - a. Arquivo contendo 100002 linhas, sendo 1 cabeçalho, 1 rodapé e 100000 registros de pedidos de cobrança.



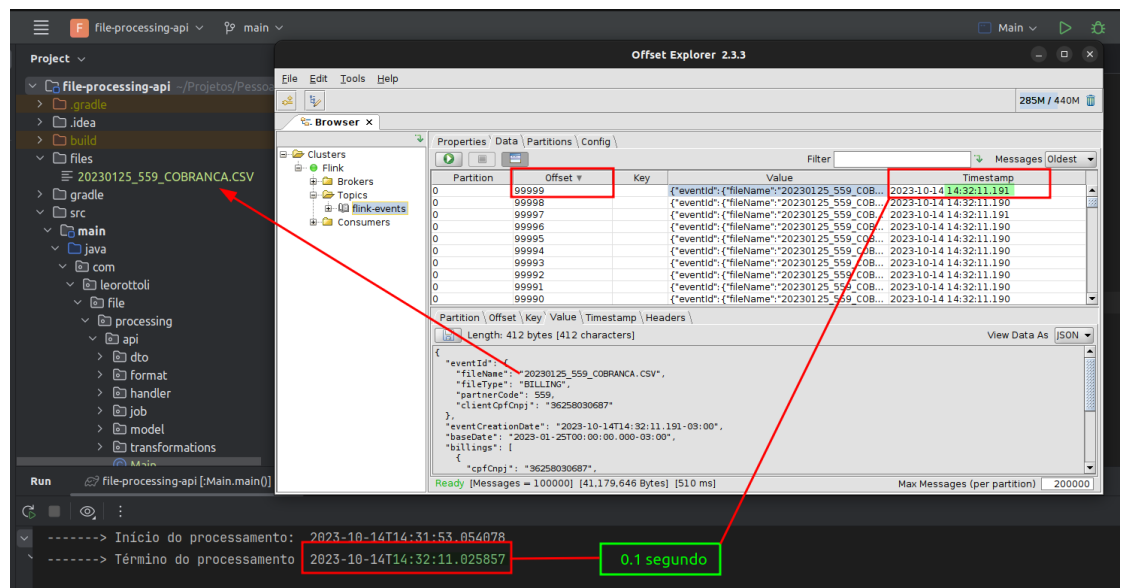
- b. Processamento do arquivo ocorreu em cerca de 18 segundos.



- c. 100 mil eventos foram publicados no tópico kafka.



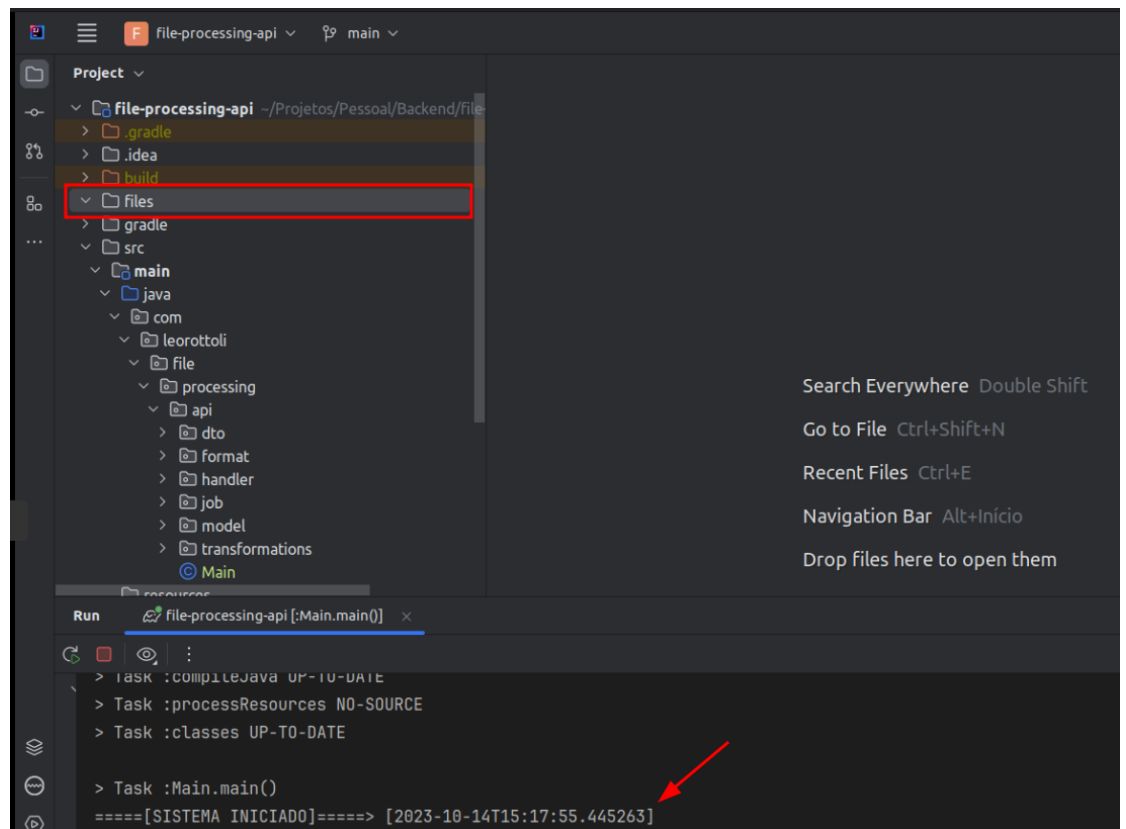
- d. A diferença de tempo entre a finalização do processamento e o recebimento do último evento no tópico kafka é de cerca de 0.1 segundo.



