Proposta de Solução para Comunicação de Microsserviços via Mensageria

1. Contexto do Desafio e Objetivos da Solução

1.1 Contexto do Desafio

Nesse desafio, temos um cenário onde:

- É necessário cadastrar novos clientes em um sistema.
- A partir do cadastro de novos clientes, é informada uma proposta de crédito.
- Para clientes aprovados na proposta, é feita a emissão de um ou mais cartões de crédito.
- Pode haver erros na geração da proposta de crédito ou na emissão dos cartões, necessitando um mecanismo de tratamento de erros e resiliência.
- Os microsserviços envolvidos devem se comunicar de forma eficaz e confiável para garantir que todas as etapas sejam concluídas com sucesso.

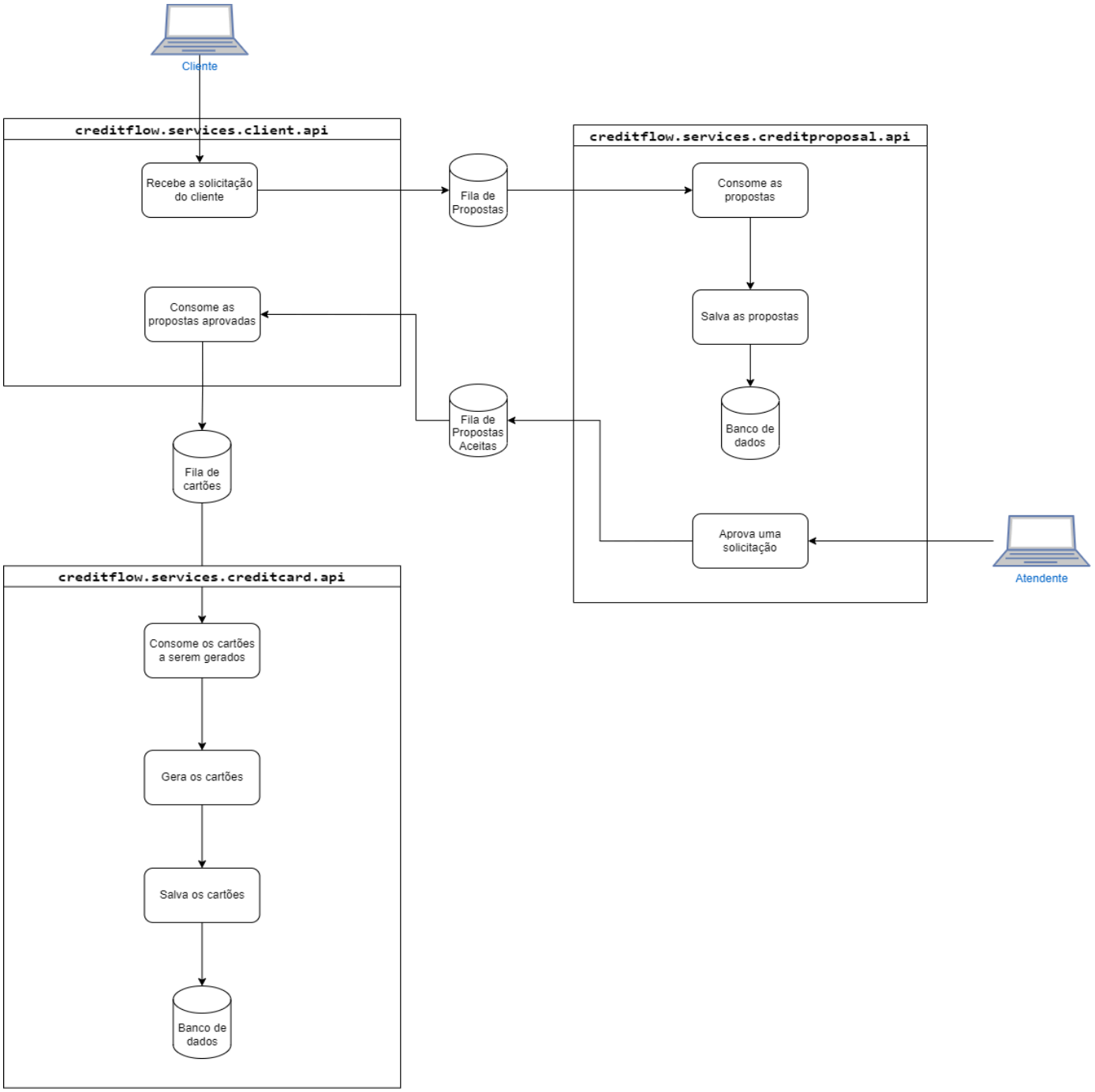
1.2 Objetivos da Solução

A solução proposta tem como objetivos:

- **Comunicação assíncrona entre microsserviços:** Utilizando mensageria para garantir a comunicação eficiente e desacoplada entre os serviços.
- **Desenvolvimento de microsserviços autônomos e escaláveis:** Cada serviço deve ser independente, permitindo escalabilidade conforme a demanda.
- **Implementação de mecanismos de resiliência:** Lidar com falhas de comunicação e outros erros através de mecanismos de reentrega e dead-letter queues.
- **Geração de eventos para notificação:** Informar o microsserviço de clientes sobre o status das propostas de crédito e emissão de cartões.
- **Conformidade com os requisitos técnicos:** Uso de .NET 8.0, RabbitMQ (com o MassTransit) para mensageria, além de boas práticas no design de microserviços.

