Arquitetura de sistemas de pagamentos e padrões de mensagem financeira:

Uma abordagem prática na interpretação de mensagens financeiras

Arthur Cícero Costa Bezerra

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Arthur Cícero Costa Bezerra

Arquitetura de sistemas de pagamentos e padrões de mensagem financeira

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Alisson Vasconcelos De Brito

Novembro de 2023

Catalogação na publicação
Seção de Catalogação e Classificação

B574a Bezerra, Arthur Cicero Costa.
Arquitetura de sistemas de pagamentos e padrões de mensagem financeira: uma abordagem prática na interpretação de mensagens financeiras / Arthur Cicero Costa Bezerra. - João Pessoa, 2023.
38 f. : il.

Orientação: Alisson Brito.
TCC (Graduação) - UFPB/CI.

1. Arquitetura de sistemas de pagamento. 2. Padrão ISO8583. 3. Processamento de mensagem financeira. 4. Framework spring. 5. Desenvolvimento e testes de API. I. Brito, Alisson. II. Título.

UFPB/CI CDU 004.2

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computaçãointitulado ***Arquitetura de sistemas de pagamentos e padrões de mensagem financeira*** de autoria de Arthur Cícero Costa Bezerra, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Alisson Vasconcelos De Brito
Universidade Federal da Paraíba

Prof. Dr. Maelso Bruno Pacheco Nunes Pereira
Universidade Federal da Paraíba

Prof. Dr. Romulo Calado Pantaleao Camara
Universidade Federal da Paraíba

João Pessoa, 24 de novembro de 2023

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

AGRADECIMENTOS

Primeiramente agradeço minha família pelo apoio, em especial aos meus pais João Bezerra e Maria José que sempre me apoiarem na carreira acadêmica e terem proporcionado todo o suporte nessa jornada, e ao meu irmão Lucas por ter sempre me apoiado nos momentos difíceis.

Ao meu orientador, professor Alisson Vasconcelos De Brito por me oferecer oportunidades que mudariam minha vida para sempre.

Agradeço aos meus colegas de curso, por todo o suporte ao longo do curso e palavras de encorajamento.

Agradeço também aos meus professores pela formação e suporte ao longo do curso.

RESUMO

Os meios de pagamentos eletrônicos veem crescendo com o passar do tempo, principalmente os pagamentos via cartão de credito, esse trabalho visa dar uma visão geral do processo para que novos profissionais e interessados na área possam ter um melhor e mais rápido entendimento do processo. Serão explorados os principais atores e conceitos do processo, e para auxiliar de maneira prática, será desenvolvida uma aplicação capaz de interpretar mensagens financeiras baseadas no padrão de mensagem ISO8583.

O trabalho inicialmente investiga a arquitetura de pagamento via cartão de crédito de modo geral, como os principais atores e sistemas envolvidos no processo, os possíveis fluxos de uma transação, métodos de segurança, e por fim, será abordado o estudo de padrões de mensagem transacionais.

Uma vez entendido os principais conceitos quanto à arquitetura de pagamentos, o trabalho aprofundará no padrão de mensagem financeira ISO8583, a partir do desenvolvimento de uma aplicação capaz de interpretar e validar mensagens baseadas nesse padrão.

Palavras-chave: <Arquitetura de sistemas de pagamento>, <Padrão ISO8583>, <Processamento de mensagem financeira>, <Framework Spring>, <Desenvolvimento e testes de API>.

ABSTRACT

Electronic payment methods have been steadily growing over time, especially credit card payments. This work aims to provide an overview of the process so that new professionals and those interested in the field can have a better and faster understanding of the process. The main actors and concepts of the process will be explored, and to assist in a practical way, an application capable of interpreting financial messages based on the ISO8583 message standard will be developed.

The paper initially investigates the general architecture of credit card payment, including the main actors and systems involved in the process, the possible transaction flows, security methods , and finally, the study of transactional message standards.

Once the main concepts regarding payment architecture are understood, the paper will focus into the study of the ISO8583 financial message standard and the development of an application able to valid and read messages based on this standard.

key-words: <Payment system architecture>, <ISO8583 standard>, <Financial message processing>, <Spring Framework>, <API development>.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

SIGLA - NOME COMPLETO

ISO – Organização Internacional para Padronização (do inglês, International Organization for Standardization)

HTTP – Hypertext Transfer Protocol

API – Interface de programação de aplicações (do inglês, Application Programming Interface)

IDE - Ambiente de desenvolvimento integrado (do inglês, Integrated Development Environment)

PDV - Ponto De Venda

POS - Ponto de venda (do inglês, Point Of Sale)

EMV - Europay, Mastercard e Visa

NFC - Near Field Communication

ATM - Automated Teller Machine

SMS - Short Message Service

MCC - Código de Categoria do Comerciante (do inglês, Merchant Category Code)

MTI - Indicador do tipo de mensagem(do inglês, Message Type Indicator)

Sumário

1 INTRODUÇÃO

Com o crescimento dos meios de pagamentos eletrônicos, o desenvolvimento de novas tecnologias para garantir a confiabilidade e segurança no processo transacional é imprescindível. Hoje um dos principais meios de pagamento é o via cartão de crédito/débito, a padronização na comunicação entre os sistemas envolvidos e criação de protocolos é essencial para garantir segurança e eficiência necessárias nos processos de pagamento. O estudo visa contribuir para o entendimento desse processo, a partir do desenvolvimento de uma aplicação capaz de interpretar e validar mensagens financeiras baseadas no padrão ISO8583.

O cartão de crédito, que tem como objetivo facilitar o processo de compra, oferecendo diversos serviços de pagamento ao cliente. Uma de suas principais vantagens é a alta velocidade na aprovação das transações, além da garantia de segurança e confiabilidade, obtida através de diversas formas de validação [1], como o uso de senha e código de segurança.

O processo envolvido na aprovação de uma compra ocorre por meio da troca de mensagens financeiras [12]. Por exemplo, ao realizar uma compra em uma máquina de cartão de crédito de um estabelecimento comercial, as informações da compra realizam um percurso a partir da máquina de cartão até chegarem no banco emissor do cartão, que é onde essas informações serão realmente validadas.

Para que esse processo seja confiável e usado em larga escala inclusive internacionalmente, foi necessário desenvolver um padrão para essas trocas de mensagens. E o primeiro padrão que recebeu larga adoção internacional foi o ISO8583, com sua primeira versão lançada em 1987, esse padrão recebe atualizações frequentemente até hoje, sendo usado pelas principais redes de pagamentos como VISA e Mastercard.

1.1 Definição do Problema

Apesar dessas tecnologias e padrões terem uma longa história, até hoje são lançadas novas atualizações e aprimoramentos, até mesmo para os padrões de mensagem mais antigos. Este trabalho busca acelerar e facilitar a integração de novos interessados na área, apresentando uma visão geral sobre a estrutura de troca de mensagens financeiras e interpretação e validação dessas mensagens, para que seja possível compreender como podemos receber e interpretar essas mensagens por meio de uma aplicação, incluindo a adição de regras para validação do conteúdo da mensagem.

Foi escolhido o framework Spring para o desenvolvimento, devido a sua larga aceitação no mercado e capacidade de proporcionar um ecossistema robusto para criação de APIs(Interface de programação de aplicações), e com o apoio da ferramenta Postman,

podemos mapear diversas requisições da API para facilitar os testes.

A principal motivação do projeto foi facilitar a compreensão geral do funcionamento da arquitetura de transações financeiras via cartão de crédito, através do desenvolvimento de uma aplicação para interpretar essas mensagens financeiras, acredito que novos profissionais da área possam ter uma integração mais rápida.

1.2 Objetivo geral

O objetivo do trabalho é desenvolver uma aplicação flexível capaz de interpretar e aplicar validações em mensagens baseadas no padrão ISO8583 via requisições HTTP, visando auxiliar no entendimento do processo de pagamentos eletrônicos via cartão de crédito.

1.3 Objetivos específicos

1. Usando os conceitos apresentados, definir um padrão de mensagem financeira baseado no ISO8583 para que possa ser interpretado.