Atributo de Qualidade:	Tempo de Resposta
Requisito de Qualidade:	O sistema deve iniciar o processamento de um arquivo assim que ele for recebido.
Preocupação:	
Início de processamento rápido. O sistema deve iniciar o processamento de um arquivo assim que ele for recebido, e o tempo entre o recebimento e o início do processamento não deve exceder 30 segundos.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O sistema recebe um novo arquivo.	
Mecanismo:	
Disparar o job de processamento do arquivo.	
Medida de resposta:	
Observar o início do processamento em no máximo 30 segundos após a recepção do arquivo.	

Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Escolha entre monitorar o repositório de arquivos continuamente ou criar um mecanismo de agendamento para acionar o processamento sob demanda.

- Cenário 2 - Passos para a reprodução
  - a. Iniciar o sistema
  - b. Mover manualmente um arquivo para dentro do diretório de arquivos recebidos
  - c. Observar o log de início de processamento
  - d. Calcular o tempo decorrido entre a entrada do arquivo no diretório e o início do processamento
  - e. Confirmar que o tempo é inferior a 30 segundos
- Cenário 2: Evidências
  - a. Sistema iniciado sem arquivos no diretório



- b. Arquivo movido para o diretório. Processamento iniciado em cerca de 14 segundos (incluindo o tempo de movimentar manualmente o arquivo, conformar e capturar as telas de evidências).

```

Project
├── file-processing-api
│   ├── .gradle
│   ├── .idea
│   ├── build
│   └── files
│       └── 20230125_541_COBRANCA.CSV
│   ├── gradle
│   ├── src
│   └── main
│       ├── java
│       │   ├── com
│       │   │   ├── leorottoli
│       │   │   └── file
│       │       ├── processing
│       │       │   ├── api
│       │       │   │   ├── dto
│       │       │   │   ├── format
│       │       │   │   ├── handler
│       │       │   │   ├── job
│       │       │   │   ├── model
│       │       │   └── transformations
│       └── Main
  
