UNIVERSIDADE FEDERAL FLUMINENSE

INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

DANIEL PEREGRINO DE MOURA CAVALCANTE

DESENVOLVIMENTO DE UM GATEWAY CONVERSOR HTTP/ISO 8583

RIO DE JANEIRO
2010

DANIEL PEREGRINO DE MOURA CAVALCANTE

DESENVOLVIMENTO DE UM GATEWAY CONVERSOR HTTP/ISO 8583

Dissertação apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense como requisito para obtenção da Graduação.

Orientadora: Profª. Anna Dolejsi Santos

Rio de Janeiro

2010

Universidade Federal Fluminense
Daniel Peregrino de Moura Cavalcante

DESENVOLVIMENTO DE UM GATEWAY CONVERSOR HTTP/ISO 8583

Monografia apresentada como pré-requisito para o obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal Fluminense, submetida a aprovação da banca examinadora composta pelos seguintes membros:

_____

Professora Orientadora: Anna Dolejsi Santos

_____

Professor: Leonardo Cruz da Costa

_____

Professor: Luiz Valter Brand Gomes

Niterói
2010

Resumo

Permitir a troca de mensagens entre aplicações que utilizam protocolos de comunicação diferentes é a tarefa dos chamados *Gateways*. Na telefonia celular os *Gateways* permitem a comunicação entre SMSC's que utilizam protocolos de comunicação diferentes. Na Internet, os *Gateways* permitem que sites de *e-commerce* se comuniquem com sistemas de vários bancos, sem terem que se preocupar com as particularidades dos protocolos de comunicação utilizados pelos sistemas de cada banco.

Nesse trabalho desenvolvemos um *Gateway* que converte mensagens HTTP para mensagens ISO 8583. Desenvolvemos também uma aplicação web, que permite a configuração do Gateway e das regras de roteamento e de conversão das mensagens.

Posteriormente, realizamos experimentos para avaliar a capacidade do *Gateway* de tratar requisições HTTP simultâneas. Para os experimentos foram desenvolvidos duas aplicações, sendo uma responsável por enviar requisições HTTP e outra responsável por receber as mensagens ISO 8583.

Palavras-chave: Gateway, ISO 8583, HTTP, POS, MTI

Sumário

Lista de Abreviaturas

ANS – Agência Nacional de Saúde

ASCII – *American Standard Code for Information Interch*

GPRS – *General Packet Radio Service*

HTTP – *Hypertext Transfer Protocol*

IIC – *Institution Identification Codes*

IP – *Internet Protocol*

IoC – *Inversion of Control*

ISO – *International Organization for Standardization*

J2ME – *Java2 Micro Edition*

JSF – JavaServer Faces

MIME – *Multipurpose Internet Mail Extensions*

MTI – Message Type Identifier

NSU – Número Sequencial Único

POS – *Point of Sale*

SMS – *Short Message Service*

SMSC – *Short Message Service Center*

TCP – *Transmission Control Protocol*

TISS – Troca de Informação em Saúde Suplementar

URL – *Uniform Resource Locator*

WAR – *Web Application Archive*

1 Introdução

Aplicações cujo objetivo principal seja integrar sistemas que se comunicam através de protocolos diferentes são conhecidas como *Gateways*. Basicamente os *Gateways* agem como conversores e roteadores, integrando sistemas que por não utilizarem o mesmo protocolo de comunicação não podem trocar mensagens entre si. Quando um *Gateway* recebe uma mensagem ele converte a mensagem para o formato esperado por quem vai receber a mensagem e faz o roteamento para o destinatário. Em seguida recebe uma resposta do destinatário, converte para o formato esperado pelo remetente e devolve a resposta. Usualmente, na mensagem do remetente para o *Gateway*, podem haver informações sobre o destinatário, como o protocolo de comunicação utilizado por ele e o endereço de destino.

Um exemplo de *Gateway* são os chamados *Gateways* SMS (*Short Message Service)*. Quando um usuário de telefone celular envia um SMS, a mensagem é armazenada em um SMSC (*Short Message Service Center*) e depois é enviada para o destinatário. O problema surge do fato que as empresas usam protocolos diferentes e até mesmo proprietários para se comunicarem com os SMSCs. Dessa forma dois SMSCs que utilizam protocolos diferentes não podem se comunicar entre si. O *Gateway* SMS surge então para permitir a comunicação entre SMSCs que utilizam protocolos diferentes para se comunicarem [1]. Exemplos de *Gateways* SMS são SMSxchange [19] e o TM4B Bulk SMS *Gateway* [20].