2. Desenvolver uma aplicação API capaz de receber essa mensagem e interpretá-la.

3. Desenvolver validações e cenários de teste com o objetivo de avaliar os resultados, desempenho e confiabilidade da aplicação.

1.4 Estrutura da monografia

No capítulo 2, sera abordado de forma geral a arquitetura de pagamentos via cartão de crédito, tecnologias envolvidas, o que motivou a criação de um padrão de mensagem e um foco no padrão ISO8583, que usaremos como base para aplicação mais adiante.

No capítulo 3, vamos definir as ferramentas usadas na construção da aplicação, definir nosso padrão de mensagem baseado em ISO8583 e as estrategias de validações e processamento da mensagem que serão utilizadas na mensagem.

No capítulo 4, será apresentado as etapas do processo de desenvolvimento de uma aplicação para interpretação de uma mensagem baseada no padrão ISO8583, com base nas definições do capitulo anterior. Assim como uma analise dos resultados e como a aplicação se comporta de forma geral.

E por último no capítulo 5, as conclusões obtidas e um questionamento quanto trabalhos futuros visando expandir as funcionalidades da aplicação.

2 CONCEITOS GERAIS E FUNDAMENTAÇÃO TEÓRICA

Nesta seção, será explorada a arquitetura de sistema de pagamento envolvendo cartões de crédito e débito, assim como a apresentação das tecnologias envolvidas no processo. Desde o estímulo da transação pelo portador do cartão até o processamento pelo emissor do cartão. Será analisada a interação entre o cartão e o terminal de pagamento do estabelecimento comercial, a diferenciação entre transações "private label"[2] e "bandeiradas"[3], e por fim, técnicas de seguranças e os tipos de mensagens financeiras.

2.1 Cartão, métodos de captura e interação do portador

Um dos principais instrumentos financeiros é o cartão de crédito, permitindo o portador realizar compras e pagar por bens e serviços. É emitido por instituições financeira, bancos ou empresa de cartão de crédito de forma geral. Ao longo desse trabalho muitas vezes vamos nos referir a essas instituições como emissor.

O cartão de créditos armazena informações necessárias para realizar pagamentos, tais como nome do portador, data de validade, número do cartão, código de segurança, entre outros. No momento da transação esses dados podem ser fornecidos pelo o próprio titular do cartão, por exemplo, ao digitar os dados do cartão para efetuar uma compra online, porém, essas informações também podem ser obtidas diretamente do próprio cartão, como ocorre ao utiliza-lo em uma máquina de cartão de crédito, onde a informação pode ser disponibilizada através de tarja magnética [4] ou chip padrão EMV(Europay, Mastercard, Visa) [5]. A figura 1 representa uma visão geral do cartão de crédito.

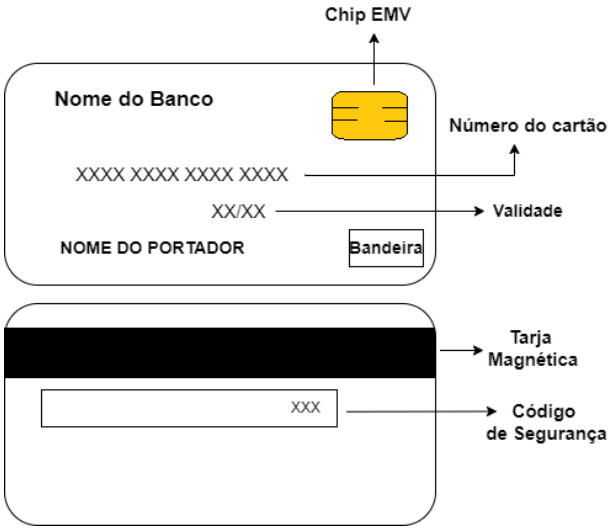

Figura 1: Visão geral dos componentes de um cartão de crédito

Fonte: Autoria própria

Porém, alguns desses métodos vem caindo em desuso, como é o caso da tarja magnética. Esse método contêm dados estáticos e não realiza a criptografia ao se comunicar com os terminais [6], se tornando uma vulnerabilidade para fraudadores. Enquanto métodos de pagamento mais recentes, como os pagamentos via aproximação, se tornaram populares devido ao avanço da tecnologia de Near Field Communication(NFC) [7].

Além da forma como o cartão de crédito transmite os dados, o método de captura dessas informações também é relevante, como, por exemplo, o PDV(Ponto de Venda) ou POS(point of sale) [8], comumente usado em estabelecimentos comerciais, ou o ATM (Automated teller machine)[9], mais conhecido como caixa eletrônico. O tipo de POS/PDV mais conhecido é o aparelho composto por um teclado numérico, leitor de tarja magnética, leitor de chip do cartão, pequena impressora de recibo e conexão a rede[15], no brasil, são popularmente chamadas de "maquininha de cartão de crédito", na figura 2 temos um exemplo desse tipo de POS.

Figura 2: POS maquininha de cartão de crédito com leitor de tarja magnética, leitor de chip, teclado e impressora de recibo

Fonte: Autoria própria

Essas informações inicialmente são recebidas de acordo com os métodos internos de cada empresa responsável pela captura, sendo submetidas a um padrão de mensagem, como o próprio ISO8583, para então seguirem adiante na rede, seja para um fluxo de conexão direta com o emissor ou fluxo envolvendo rede de pagamento bandeirado.

2.2 Fluxo bandeirado e private label

Quando os dados da transação são enviados para a rede, nos deparamos com 2 fluxos, o bandeirado e o private label. Private label são os cartões vinculados a uma rede

específica de varejo, sendo possível realizar transações apenas nas lojas dessa rede, costuma ser um serviço mais customizável, onde o estabelecimento tem controle de praticamente todas as etapas da transação, sem precisar passar pelo intermédio da bandeira. A figura 3 ilustra ambos os fluxos.

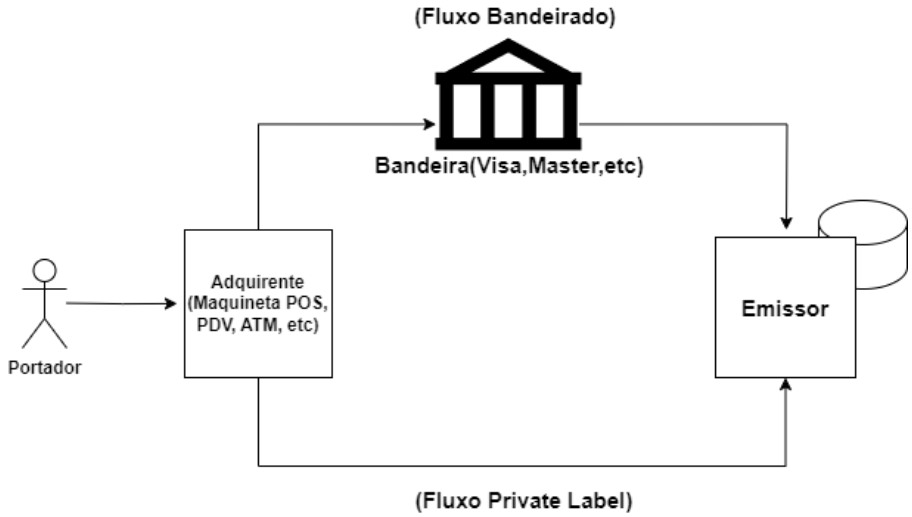

Figura 3: Visão do fluxo bandeira e private label

Fonte: Autoria própria

Os cartões bandeirados possuem empresas responsáveis pelo intermédio em uma transação [6], tais como a VISA, Mastercard, American Express, entre outras, essas empresas, nesse contexto, são classificadas como "bandeiras". Nesse fluxo, a empresa ou banco que pretende emitir e processar cartões, necessita obedecer uma série de certificações e normas estabelecidas pela bandeira [13], o padrão de mensagem usado é descrito por meio de manuais que são atualizados frequentemente. Dentre as principais vantagens do fluxo bandeirado, é possível mencionar a aceitação em uma maior variedade de estabelecimentos, recursos de prevenção a fraude, e a alta disponibilidade da rede.