2. Arquitetura da Solução

2.1 Diagrama de Arquitetura



O diagrama de arquitetura ilustra a solução proposta, incluindo:

- **Microsserviços:**
  - **Cadastro de Clientes (`creditflow.services.client.api`):** Gerencia o cadastro de clientes e inicia o processo de proposta de

crédito. Além de solicitar a emissão de cartões, quando uma proposta é aceita.
    - **Proposta de Crédito (`creditflow.services.creditproposal.api`):** Processa a proposta de crédito e notifica o serviço de clientes.
    - **Cartão de Crédito (`creditflow.services.creditcard.api`):** Emite cartões de crédito para propostas aprovadas.
- **Interações:**
    - Mensagens enviadas e recebidas entre os microsserviços.
    - Notificações de eventos entre os serviços.
- **Fluxo de Mensagens:**
    - O fluxo detalhado das mensagens desde o cadastro até a geração da proposta, emissão do cartão e tratamento de erros.

2.2 Descrição Detalhada dos Microsserviços

2.2.1 Cadastro de Clientes (creditflow.services.client.api)

- **Função:** Gerenciar o cadastro de clientes.
- **Responsabilidades:**
    - Criar e atualizar clientes.
    - Iniciar a geração de propostas de crédito ao cadastrar um novo cliente ou para clientes existentes.
    - Receber notificações de propostas aceitas e enviar mensagens para emissão de cartões.
- **Interações:**
    - Envia mensagens para o serviço de Proposta de Crédito.
    - Consome mensagens do serviço de Proposta de Crédito.
    - Envia mensagens para o serviço de Cartão de Crédito.
- **Lógica de Negócio:**
    - Ao cadastrar um cliente ou ao receber uma proposta para um cliente existente, publica uma mensagem para iniciar a proposta de crédito.
    - Recebe notificações sobre propostas aceitas e emite cartões conforme necessário.

2.2.2 Proposta de Crédito (`creditflow.services.creditproposal.api`)

- **Função:** Gerenciar propostas de crédito.
- **Responsabilidades:**
    - Criar e atualizar propostas de crédito.
    - Aceitar ou negar propostas.
    - Notificar o serviço de clientes sobre propostas aceitas.
- **Interações:**
    - Consome mensagens do serviço de Cadastro de Clientes.

    - Envia mensagens para o serviço de Cadastro de Clientes.
- **Lógica de Negócio:**
    - Processa a proposta de crédito ao receber uma mensagem do serviço de cliente.
    - Envia o resultado da proposta para o serviço de clientes.

2.2.3 Cartão de Crédito (`creditflow.services.creditcard.api`)

- **Função:** Gerenciar a emissão de cartões de crédito.
- **Responsabilidades:**
    - Criar, atualizar cartões de crédito.
    - Bloquear ou expirar cartão.
- **Interações:**
    - Consome mensagens do serviço de Cadastro de Clientes.
- **Lógica de Negócio:**
    - Emite cartões de crédito ao receber uma mensagem de proposta aprovada.

3. Implementação dos Microsserviços

3.1 Implementação do Microsserviço de Cadastro de Clientes

O microsserviço de Cadastro de Clientes gerencia o cadastro e atualização de clientes. Quando um cliente é cadastrado ou é recebido uma proposta de um cliente existente, uma mensagem é publicada na fila do RabbitMQ para iniciar a geração da proposta de crédito. Além disso, ele recebe notificações sobre propostas aprovadas e emite cartões de crédito conforme necessário. Esse serviço permite gerenciar (criar, atualizar e deletar) e pesquisar por clientes.

3.2 Implementação do Microsserviço de Proposta de Crédito

O microsserviço de Proposta de Crédito consome mensagens do RabbitMQ publicadas pelo serviço de cliente. Ele aguarda o processamento da proposta de crédito, determinando se deve ser aprovada ou negada. Em caso de aprovação, envia uma nova mensagem para o serviço de Cadastro de Clientes. Além disso, esse serviço permite gerenciar (atualizar proposta, aceitar, negar e deletar) e obter as propostas, inclusive por cliente.

3.3 Implementação do Microsserviço de Cartão de Crédito

O microsserviço de Cartão de Crédito consome mensagens do RabbitMQ sobre propostas aprovadas e emite cartões de crédito conforme as propostas. Além disso, esse serviço permite gerenciar (atualizar limite, bloquear, expirar e deletar) e obter cartões, inclusive por cliente.

4. Tratamento de Erros e Resiliência

4.1 Filas de Mensagens Persistentes

As filas de mensagens são configuradas para serem persistentes, garantindo que as mensagens sejam armazenadas em disco e entregues mesmo em casos de falhas temporárias dos consumidores ou do próprio sistema de mensageria.

4.2 Reentrega de Mensagens

O sistema de mensageria é configurado para suportar políticas de reentrega de mensagens, permitindo que mensagens que falharam no processamento inicial sejam tentadas novamente. Em casos onde a mensagem não pode ser processada após várias tentativas, ela é movida para uma fila de dead-letters para análise e tratamento manual.

4.3 Dead-Letter Queues

Dead-letter queues são utilizadas para armazenar mensagens que não puderam ser processadas após várias tentativas. Isso permite que erros sejam analisados e corrigidos sem perder a integridade do sistema.

5. Considerações Finais

A solução proposta visa atender aos requisitos do desafio, proporcionando uma arquitetura de microsserviços robusta e escalável utilizando .NET 8.0 e RabbitMQ. A comunicação assíncrona garante que os serviços possam operar de forma desacoplada, enquanto os mecanismos de resiliência e tratamento de erros garantem a confiabilidade e a integridade do sistema. A implementação segue as

melhores práticas de design de microsserviços, assegurando uma solução suave e escalável para o futuro.