Outro tipo de *Gateway*, é o *Gateway* de Pagamento (*Payment Gateway*). Esse tipo de *Gateway* pode ser visto como um simples conector. Ele conecta os web sites de lojas virtuais com as redes de bancos e entidades financeiras que fazem o processamento dos pagamentos dos clientes. O *Gateway* de Pagamento não processa as transações, ele simplesmente repassa as informações para os sistemas que podem aprovar os pagamentos.

O *Gateway* recebe os dados da compra do web site e converte para o formato esperado pelas aplicações que podem processar e validar os dados [2]. Alguns exemplos de *Gateways* existentes são *Payflow Payment Gateway* [21], Authorize.net [22].

1.1 Objetivos

O *Gateway* desenvolvido nesse trabalho tem o objetivo de integrar sistemas que se comunicam utilizando o protocolo HTTP (*Hypertext Transfer Protocol*) [3] com sistemas que utilizam para se comunicar o protocolo ISO 8583. Ou seja, o *Gateway* desenvolvido converte mensagens HTTP para mensagens ISO 8583 [4]. O padrão ISO 8583, *Financial Transaction Card Originated Messages*, é um padrão da *International Organization for Standardization* para sistemas que trocam mensagens em transações financeiras. Esse padrão é abordado no Capítulo 2.

Uma aplicação prática para o projeto é promover uma maneira simples de aplicações desenvolvidas em J2ME (Java 2, *Micro Edition*) [18] se comunicarem com outras aplicações que se comunicam utilizando ISO 8583. Isso porque, J2ME tem suporte a HTTP, mas não para ISO 8583. J2ME é a versão do Java para desenvolvimento de aplicações para telefones celulares. No entanto, o *Gateway* desenvolvido nesse trabalho pode ser útil a qualquer aplicação que deseje utilizar o protocolo HTTP para se comunicar com aplicações que se comunicam utilizando ISO 8583.

O trabalho de implementação está divido em 2 sistemas: *Gateway Manager* e *Gateway* Conversor.

O *Gateway Manager* é um sistema web onde os usuários podem se cadastrar e configurar os parâmetros de conversão e de roteamento de mensagens (por exemplo, IP e Porta para onde as mensagens serão roteadas). Além disso, é possível configurar os parâmetros de funcionamento do *Gateway* Conversor, como a porta em que ele vai escutar, por exemplo.

O *Gateway* Conversor faz a conversão da mensagem HTTP para ISO 8583 e o roteamento para o autorizador configurando anteriormente no *Gateway Manager*.

Para auxiliar os testes do *Gateway* Conversor foram desenvolvidos outros 2 programas: Disparador HTTP e Rebatedor ISO 8583.

O Disparador HTTP é um aplicativo de linha de comando utilizado para enviar mensagens HTTP para o *Gateway* Conversor. Os parâmetros da requisição HTTP são configurados através de um arquivo de propriedades. Esse programa permite a configuração do número de transações enviadas por segundo. Dessa forma, podemos colocar para rodar várias instâncias do Disparador HTTP ao mesmo tempo e com isso simular diversos clientes enviando requisições simultâneas para o *Gateway* Conversor. Isso nos permite realizar testes de escalabilidade do *Gateway* Conversor.

O Rebatedor ISO 8583 recebe as mensagens do *Gateway* Conversor depois delas terem sido convertidas para o formato ISO 8583. Assim como o Disparador HTTP, a configuração do Rebatedor ISO 8583 também é feita por arquivo de propriedades.

1.2 Organização do trabalho

O trabalho está organizado em 7 capítulos. No Capítulo 2 é feita uma abordagem sobre os protocolos de comunicação HTTP e ISO8583. O Capítulo 3 aborda as ferramentas utilizadas no desenvolvimento, como *frameworks*, API's, SGBD e servidor de aplicações. O Capítulo 4 descreve as aplicações desenvolvidas. São elas: Disparador HTTP, Rebatedor ISO 8583, *Gateway* Conversor e *Gateway Manager*. O Capítulo 5 descreve os experimentos realizados e os resultados dos testes. Finalmente, no Capítulo 6, são apresentadas as conclusões e os possíveis trabalhos futuros.