2.3 Segurança

Segurança é um dos pontos fundamentais para que toda a arquitetura funcione como esperado, cada etapa durante uma transação emprega métodos para combater possíveis fraudes. Iniciando pelo POS [14], que geralmente inclui: a contagem de vezes que o portador errou a senha; o valor máximo permitido em uma transação no modo de pagamento por aproximação sem a necessidade de informar a senha; e a troca de mensagens criptografadas pela rede.

O fluxo bandeirado adiciona uma camada extra de segurança, uma vez que a bandeira valida a transação antes que a mesma continue seu fluxo na rede. O chip

padrão EMV[6], armazena informações do cartão de forma segura e dificulta a clonagem de cartões. A bandeira mantém um monitoramento rígido através de diversos alertas para detecção de anormalidades, também possui serviços como o cálculo de risco da transação, atribuindo um valor que representa a chance da transação ser fraudulenta. A bandeira também inclui serviços para segurança em compras online, como o 3D-Secure [19], que adiciona uma autenticação extra para transação, usando métodos de verificação através de mensagem SMS ou reconhecimento facial. Como o sistema da bandeira é atualizado e validado constantemente, as brechas para fraudes são cada vez menores.

Também são implantadas medidas de segurança do lado do próprio banco emissor do cartão. O banco emissor do cartão faz a validação dos dados do cartão, a partir de uma comparação entre os dados enviados pela mensagem de transação com os dados disponíveis no banco de dados desse emissor, como por exemplo, verificar o número e validade do cartão, valor da transação, saldo do portador, local que foi feita a transação, a verificação do código de categoria do comerciante (MCC)[20], entre outras.

São realizadas algumas validações mais complexas, como a validação de senha e o código de verificação do cartão, código de 3 dígitos no verso do cartão que também pode estar gravado no chip e tarja. Essas validações envolvem o uso de chaves criptografadas para decodificar a informação enviada pela bandeira.

2.4 Padrões de mensagem

A mensagem de transação financeira contém os dados necessários para que a transação venha a ser validada, gerando uma resposta para o portador se sua transação foi aprovada ou não. Para que essas mensagens possam ser utilizadas com confiabilidade e segurança foi necessária a criação de padrões de mensagem, visto que, em um cenário em que cada empresa usa um padrão de mensagens próprio, a integração entre esses sistemas se torna mais complexa e custosa.

Um dos primeiros padrões adotados internacionalmente foi o ISO-8583, simplificando a comunicação entre as empresas de meios de pagamento. São definidos vários campos com diversas informações a respeito da transação, como, por exemplo, número do cartão, data de validade, valor da transação, horário da transação, entre outros. A mensagem em si é escalável quanto a seu tamanho e adição de novas informações, e apesar da ISO(Organização Internacional de Normalização) estabelecer padrões para cada campo da mensagem, as empresas do meio financeiro também podem adaptá-los para suas necessidades particulares, esse será o padrão que usaremos como base para aplicação[16].

Outro padrão mais recente é o ISO-20022, que visa maior flexibilidade, pois poderia ser usado em diversos domínios financeiros, não se limitando a transações de cartão de crédito. As principais características do padrão ISO-20022 são, estrutura hierárquica

facilitando a identificação dos atributos pelo sistema, facilidade na leitura a olho humano e adição de mais campos aumentando a sua flexibilidade[17].

2.5 Padrão ISO8583

O padrão ISO 8583 foi criado inicialmente em 1987 e existem três versões: 1987,1993 e 2003[16], com a primeira versão estabelecendo uma base com os conceitos principais para as versões futuras, em 1993 ganhou uma nova versão, visando abranger uma maior quantidade de informações, também ganhou outra versão em 2003, com objetivo de flexibilizar a definição dos campos considerando as tecnologias emergentes.

Uma mensagem ISO 8583 possui 3 principais componentes: o MTI(message type indicator), mapa de bits e conteúdo da mensagem, esses componentes são montados em sequência na mensagem[12]. Algumas redes adicionam cabeçalhos extras no início da mensagem, seja para indicar o tamanho total ou alguma informação específica. A figura 4 ilustra sua estrutura e tamanho de cada campo.

Figura 4: Estrutura de uma mensagem ISO8583 e tamanho de seus campos

Fonte: Autoria própria

O MTI é um campo de 4 caracteres, que assume valores de acordo com o tipo da transação[10], pode definir, como por exemplo, se é uma transação de crédito ou débito, se é uma requisição ou uma resposta, e outros valores de acordo com as necessidades da rede. Como as mensagens ISO 8583 são lidas da esquerda para direita, o MTI é o primeiro item validado.

O mapa de bits é o segundo componente, ele representa quais bits estão presentes na mensagem, toda mensagem possui pelo menos um mapa de bits primário, no qual indica quais os campos entre o 1 ao 64 estão presentes, quando o campo 1 está presente, significa que existe um segundo mapa de bits, que por sua vez, indica a presença dos

campos 65 ao 128[11]. O mapa de bits é composto por 16 caracteres em hexadecimal, ou 32 no caso de mensagens com mapa secundário. O primeiro mapa é convertido em binário, gerando 64 caracteres binários, caracteres que assumem o valor 1 indicam a presença desse campo de acordo com sua posição. Por exemplo, caso o caractere na terceira posição do mapa de bits assuma o valor 1, significa que a mensagem possui o campo 3 ou bit 3. A figura 5 ilustra a quebra do mapa de bits.

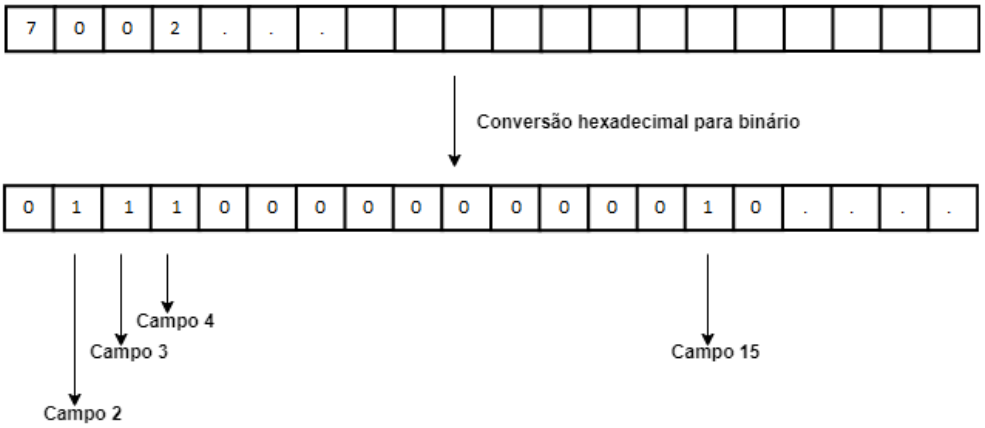

Figura 5: Quebra do mapa de bits em hexadecimal para obtenção dos campos

Fonte: Autoria própria

O conteúdo da mensagem armazena as informações dos campos indicados pelo mapa de bits, no qual são dispostos em ordem crescente. A ISO-8583 especifica o valor desses campos, porém eles são frequentemente adaptados para atender as necessidades de cada rede. Os campos podem possuir tamanho fixo ou variável, campos com tamanho variável possuem um cabeçalho de 2 ou 3 dígitos indicando o tamanho total do campo[11]. Cada rede é responsável pela definição de cada campo do qual faz uso, indicando seu tamanho, os possíveis valores assumidos pelo campo e demais regras vinculadas ao seu envio e/ou resposta de acordo com as características da transação.

Na figura 6, apresenta um exemplo da leitura do conteúdo do campo 2, no qual possui tamanho fixo igual a 4.