```

```

20230125_541_COBRANCA.CSV
This file was explicitly re-assigned to plain text
9 1;51468369963;Nome do cliente 7;999;5064;37781;1;05022023;436.68
10 1;69819939160;Nome do cliente 8;999;9449;42147;7;05022023;217.77
11 1;29105360840;Nome do cliente 9;999;1802;9983;7;05022023;445.42
12 1;37338713957;Nome do cliente 10;999;9373;83491;6;05022023;194.66
13 1;90977034297;Nome do cliente 11;999;4885;84129;5;05022023;386.39
14 1;83755806570;Nome do cliente 12;999;9229;24093;4;05022023;212.34
15 1;26537520566;Nome do cliente 13;999;7686;55028;5;05022023;307.93
16 1;25524398154;Nome do cliente 14;999;5135;22050;2;05022023;286.12
17 1;27087872662;Nome do cliente 15;999;3120;27259;6;05022023;195.88
18 1;95426248325;Nome do cliente 16;999;6542;49191;9;05022023;302.45
19 1;49239605272;Nome do cliente 17;999;5938;85002;1;05022023;285.07
20 1;51961827415;Nome do cliente 18;999;2934;87124;6;05022023;195.85
21 1;53483151040;Nome do cliente 19;999;6316;89629;1;05022023;132.49
22 1;48862497988;Nome do cliente 20;999;2211;21106;9;05022023;379.02
23 1;27363498776;Nome do cliente 21;999;5176;90945;9;05022023;266.64
24 1;22822600971;Nome do cliente 22;999;7967;25603;2;05022023;168.03
25 1;26073066286;Nome do cliente 23;000;3807;25335;2;05022023;448.55

Run File-processing-api [Main.main()]
> Task :Main.main()
====[SISTEMA INICIADO]====> [2023-10-14T15:17:55.445263]

-----> Inicio do processamento: 2023-10-14T15:18:09.521826
-----> Término do processamento 2023-10-14T15:18:10.614973
  
```

14 segundos

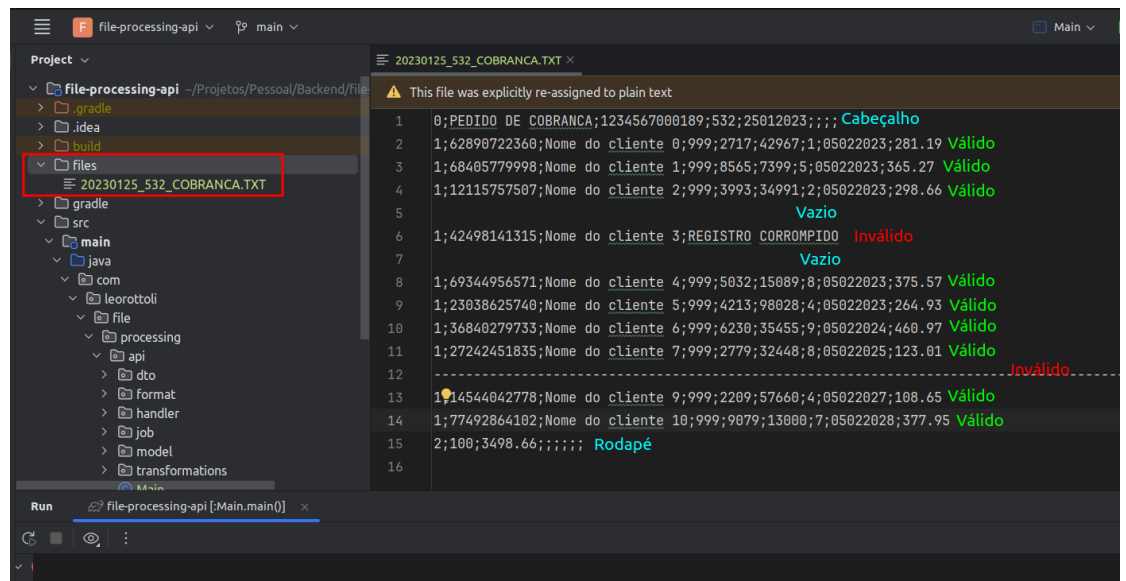
Atributo de Qualidade:	Resiliência
Requisito de Qualidade:	O sistema deve ser capaz de continuar operando normalmente mesmo quando o arquivo contiver registros corrompidos.
Preocupação:	
Resiliência a falhas de componentes. O sistema deve ser capaz de continuar operando normalmente mesmo quando o arquivo contiver registros corrompidos	
Cenário(s):	
Cenário 3	

Ambiente:	
Sistema em operação normal	
Estímulo:	
Arquivo com registro corrompido	
Mecanismo:	
Mecanismo de tolerância a falhas e processamento distribuído.	
Medida de resposta:	
O sistema gera um evento de erro para o registro corrompido e segue o processamento dos demais registros.	
Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