2 Protocolos de Comunicação

A transferência de dados através da Internet depende de um conjunto de protocolos. As troca de mensagens e as ações tomadas quando mensagens são enviadas e recebidas são os elementos chave para se definir um protocolo.

Um protocolo define tanto a sintaxe quanto a semântica das mensagens trocadas entre os emissores e os receptores. Ou seja, um protocolo é uma linguagem com uma gramática, uma estrutura sintática como formatos de mensagens e um conjunto de regras semânticas que indicam como os campos das mensagens devem ser interpretados [3]. Dessa forma, um protocolo define o formato e a ordem das mensagens trocadas entre as entidades que se comunicam através de uma rede. E define também as ações que devem ser tomadas durante a transmissão e o recebimento de mensagens [23].

Como dito anteriormente o objetivo principal do projeto é desenvolver um software que permita a comunicação entre sistemas que utilizem diferentes protocolos de comunicação. Os protocolos de comunicação envolvidos são: HTTP (*Hypertext Transfer Protocol*) [3] e ISO 8583 [4] .

Nesse capítulo é dada uma breve visão sobre o HTTP e uma abordagem um pouco mais profunda sobre o padrão ISO 8583. O motivo para uma abordagem mais detalhada sobre ISO 8583 é que esse protocolo é menos conhecido e sua utilização não é tão ampla como o HTTP, que hoje é o protocolo mais utilizado na grande rede.

2.1 HTTP

O HTTP é um protocolo da camada de aplicação amplamente utilizado na Web, sendo a forma mais comum de transferir recursos através da rede. Ele define o formato e o significado das mensagens trocadas entre os componentes na Web, como clientes e servidores [3].

O HTTP define entre outras coisas, a sintaxe das mensagens e como os campos de cada linha da mensagem devem ser interpretados. Sendo um protocolo do tipo *request-response*, onde o cliente envia mensagens de requisição e o servidor envia respostas para as requisições. Além disso, o HTTP é um protocolo do tipo *stateless*, o que significa que clientes e servidores tratam cada mensagem trocada independentemente, não mantendo qualquer estado através das requisições e respostas.

Na Internet, a comunicação via HTTP é feita sobre conexões TCP/IP, mas isso não é uma restrição, ou seja, outros protocolos das camadas de transporte e rede podem ser utilizadas por pilhas de protocolos não Internet.

As mensagens são as unidades fundamentais para a comunicação no HTTP. Uma mensagem pode ser uma requisição enviada por um cliente ou uma resposta enviada por um servidor.

Uma mensagem de requisição é composta por uma *Request Line* que indica o método e o recurso requisitado, linhas de cabeçalho de propósito geral que podem informar ao servidor dados do cliente, como o navegador de Internet utilizado e um corpo, que pode conter qualquer informação.

Uma mensagem de resposta começa com uma *Status Line*, que inclui a versão do HTTP utilizada pelo servidor e o código de status da resposta. Em seguida vem as linhas de cabeçalho que podem trazer informações da mensagem, como o tamanho do conteúdo e por fim o corpo da mensagem que pode conter qualquer informação.

2.2 ISO 8583

O padrão ISO 8583, *Financial Transaction Card Originated Messages*, é um padrão da *International Organization for Standardization* para sistemas que trocam mensagens em transações financeiras. Esse padrão é divido em 3 partes:

Parte 1: *Messages, data elements and code values*. [4]

Parte 2: *Application and registration procedures for Institution Identification Codes (IIC)* [5]

Parte 3: *Maintenance procedures for messages, data elements and code values* [6]

A parte 1 especifica o formato das mensagens trocadas entre as instituições envolvidas.

A parte 2 descreve o procedimento de candidatura e matrícula dos Códigos de Identificação da Instituição (IIC). O padrão estabelece um sistema de numeração de códigos para identificação das instituições.

A parte 3 define o papel da Agência de Manutenção e determina os procedimentos para a inclusão de mensagens e elementos de dados. A responsabilidade da Agência de Manutenção é padronizar tipos de identificadores de mensagens, classes de mensagens e elementos de dados e sub-elementos.