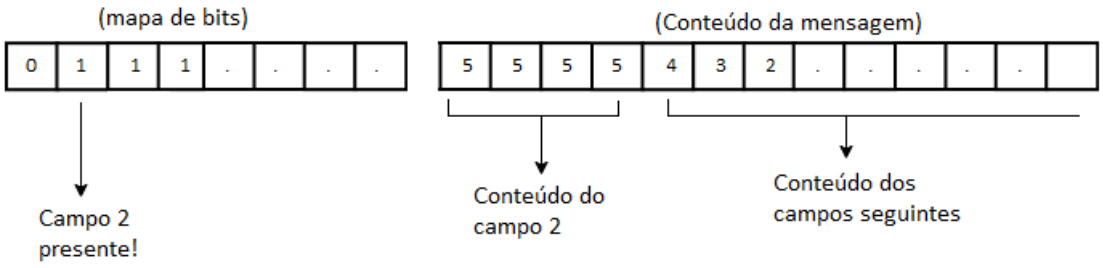

Figura 6: Identificação do conteúdo da mensagem do campo 2 através do mapa de bits

Fonte: Autoria própria

3 METODOLOGIA

Nesse trabalho será desenvolvida uma API capaz de interpretar mensagens baseadas no padrão ISO8583. Por meio de requisições HTTP, será possível enviar uma mensagem de acordo com o padrão ISO8583 definido, no qual terão seus dados interpretados e validados, e por fim, será atribuído um status para a mensagem de acordo com as etapas de validação aplicadas. O resultado da requisição será apresentado no formato JSON, um formato de fácil leitura que usa estrutura de pares chave-valor.

Visando a flexibilidade da aplicação, foi considerado um modelo personalizado seguindo os fundamentos da ISO8583, dessa forma, a aplicação pode ser facilmente adaptada para diversas redes e suas particularidades. Inicialmente foram definidos 10 campos, abrangendo diferentes tipos de campos que podemos nos deparar em trabalhos envolvendo o padrão ISO8583.

Serão definidas estrategias de processamento e validação da mensagem, as validações buscam aumentar a confiabilidade da aplicação atribuindo um status para a mensagem recebida.

3.1 Ferramentas

Para implementação, foi utilizada linguagem de programação Java(versão 17) e suas bibliotecas padrões, o framework Spring Boot(versão 3.1.5), com o auxílio das dependências de desenvolvimento Spring Web e Spring Boot Dev Tools, usando o gerenciador de projeto Apache Maven(versão 3.9.5) e a biblioteca Apache Commons. O ambiente de desenvolvimento usado foi a IDE Intellij(versão 2023.2.3 ultimate edition), e a ferramenta Postman(versão 10.19.10) para realização dos testes e solicitações HTTP.

A aplicação foi construída em um sistema com 16GB de memória RAM, composta por dois módulos de 8GB com frequência de 2400MHz, com SSD de 512GB para armazenamento, processador Intel(R) Core™ i5-12400 de frequência até 4.40GHz e placa de vídeo NVidia GeForce RTX 3060 Ti, com 8GB de memória dedicada.

3.2 Definições dos campos e repostas

Vamos precisar definir alguns campos que estarão presentes no conteúdo da mensagem, para que seja possível explorar os diferentes formatos que cada rede pode vir a usar. Primeiro definimos o nome do campo, na definição do tamanho, consideramos o tamanho numérico fixo e caso possua tamanho variável usamos as notações "LLVAR"e "LLLVAR", onde o "L"representa o tamanho variável do campo e "VAR"o conteúdo, assim campos "LLVAR"podem assumir tamanho de 01-99 enquanto "LLLVAR"de 001 á 999. O tipo

será usado para validação do campo, usando as notações "n", "a"e "an"definimos os campos como numérico, alfabeto e alfanumérico, respectivamente. Por ultimo, definimos a descrição do campo, para facilitar o sua analise.

Na Tabela 1, a definição dos campos iniciais usados como base para desenvolvimento da aplicação.

Tabela 1: Definição dos campos da mensagem

CAMPO	Tamanho	Tipo	Descrição
002	16	n	Número Conta
003	6	n	Tipo Transação
004	8	n	Valor Transação
015	4	n	Validade Cartão
035	8	n	Código Transação
038	2	an	Código Resposta
043	LLVAR	an	Nome Estabelecimento
050	LLVAR	n	País Transação
070	LLLVAR	n	Nome Portador
120	LLLVAR	an	Informações Extras

Com os campos a serem desenvolvidos definidos, já se torna possível montar mensagens de teste baseadas nessa definição. Também será preciso definir os possíveis status da mensagem processada, foi criada uma lista com as possíveis status da mensagem. Como demonstrado na Tabela 2.

Tabela 2: Tabela com possíveis status da mensagem

STATUS
OK
CAMPO NAO PROCESSADO(+CAMPO)
TAMANHO INVALIDO
MENSAGEM COM TIPOS INVALIDOS(+CAMPO)

A nomenclatura "(+CAMPO)"é usada para status que fazem uma indicação mais detalhada do erro, por exemplo, em casos de "CAMPO NAO PROCESSADO"a aplicação também deve ser capaz de mapear quais campos não foram processados, podendo ser gerada uma resposta do tipo: "CAMPO NAO PROCESSADO - 006 007 015", onde o "006", "007"e "015"seriam os campos não processados de fato.

3.3 Recebimento e processamento da mensagem

A aplicação irá receber a mensagem como uma sequência de caracteres, será quebrada em 3 campos: MTI, mapa de bits e conteúdo da mensagem. Dividir esses três campos será nosso primeiro processamento da mensagem. O MTI tem o valor fixo de 4 caracteres, o mapa de bits é recebido em hexadecimal podendo assumir um tamanho de 16 ou 32 caracteres, inicialmente consideramos tamanho 16 e o convertemos para binário, resultando em um mapa de 64 posições, caso a primeira posição seja igual a 1, podemos considerar o segundo mapa de bits, que também será convertido para binário, totalizando 128 posições.

Após separados MTI e mapa de bits, o conteúdo da mensagem será percorrido com base no mapa de bits e a definição dos campos estabelecida anteriormente. Teremos algumas validações quanto ao tamanho e a existência do campo, caso um campo definido no mapa de bits não exista na nossa definição, o status da transação deve ser alterado para "CAMPO NAO PROCESSADO"e caso o tamanho da mensagem não esteja de acordo com o tamanho dos campos na definição será atribuído status "TAMANHO INVALIDO".

Uma vez com a mensagem interpretada e com seus campos definidos, será aplicada uma validação quanto ao tipo de dados permitido, será avaliado campo a campo comparado o tipo recebido e o tipo especifico na definição, caso seja diferente da especificação, o status da transação deve ser alterado para "MENSAGEM COM TIPOS INVALIDOS"seguido da lista de campos com tipos inválidos. A figura 7 ilustra de forma geral o fluxo da aplicação.

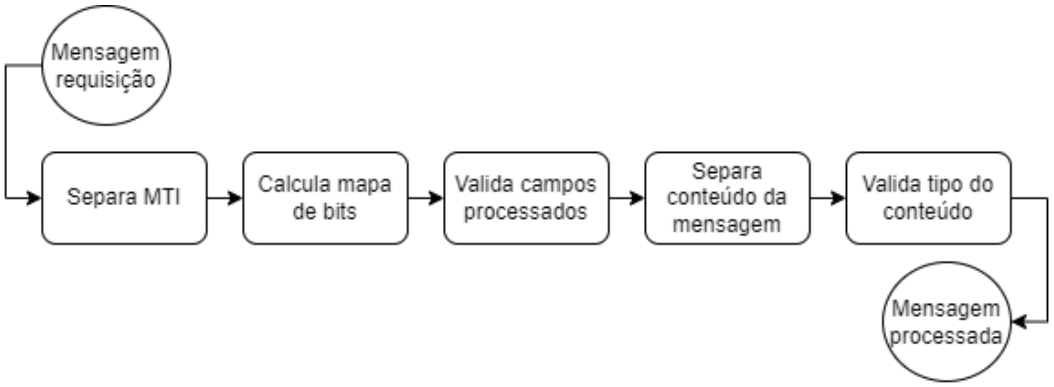

Figura 7: Fluxo de processamento da mensagem

Fonte: Autoria própria

4 DESENVOLVIMENTO E ANÁLISE DOS RESULTADOS

4.1 Implementação API

Como ponto de partida, foi usado o "Spring Initializr"[18] para gerar um projeto base em Spring com as dependências e configurações necessárias. Para realização dessa etapa na IDE Intellij, é necessário apenas criar um novo projeto e selecionar o gerador Spring Initializr, e na opção de linguagem selecionar Java . Na figura 8, é possível observar a estrutura gerada pelo Spring Initializr.