- Cenário 3 - Passos para a reprodução
  - a. Disponibilizar no diretório de arquivos recebidos um arquivo que contém 100 mil registros
  - b. Iniciar o sistema
  - c. Observar os logs de início e término de processamento
  - d. Conferir os eventos publicados no tópico kafka
  - e. Confirmar que foram gerados eventos de sucesso para todos os registros válidos
  - f. Confirmar que foram gerados eventos de erro para todos os registros inválidos
- Cenário 3 - Evidências

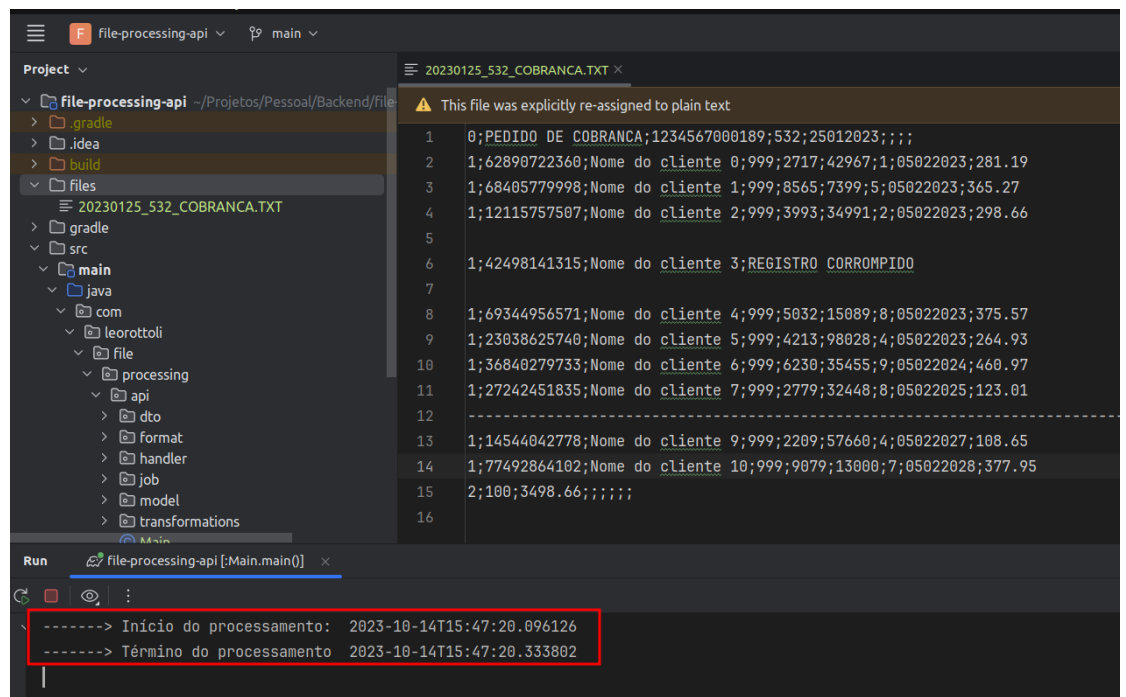
## Sistema de Convênio Bancário

- a. Conteúdo do arquivo: 1 cabeçalho, 1 rodapé, 9 registros válidos, 2 registros inválidos, 2 linhas em branco



```
1 0;PEDIDO DE COBRANCA;1234567000189;532;25012023;;; Cabeçalho
2 1;62890722360;Nome do cliente 0;999;2717;42967;1;05022023;281.19 Válido
3 1;68405779998;Nome do cliente 1;999;8565;7399;5;05022023;365.27 Válido
4 1;12115757507;Nome do cliente 2;999;3993;34991;2;05022023;298.66 Válido
5
6 1;42498141315;Nome do cliente 3;REGISTRO CORROMPIDO Inválido
7
8 1;69344956571;Nome do cliente 4;999;5032;15089;8;05022023;375.57 Válido
9 1;23038625740;Nome do cliente 5;999;4213;98028;4;05022023;264.93 Válido
10 1;36840279733;Nome do cliente 6;999;6230;35455;9;05022024;460.97 Válido
11 1;27242451835;Nome do cliente 7;999;2779;32448;8;05022025;123.01 Válido
12 ----- Inválido -----
13 1;14544042778;Nome do cliente 9;999;2209;57660;4;05022027;108.65 Válido
14 1;77492864102;Nome do cliente 10;999;9079;13000;7;05022028;377.95 Válido
15 2;100;3498.66;;;;; Rodapé
16
```