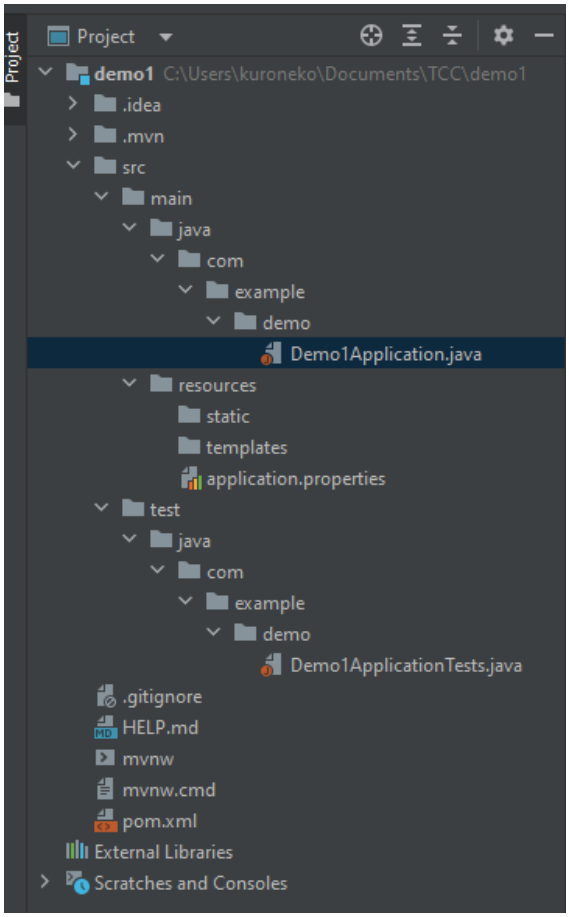

Figura 8: Estrutura de projeto Spring gerado pelo Spring Initializr.

Fonte: Autoria própria

Foi criada uma classe enum "ListaCampos"para mapear as definições da ISO8583 personalizada, definida anteriormente. A classe tipo enum é usada para representar um conjunto de constantes, onde cada constante pode possuir um ou mais valores associados, nesse caso, cada constante representa um campo da mensagem da ISO8583 personalizada. Como demonstrada na figura 9.

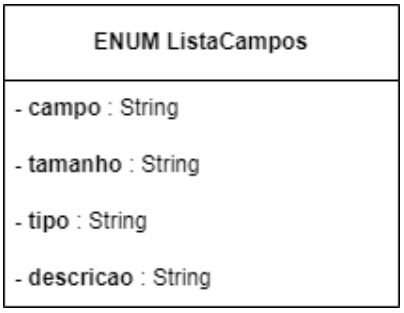

Figura 9: Estrutura da classe enum usada para definir constantes.

Fonte: Autoria própria

Em seguida, criamos uma classe para armazenar os atributos da mensagem processada, considerando o que sera exibido na mensagem resposta. A Figura 10 defini a estrutura da mensagem resposta.

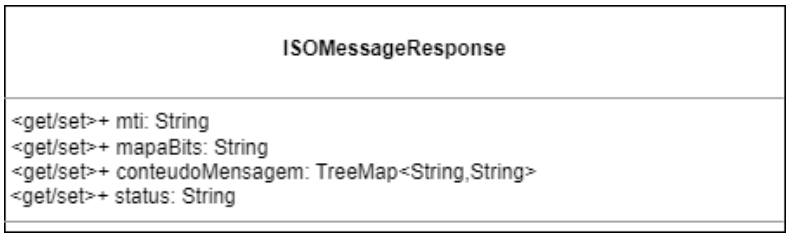

Figura 10: Definição da classe mensagem resposta ISOMessageResponse

Fonte: Autoria própria

Com essas definições, foi criada as classes Controller e Service. O Controller é responsável por definir os endpoints da API, e na classe Service definimos a função principal para interpretar e validar a mensagem ISO8583 personalizada.

Na classe Controller, adicionamos a anotação @Controller do Spring para indicar que é um controlador, ele será o intermediário das solicitações HTTP. Também foi criada uma função para mapear nossa requisição HTTP GET, que será responsável por receber a mensagem ISO8583 personalizada pela API e chamar o Service para tratar e validar a mensagem, mapeada para rota “/parseMsg”.

Na classe Service, adicionamos a anotação @Service do Spring para indicar que o componente é do tipo ”serviço”, nele implementamos o método principal que separa os componentes da mensagem, aplica validações ao conteúdo da mensagem e atribui um status para mensagem. Na figura 11 definimos a estrutura da classe Service.

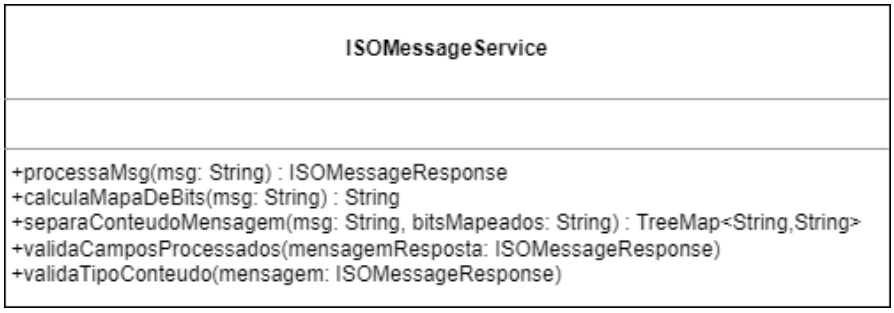

Figura 11: Definição da classe Service

Fonte: Autoria própria

O método principal("processaMsg"), invoca outros 5 métodos para processar a mensagem de acordo com as seguintes etapas: Separar MTI, calcular mapa de bits, validar campos processados, separar conteúdo da mensagem e validar tipo do conteúdo.

No método que separa o MTI, basta obter os 4 primeiros dígitos da String da mensagem da requisição. Para as manipulações de strings, foi utilizada a classe StringUtils da biblioteca Apache Commons Lang.

No método que calcula o mapa de bits, primeiro é verificado se a mensagem possui 1 ou 2 mapas de bits, podendo assumir o tamanho de 16 ou 32 caracteres, por padrão definimos tamanho 16, porém se a primeira posição do mapa de bits for 1, consideramos 32. Para simplificar este processo, podemos considerar o primeiro caractere em hexadecimal do mapa de bits e validar quais deles podem assumir o valor 1 na sua primeira posição.

Após definir o tamanho, podemos recortar o mapa de bits da mensagem e percorrê-lo, cada caractere em hexadecimal será convertido em um binário de 4 bits, foi usado a classe Integer da biblioteca padrão do Java para essa conversão. A figura 12 demonstra a tabela de conversão de hexadecimal para binário, com a indicação dos valores que geram mapa de bits secundário.

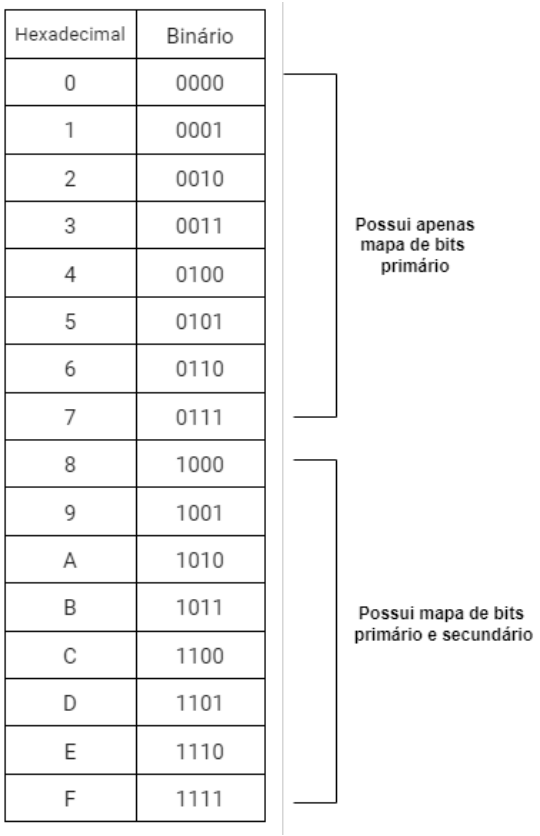

Figura 12: Tabela de conversão hexadecimal para binário e indicação dos caracteres iniciais que geram mapa de bits secundário

Fonte: autoria própria

Ao final da conversão e feita uma validação para conferir se o mapa de bits possui campos que não foram mapeados na definição, para isso foi criado um método que percorre os campos mapeados pelo mapa de bits, caso algum campo ativo(1) não esteja mapeado no enum de campos,é atribuído o status "CAMPO NAO PROCESSADO", seguida da lista de campos não processados enviados na mensagem.

Com o retorno do mapa de bits pelo método anterior, é criado o método que separa o conteúdo da mensagem, associando o campo ao conteúdo. Para representar o conteúdo da mensagem usamos a estrutura treeMap, onde o campo é usado como chave e o conteúdo do campo como valor. Percorremos o mapa de bits e ao verificar uma posição igual a 1, comparamos a posição do mapa de bits em que o "1" foi encontrado com o nome do campo, por exemplo, na posição "2" vamos usar a definição do campo "002", assim carregamos as informações daquele campo. Em seguida, o conteúdo da mensagem é percorrido com base no tamanho do campo usando o auxílio de um índice que incrementa de acordo com o tamanho dos campos, para casos de campos do tipo "LLLVAR" e "LLVAR", o tamanho do campo será considerado pelas próximas 3 ou 2 posições do conteúdo da mensagem respectivamente. Na figura 13 é demonstrado como analisar os tipos LLVAR e LLLVAR.

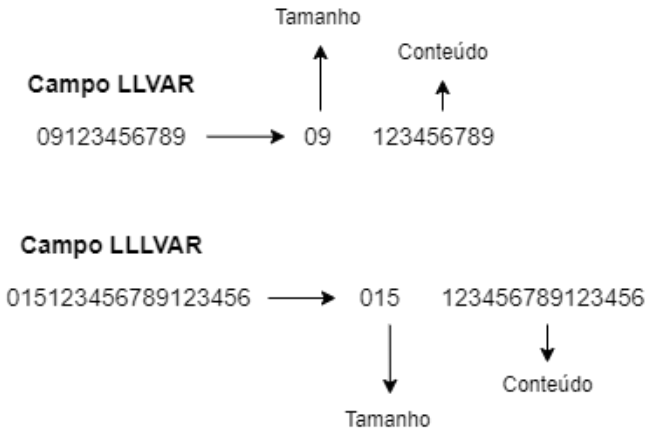

Figura 13: Análise de campos tipo LLVAR e LLLVAR

Fonte: Autoria própria

Após a separação dos campos, é possível aplicar as validações da mensagem. Nesse caso aplicamos validações quanto ao tipo da mensagem. Criamos um método para percorrer os campos da mensagem já interpretada, e comparamos seu conteúdo com base no seu tipo definido no enum "ListaCampos", é criado um switch case, estrutura de controle condicional, para validar cada uma das possibilidades("n","a","an"), foi usado o método "matches"da classe String padrão do Java para comparar o conteúdo do campo com os possíveis caracteres. Caso o conteúdo recebido na mensagem interpretada não esteja de acordo com o definido no enum, o status da mensagem é alterado para "MENSAGEM COM TIPOS INVALIDOS", seguida da lista de campos inválidos.

4.2 Analise dos resultados e testes

Para analisar a aplicação vamos considerar os status esperados e validar se a aplicação os responde corretamente. Ao longo dos testes usaremos as seguintes mensagens:

1. Teste Status "OK-
   0200700200002420400 0
   1111222233334444010000000100002312999999990021
   ESTABELECIMENTO TESTE03BRA

2. Teste Status "OK"com mapa de bits secundário -
   0200F00200002420400004000000000000100
   1111222233334444010000000100002312999999990021
   ESTABELECIMENTO TESTE03BRA010MARIO JOSE0191234567890123456789

3. Teste Campo não processado -
   0200F0820408242040000400000000000100
   1111222233334444010000000100002312999999990021
   ESTABELECIMENTO TESTE03BRA010MARIO JOSE01912345678901234567890

4. Teste Tamanho inválido -
   0200F0020000242040000400000000000100
   11112222333344445010000000100002312999999990021
   ESTABELECIMENTO TESTE03BRA010MARIO JOSE01912345678901234567890

5. Teste Tipos inválidos -
   0200F0020000242040000400000000000100
   11AA222233334444010000000100002312999999990021
   ESTABELECIMENTO TESTE03BRA010MARIO JOSE01912345678901234567890

6. Teste Tipos inválidos(múltiplas ocorrências) -
   0200F0020000242040000400000000000100
   11AA222233334444010000000100002312999999990021
   ESTABELECIMENTO TESTE03BR1010M4RIO JOSE01912345678901234567890

Será usado o Postman para mapear o endpoint criado anteriormente e enviar as requisições. Nele é criado um workspace(ISOMsgParser), no qual é adicionado um novo request, com endereço, porta e o nome do endpoint. Como a aplicação foi executada localmente na porta padrão, teremos um modelo como na figura 14. Em seguida, mapeamos o parâmetro "msg" que foi definido como a entrada na classe controller, sendo responsável pela recepção da mensagem.

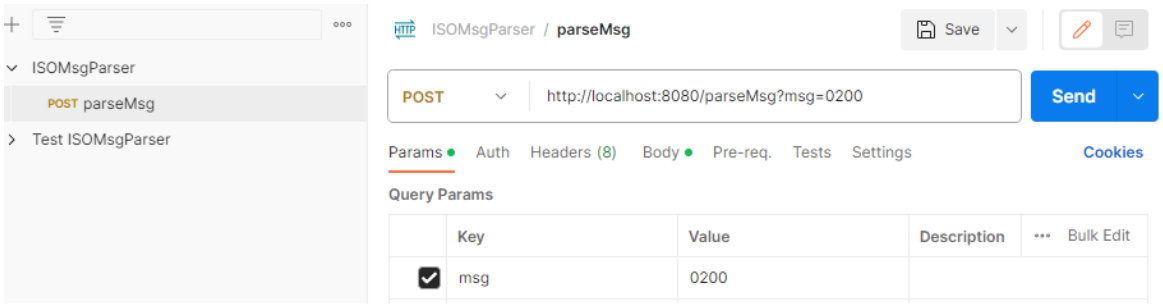

Figura 14: Requisição base na ferramenta Postman

Fonte: Autoria própria

Começando pelo status de sucesso("OK"), as mensagens com mapa de bit primário e secundário foram testadas. Na figura 15 é possível observar o resultado com mapa de bit primário.

```
{
    "mti": "0200",
    "mapaBits": "01110000000000010000000000000000000010010000100000010000000000000",
    "conteudoMensagem": {
        "002": "1111222233334444",
        "003": "010000",
        "004": "00010000",
        "015": "2312",
        "035": "99999999",
        "038": "00",
        "043": "ESTABELECIMENTO TESTE",
        "050": "BRA"
    },
    "status": "OK"
}
```


Figura 15: Mensagem status "OK"com mapa de bits primário

Fonte: Autoria própria

Adicionando os campos 70 e 120, podemos validar mensagens com um segundo mapa de bits, como demonstrado na figura 16.


```
{
    "mti": "0200",
    "mapaBits":
        "1111000000000010000000000000000000010010000100000010000000000000000001000000000000000000000000000000000000000000000000100000000",
    "conteudoMensagem": {
        "002": "1111222233334444",
        "003": "010000",
        "004": "00010000",
        "015": "2312",
        "035": "99999999",
        "038": "00",
        "043": "ESTABELECIMENTO TESTE",
        "050": "BRA",
        "070": "MARIO JOSE",
        "120": "1234567890123456789"
    },
    "status": "OK"
}
```


Figura 16: Mensagem status "OK"com mapa de bits primário e secundário

Fonte: Autoria própria

Nos testes anteriores usamos apenas campos definidos, ao adicionar campos não definidos como 9 e 29 ao nosso mapa de bits, obtivemos o resultado "CAMPO NAO PROCESSADO 009 029". (Figura 17).