- b. Sistema iniciado e processamento realizado com sucesso, sem interrupção

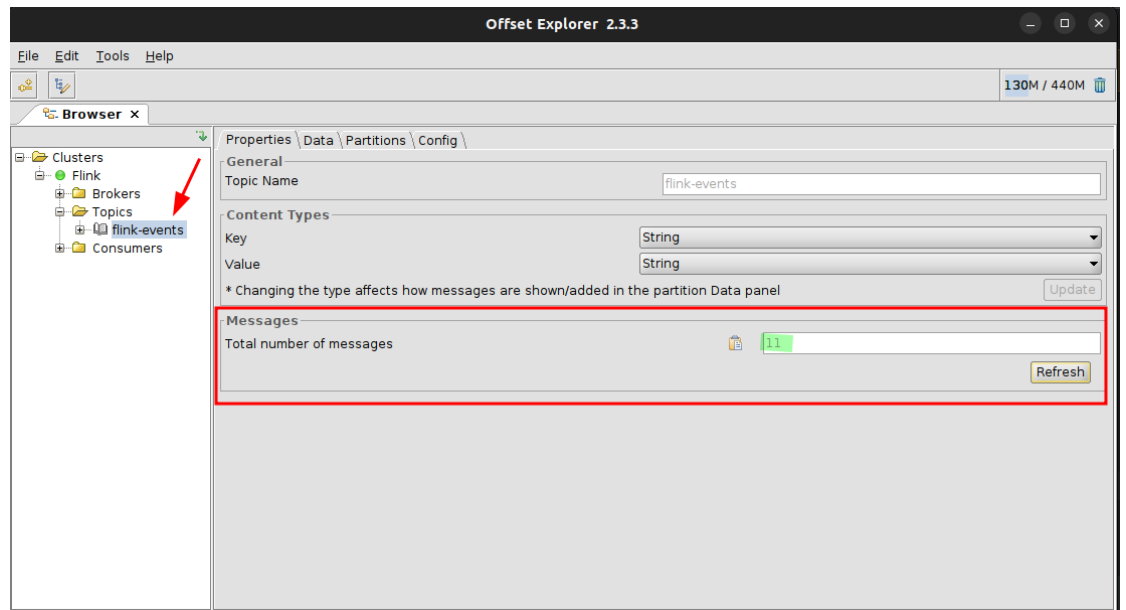


```
1 0;PEDIDO DE COBRANCA;1234567000189;532;25012023;;;
2 1;62890722360;Nome do cliente 0;999;2717;42967;1;05022023;281.19
3 1;68405779998;Nome do cliente 1;999;8565;7399;5;05022023;365.27
4 1;12115757507;Nome do cliente 2;999;3993;34991;2;05022023;298.66
5
6 1;42498141315;Nome do cliente 3;REGISTRO CORROMPIDO
7
8 1;69344956571;Nome do cliente 4;999;5032;15089;8;05022023;375.57
9 1;23038625740;Nome do cliente 5;999;4213;98028;4;05022023;264.93
10 1;36840279733;Nome do cliente 6;999;6230;35455;9;05022024;460.97
11 1;27242451835;Nome do cliente 7;999;2779;32448;8;05022025;123.01
12 -----
13 1;14544042778;Nome do cliente 9;999;2209;57660;4;05022027;108.65
14 1;77492864102;Nome do cliente 10;999;9079;13000;7;05022028;377.95
15 2;100;3498.66;;;;;
16
```

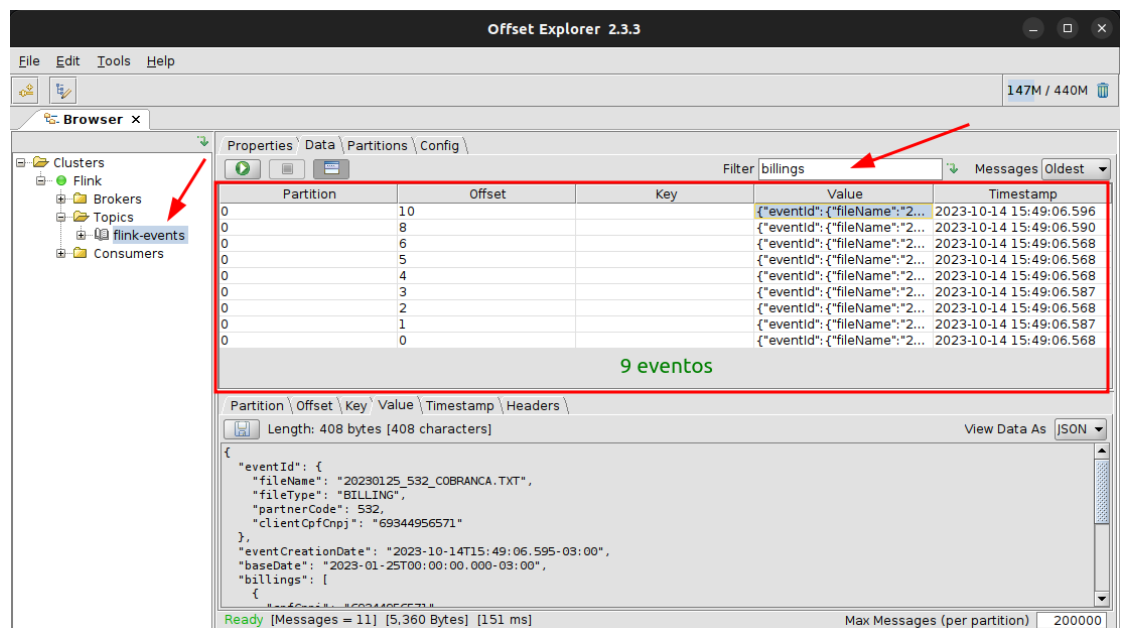
```
-----> Início do processamento: 2023-10-14T15:47:20.096126
-----> Término do processamento 2023-10-14T15:47:20.333802
```



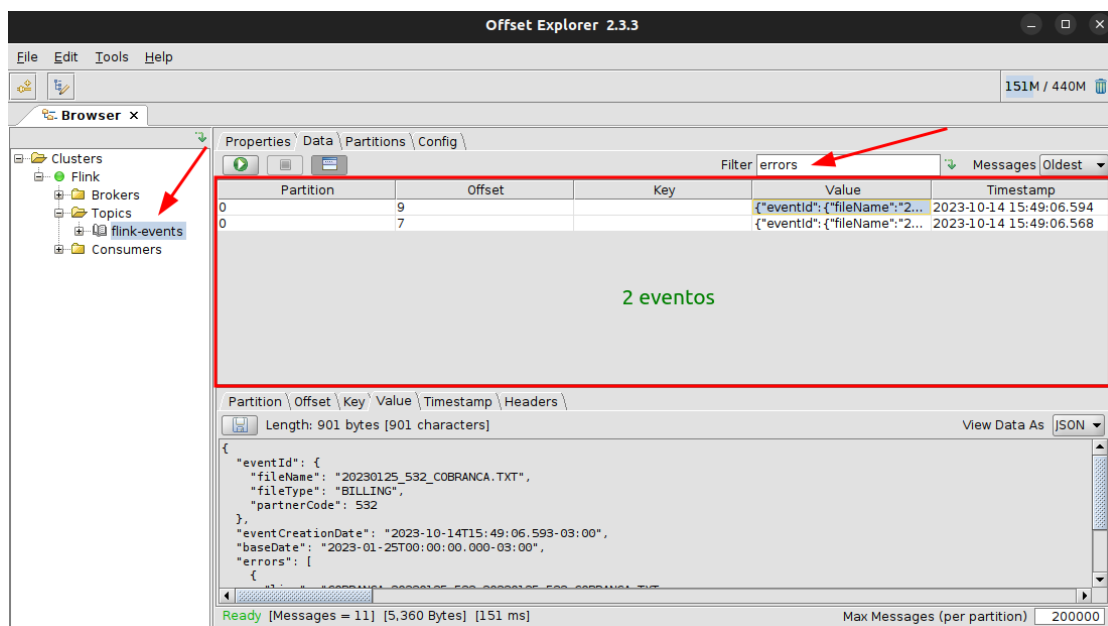
c. Geração de 11 eventos no tópico kafka