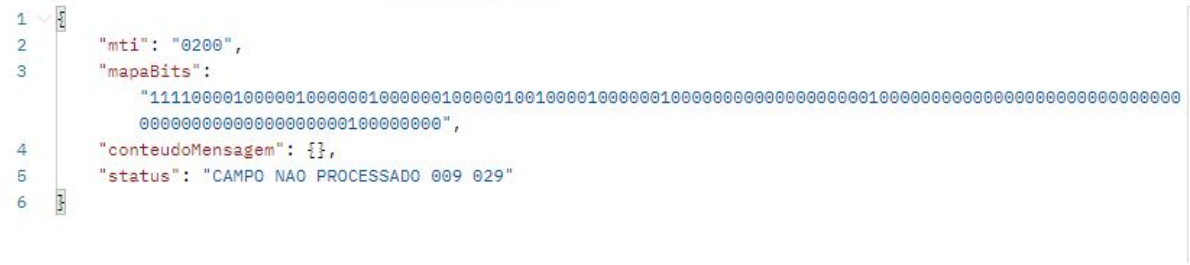

Figura 17: Mensagem contendo campos não processados

Fonte: Autoria própria

Voltando para mensagem com mapa de bits secundário, foi testado o status "TAMANHO INVALIDO", para isso basta adicionar qualquer valor indefinido, no nosso caso, na figura 18, foi adicionado 1 dígito ao número da conta (campo 2).

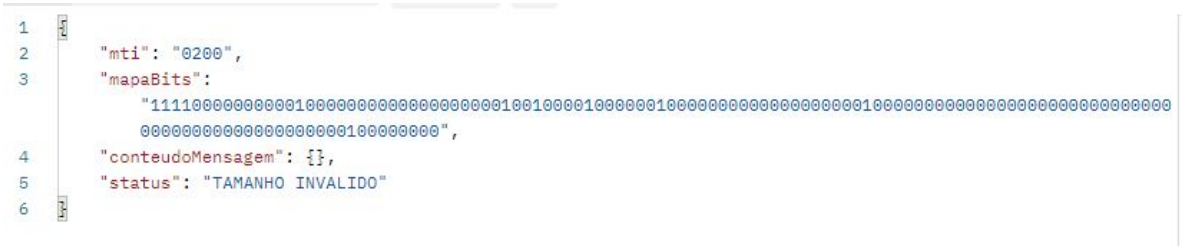

Figura 18: Mensagem com tamanho inválido

Fonte: Autoria própria

Novamente, com base na mensagem que possui mapa de bit secundários, foi verificado a validação dos campos, essa validação é mais específica pois só é executada uma vez que a mensagem conseguiu ser interpretada, ela pode ser demonstrada ao adicionar "AA"no numero da conta, como esse campo aceita apenas caracteres numéricos, obtivemos o resultada da figura 19.

```
{
    "mti": "0200",
    "mapaBits":
        "1111000000000010000000000000000000100100001000000100000000000000000100000000000000000000000000000000
        00000000000000000000100000000",
    "conteudoMensagem": {
        "002": "11AA222233334444",
        "003": "010000",
        "004": "00010000",
        "015": "2312",
        "035": "99999999",
        "038": "00",
        "043": "ESTABELECIMENTO TESTE",
        "050": "BRA",
        "070": "MARIO JOSE",
        "120": "1234567890123456789"
    },
    "status": "MENSAGEM COM TIPOS INVALIDOS -002"
}
```


Figura 19: Mensagem tipo invalido

Fonte: Autoria própria

Essa validação verifica todos os campos da mensagem que estejam definidos, caso exista campos com tipo inválido, a aplicação deve adiciona-los à lista de campos com falha, como podemos ver na figura. 20.


```
{
    "mti": "0200",
    "mapaBits":
        "1111000000000010000000000000000000100100001000000100000000000000000100000000000000000000000000000000
        00000000000000000000100000000",
    "conteudoMensagem": {
        "002": "11AA222233334444",
        "003": "010000",
        "004": "00010000",
        "015": "2312",
        "035": "99999999",
        "038": "00",
        "043": "ESTABELECIMENTO TESTE",
        "050": "BR1",
        "070": "M4RIO JOSE",
        "120": "1234567890123456789"
    },
    "status": "MENSAGEM COM TIPOS INVALIDOS -002 050 070"
}
```


Figura 20: Mensagem com lista de tipos inválidos

Fonte: Autoria própria

Após todos os possíveis retornos testados, podemos verificar o desempenho da API usando o Postman ao criar uma coleção de testes considerando os cenários executados anteriormente (tabela 3), mesmo com o aumento da quantidade de interações, a API não apresentou erros e seu tempo de resposta se manteve na média de 2ms. A duração total de cada teste é um pouco maior pois conta com os processamentos do Postman.

Tabela 3: Análise de desempenho da aplicação

Total de teste	Tempo Total	Tempo Reposta Médio
300	26s46ms	2ms
600	51s132ms	2ms
1080	1m 30s	2ms

4.3 Simulação de Evolução da definição

Após desenvolver uma aplicação funcional, é possível facilmente evoluir a definição da mensagem esperada na requisição. Por exemplo, em um cenário em que seja necessário a adição de novos campos de acordo com a tabela 4.

Tabela 4: Definição dos novos campos a serem adicionados

CAMPO	Tamanho	Tipo	Descrição
051	6	n	Horário da transação
054	2	n	Número de parcelas
065	LLVAR	an	Endereço do portador

Basta adicioná-los na classe enum("ListaCampos") criada anteriormente com a definição dos campos, como mostra a figura 21.

```
16    C051( campo: "051", tamanho: "6", tipo: "n", descricao: "HORARIO DA TRANSACAO"),
17    C054( campo: "054", tamanho: "2", tipo: "n", descricao: "NUMERO DE PARCELAS"),
18    C065( campo: "065", tamanho: "LLVAR", tipo: "an", descricao: "ENDERECO DO PORTADOR"),
```

Figura 21: Fragmento do código com definição dos novos campos na classe enum "ListaCampos"

Fonte: Autoria própria

Uma vez adicionados, é necessário reiniciar a aplicação, e em seguida já será possível enviar uma mensagem contendo os novos campos. Abaixo, é possível observar a mensagem de teste com os novos campos em negrito, e na figura 22 o resultado da requisição após adição dos novos campos na mensagem.

1. 0200F00200002420640084000000000001001111222233334444010000000100002312999999990021ESTABELECIMENTO TESTE03BRA
1630150319RUA LAGOA VERDE 29010MARIO JOSE0191234567890123456789

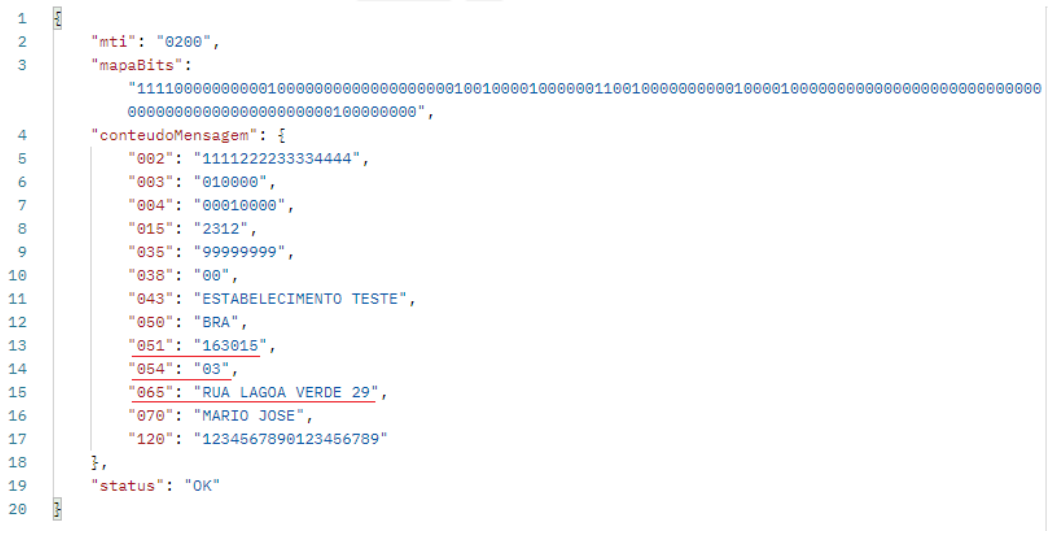

Figura 22: Resultado da mensagem com os novos campos adicionados

Fonte: Autoria própria

5 Considerações finais

Após as análises e desenvolvimento feitos no projeto, foi possível ter um entendimento de como funciona todo o percurso de uma mensagem de autorização financeira e os principais atores envolvidos no processo. Com a aplicação desenvolvida, foi possível visualizar como essas mensagens são interpretadas e como podemos aplicar inúmeras técnicas de processamento de dados e validações.