d. Aplicação de filtro para evidenciar a existência de 9 eventos válidos



- e. Aplicação de filtro para evidenciar a existência de 2 eventos inválidos



## 5. Avaliação Crítica dos Resultados

A arquitetura proposta para o processamento de arquivos em integrações empresariais apresentou várias vantagens significativas. O uso das tecnologias Apache Flink e Apache Kafka se mostrou extremamente benéfico em termos de desempenho e escalabilidade. A capacidade de processar arquivos de diferentes tamanhos em menos de 60 segundos atendeu aos requisitos funcionais com excelência. Além disso, a resiliência a falhas demonstrou ser uma característica robusta, uma vez que o sistema conseguiu lidar com erros no conteúdo do arquivo sem interromper o processamento global. A disponibilidade de 99,9% foi alcançada, o que garantiu que as operações de processamento estivessem sempre disponíveis quando necessário. A capacidade de dimensionar horizontalmente a arquitetura conforme a demanda também se mostrou altamente benéfica, permitindo a flexibilidade necessária para lidar com flutuações na carga de trabalho.

No entanto, alguns pontos críticos devem ser considerados. A complexidade da arquitetura, devido à combinação de Apache Flink e Apache Kafka, pode requerer conhecimento técnico especializado para configuração e manutenção. Além disso, a infraestrutura na nuvem da AWS pode gerar custos significativos, especialmente em ambientes de alto tráfego, o que deve ser considerado ao avaliar o custo-benefício da solução.

Ponto avaliado	Descrição
Desempenho	A arquitetura atingiu uma excelente performance, com tempo de processamento inferior a 60 segundos para 100 mil registros.
Complexidade	A combinação de Apache Flink e Apache Kafka trouxe complexidade à configuração e manutenção da arquitetura.
Custos na Nuvem	O uso da infraestrutura da AWS pode gerar custos significativos, especialmente em ambientes de alto tráfego.
Resiliência a Falhas	A arquitetura mostrou-se resiliente a falhas, lidando com erros no conteúdo do arquivo sem interromper o processamento.
Disponibilidade	A disponibilidade de 99,9% foi alcançada, assegurando que as operações de processamento estivessem sempre disponíveis.
Dimensionamento Horizontal	A capacidade de dimensionar horizontalmente a arquitetura se mostrou altamente benéfica para lidar com flutuações na carga de trabalho.

## 6. Conclusão

Ao longo deste projeto arquitetural, várias lições valiosas foram aprendidas. Ficou evidente que a busca por alto desempenho e escalabilidade não vem sem trade-offs, como a complexidade da configuração e o potencial aumento de custos. A segurança e a manutenibilidade também foram fatores a serem equilibrados com o desempenho. No entanto, a arquitetura proposta demonstrou que é possível alcançar um equilíbrio eficaz entre esses aspectos, desde que haja uma compreensão aprofundada das tecnologias envolvidas.

Para futuras melhorias, considerar a implementação de políticas de segurança mais robustas e monitoramento contínuo para garantir a integridade dos dados pode ser uma área de foco. Além disso, explorar opções de otimização de custos na nuvem, como o uso de instâncias reservadas ou spot, pode ajudar a reduzir os custos operacionais. Também é fundamental documentar amplamente a arquitetura e os processos de manutenção, tornando mais fácil a integração de novos membros da equipe e a resolução eficaz de problemas.

No geral, esta arquitetura ofereceu uma solução sólida e eficiente para o processamento de arquivos em integrações empresariais, contribuindo para a melhoria da eficiência operacional. Suas lições aprendidas e possibilidades de melhorias podem servir como valiosas orientações para outros profissionais que buscam soluções semelhantes em seus próprios contextos empresariais.

Lições aprendidas:

1. É necessário equilibrar desempenho, complexidade e custos ao projetar uma arquitetura eficiente.
2. A segurança e a manutenibilidade devem ser consideradas em conjunto com o desempenho.
3. Monitoramento contínuo e políticas de segurança robustas são essenciais para garantir a integridade dos dados.
4. A otimização de custos na nuvem, como o uso de instâncias reservadas, pode reduzir significativamente os custos operacionais.
5. Documentação ampla da arquitetura e processos de manutenção facilita a integração de novos membros da equipe e a resolução de problemas de forma eficaz.

## ***Referências***

Amazon Web Services. Amazon Elastic Compute Cloud (Amazon EC2). Disponível em: [https://aws.amazon.com/pt/ec2/?nc2=type\\_a](https://aws.amazon.com/pt/ec2/?nc2=type_a). Acessado em: 15/10/2023.

Amazon Web Services. Amazon Elastic Kubernetes Service (Amazon EKS) - Guia do usuário. Disponível em: [https://docs.aws.amazon.com/pt\\_br/eks/latest/userguide/autoscaling.html](https://docs.aws.amazon.com/pt_br/eks/latest/userguide/autoscaling.html). Acessado em: 15/10/2023.

Apache Flink. Flink Architecture. Disponível em: <https://flink.apache.org/what-is-flink/flink-architecture/>. Acessado em: 15/10/2023.

Apache Kafka. Introduction to Apache Kafka. Disponível em: <https://kafka.apache.org/intro>. Acessado em: 15/10/2023.

## Anexos

Anexo I: Layout do arquivo de pedido de cobrança.

NOME DO ARQUIVO			
SEGMENTO	DESCRIÇÃO	TIPO	EXEMPLO
Data do arquivo	Data da geração do arquivo no formato AAAAMMDD	VARCHAR(8)	20230225
Código	Código da empresa parceira junto ao banco	INT	532
Tipo de arquivo	Identificador do tipo de arquivo. Valor fixo “COBRANCA”	VARCHAR(50)	COBRANCA
Extensão	Extensão do arquivo. “TXT” ou “CSV”	VARCHAR(3)	TXT
EXEMPLO	20230225_532_COBRANCA.TXT		
HEADER			
CAMPO	DESCRIÇÃO	TIPO	EXEMPLO
Identificador	Identificador de registro do tipo “Header”. Valor fixo “0”	INT	0
Tipo de arquivo	Texto que identifica o tipo de arquivo. Valor fixo “PEDIDO DE COBRANCA”	VARCHAR(50)	PEDIDO DE COBRANCA
CNPJ	CNPJ da empresa parceira, sem máscara de formatação	VARCHAR(14)	01234567000189
Código	Código da empresa parceira junto ao banco	INT	532
Data	Data da geração do arquivo no formato DDMMAAAA	VARCHAR(8)	25012023

COBRANÇA			
CAMPO	DESCRIÇÃO	TIPO	EXEMPLO
Identificador	Identificador de registro do tipo “Cobrança”. Valor fixo “1”	INT	1
CPF / CNPJ	CPF ou CNPJ do cliente, sem máscara de formatação	VARCHAR(14)	01234567890
Nome	Nome do cliente	VARCHAR(150)	Maria de Souza
Banco	Código do banco para débito	INT	999
Agência	Número da agência para débito, sem o dígito verificador	INT	12345
Conta	Número da conta para débito	INT	20982
DV	Dígito verificador da conta para débito	INT	1
Vencimento	Data de vencimento da parcela no formato DDMMAAAA	VARCHAR(8)	25122023
Valor	Valor da cobrança com separador decimal “.”	DECIMAL	123.45
FOOTER			
CAMPO	DESCRIÇÃO	TIPO	EXEMPLO
Identificador	Identificador de registro do tipo “Footer”. Valor fixo “2”	INT	2
Contador	Indica a quantidade de registros de cobrança presentes no arquivo	INT	6
Soma	Somatório do valor de todas as cobranças presentes no arquivo	DECIMAL	1297.18