A aplicação desenvolvida usando uma definição genérica mostra como é possível facilmente adaptá-la, possibilitando testes com as mais diversas redes de pagamento. O mapeamento dos possíveis casos de erro com a atribuição de diferentes status torna a aplicação resiliente e confiável, simulando um cenário real de autorização, visto que, para diversas redes de pagamento, o tratamento de erros de formatação e validação dos campos da forma devida é essencial e em casos onde, por exemplo, a aplicação simplesmente "quebra" sem responder corretamente, pode ser gerada uma multa para o emissor.

O projeto também proporcionou o estudo de frameworks e ferramentas populares no mercado, como o Spring e o Postman. Usando o Java como linguagem de programação, temos uma documentação bem detalhada e uma larga comunidade para ajudar na solução de problemas e aprender mais sobre a linguagem. O Spring facilitou o processo de criar uma API de serviços web, uma vez que usamos requisições HTTP na nossa aplicação, ela pode ser facilmente conectada aos mais diversos sistemas.

Após a avaliação dos resultados da aplicação, podemos verificar que foi possível aplicar os conceitos do trabalho na simulação do que seria um ambiente real de interpretação de mensagens financeiras baseadas no principal padrão de mensagem, de maneira simples e com possibilidades de evolução da ferramenta.

5.1 Trabalhos futuros

O projeto abre portas para exploração de outros aspectos de um autorizador de transações financeiras, poderiam ser adicionadas novas funcionalidades e validações mais complexas. Um primeiro ponto de evolução da aplicação seria a conexão com um banco de dados, com acesso ao banco teríamos validação mais complexas, explorar o aspecto de persistência dos dados da transação e estudar qual modelo de banco de dados tornaria a aplicação mais performática.

Outro lado que poderia ser explorado são voltados para questões de segurança como os processos de validação de senha e do código de segurança do cartão, a aplicação poderia ser usada para simular uma transação com senha, assim explorando os fluxos de criptografia, como, geração de chave de criptografia e cálculos envolvidos na descriptografia da senha.

De forma geral, uma vez que conseguimos capturar e interpretar uma mensagem baseada no padrão ISO8583 como aqui apresentado no trabalho, é possível simular os mais diversos fluxos de validação de uma transação financeira de cartão de crédito.

REFERÊNCIAS

[1] Khalid Waleed Hussein1; Dr. Nor Fazlida Mohd. Sani; Professor Dr. Ramlan Mahmod; Dr. Mohd. Taufik Abdullah. Enhance Luhn Algorithm for Validation of Credit Cards Numbers . **International Journal of Computer Science and Mobile Computing**, 2, 7, 262-272, Julho, 2013.

[2] Alexandre Alves, André ; M. de S. Menezes, Octaviano. **Cartão de Crédito Private Label: a Arma de Crédito na Mão do Varejo**. São Paulo: Novatec Editora, 2007.

[3] WANG, Helena Yu Feng e IKEDA, Ana Akemi. Análise do mercado de cartão de crédito brasileiro. In: Anais São Paulo: USP/FEA/PPGA, 2004. Acesso em: 30 out. 2023.

[4] MAZZA JUNIOR, Miguel. **Estudo dos parâmetros para análise de viabilidade econômica para implementação de smart card em instituições financeiras**. Monografia Departamento acadêmico, Universidade Federal de Itajubá, Itajubá, 2004.

[5] LEÃO, L. B.; SOTTO, E. C. S. A EVOLUÇÃO DOS MEIOS DE PAGAMENTO.. **Revista Interface Tecnológica, [S. l.]**, v. 16, n. 1, 221-232, 219.

[6] PEREIRA, Lucas Carvalho; BITAR, Yasmin Adla da Costa. **Estratégias inteligentes de segurança para transação com cartão**. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) – Centro Universitário do Estado do Pará, Belém, 2018.

[7] HUSSEIN AHMAD AL-OFEISHAT; MOHAMMAD A.A.AL RABABAH. Near Field Communication (NFC) . **IJCSNS International Journal of Computer Science and Network Security**, v. 12, n. 2, Fevereiro, 2012.

[8] Fung, Ben; Huynh, Kim P.; Nield, Kerry; Welte, Angelika. Merchant acceptance of cash and credit cards at the point of sale. **Journal of Payments Strategy / Systems**, v. 12, n. 2, 150-165, 2018.

[9] Anuva Rani Saha ; Md. Mijanur Rahman. Automated Teller Machine Card Fraud of Financial Organizations in Bangladesh. **Journal of Computer Science Applications and Information Technology**, 3, 1, 1-6, 2018.

[10] ADRIANO MACHADO COSTA, Jackson. **SISTEMA DE ANÁLISE E MINERAÇÃO DE DADOS DE TRANSAÇÕES ELETRÔNICAS REGIMENTADAS PELA ISO8583**. Trabalho de Conclusão de Curso (Sistemas de informação) , FACCAT, Taquara - RS, 2011.

[11] GERMAN GIL MESSINESE, Luis. **Marco de desarrollo estándar basado en el protocolo ISO-8583 para terminales de venta**. Trabalho de Conclusão de Curso (Bacharel em Computação) , Universidad Central de Venezuela, Caracas, 2014.

[12] SETIAWAN BAHRUL ILHAM, Yanto. Implementation of High Availability Message ISO 8583 using F5 Active-Passive Failover Method **International Journal of Engineering Trends and Technology,** , vol. 71, no. 4, pp. 264-273, Abril, 2023.

[13] S. J. Murdoch, M. Bond and R. Anderson. How Certification Systems Fail: Lessons from the Ware Report **IEEE Security Privacy**, vol. 10, no. 6, pp. 40-44, Nov.-Dez, 2012.

[14] HEDEGAARD OLSEN,Anders; G. H. PEDERSEN, Allan. **Security in POS Systems**. Tese de Mestrado – Department of Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2005.

[15] FRISBY, WesLee; MOENCH, Benjamin; RECHT, Benjamin; RISTENPART, Thomas. Security Analysis of Smartphone Point-of-Sale Systems . In: Proceedings of the 6th USENIX conference on Offensive Technologies, 2012, Bellevue, WA. Acesso em: 31 out. 2023.

[16] IBM Integration Bus. Disponível em: <https://www.ibm.com/docs/pt-br/integration-bus/10.0?topic=formats-iso8583-messaging-standard>Acesso em: 07/11/2023.

[17] GALLAHER, Mike; HARPER, Chad . Demystifying ISO 20022:Evaluating the benefits and limitations of new messaging standards. **Payments Strategy and Systems, Henry Stewart Publication**, vol. 15, 4, 410-418, Dezembro, 2022.

[18] Documentação Spring Initializr. Disponível em: <https://docs.spring.io/initializr/docs/current/reference/html/create-instance>. Acesso em: 07/11/2023.

[19] Visa Secure with EMV 3-D Secure Authentication. Disponível em: <https://km.visamiddleeast.com/en/_KM/run-your-business/small-business-tools/payment-technology/visa-secure.htmlMerchant_benefits_7a6>Acesso em: 07/11/2023.

[20] Chih-Hsiung Su ; Fengjun Tu2 ; Xinyu Zhang ; Ben-Chang Shia ; Tian-Shyug Lee. A ENSEMBLE MACHINE LEARNING BASED SYSTEM FOR MERCHANT CREDIT RISK DETECTION IN MERCHANT MCC MISUSE **Journal of Data Science** , vol. 17, no. 1, pp. 81-106, 2019.