Basicamente existem duas classes de instituições: Adquirentes e Emissores. Adquirentes são as instituições financeiras, ou não, responsáveis por capturar os dados relativos a transação (dados de um cartão de crédito, por exemplo) e enviar para os emissores. Exemplos de adquirentes são Cielo e RedeCard. Emissores são instituições que emitem os cartões e que conhecem os dados dos cartões, e por isso são as responsáveis por aprovar ou não uma transação. Bancos que emitem cartões de débito ou de crédito para seus clientes são exemplos de emissores.

Para este trabalho a parte relevante da padrão ISO 8583 é a parte 1. Pois para o desenvolvimento do *Gateway* Conversor o importante é conhecer e saber trabalhar com mensagens no formato especificado pelo padrão.

Uma mensagem ISO 8583 é composta pelas seguintes partes:

- Identificador de Tipo de Mensagem(MTI)
- Um ou mais *bitmaps*
- Elementos de dados e sub-elementos

O Identificador de Tipo da Mensagem ou MTI (*Message Type Identifier*) especifica a função da mensagem. Esse campo é composto por 4 dígitos, onde cada dígito representa uma informação distinta: o primeiro dígito, mostrado na Tabela 2.2.1, indica a versão da ISO 8583; o segundo dígito, mostrado na Tabela 2.2.2, indica a classe da mensagem;o terceiro dígito, mostrado na Tabela 2.2.3, indica se é uma mensagem de requisição ou se é uma resposta; e o quarto dígito, mostrado na Tabela 2.2.4, indica quem iniciou a troca de mensagens, o adquirente ou o emissor.

Posição	Versão
0xxx	ISO 8583-1:1987
1xxx	ISO 8583-2:1993
2xxx	ISO 8583-1:2003
9xxx	Uso privado

Tabela 2.2.1 - Versões da ISO 8583

Posição	Classe
x1xx	Mensagem de Autorização
x2xx	Mensagem Financeira
x3xx	Mensagem de Ação de Arquivo
x4xx	Mensagem de Reversão
x5xx	Mensagem de Conciliação
x6xx	Mensagem Administrativa
x7xx	Mensagem de Taxa de Coleta
x8xx	Mensagem de Gerenciamento de Rede
x9xx	Reservada pela ISO

Tabela 2.2.2: Classes de Mensagens

A classe da mensagem indica o propósito da mensagem. A Tabela 2.2.2 mostra as possíveis classes de mensagem.

Mensagem de Autorização verifica se o cliente tem saldo para a transação que está está sendo feita, mas não efetiva a transação. É como se fosse uma consulta de saldo feita pelo adquirente, mas transparente para o usuário.

Mensagem Financeira é aquela que efetiva a transação no emissor, por exemplo, debita o valor da transação na conta do cliente de um banco.

Mensagens de Ação de Arquivo são mensagens enviadas quando se utiliza *hot-cards*. *Hot-cards* são os chamados cartões de vantagens e de privilégios. Por exemplo um cartão emitido por uma companhia aérea que permita o acesso à uma área VIP de um aeroporto é um *Hot-card*.

Mensagem de Reversão é a mensagem utilizada para reverter uma autorização feita previamente. Ela deve ser enviada no caso de ocorrer algum problema de comunicação entre o sistema do adquirente e o do emissor. Por exemplo, suponha que o adquirente envie uma Mensagem Financeira para emissor e essa mensagem chegue corretamente no emissor, que irá debitar o valor do cliente. Porém, no caminho de volta da mensagem ocorre um problema na rede e o sistema do adquirente não recebe a resposta do sistema do emissor. Nesse caso o sistema do adquirente não tem como saber se a mensagem chegou ou não no sistema do emissor. E então o adquirente envia uma Mensagem de Reversão para reverter o débito na conta do cliente.

Mensagem de Conciliação é uma mensagem enviada normalmente no final do dia, onde o sistema do aquirente envia uma espécie de relatório com todas as transações aprovadas desde a última Mensagem de Conciliação. Essa mensagem serve para o sincronizar os sistemas e garantir que todas as transações que foram aprovadas pelo emissor foram devidas.

Mensagem Administrativa é a mensagem utilizada para fins administrativos, normalmente para indicar alguma falha no sistema do adquirente.

Mensagem de Taxa de Coleta é uma mensagem de cobrança do adquirente por uma determinada transação efetuada.

Mensagens de Gerenciamento de Rede são mensagens utilizadas para diversos fins relacionados à rede. Por exemplo, testes de comunicação e troca de chaves de segurança.

O terceiro dígito do MTI indica a função da mensagem.

Posição	Função
xx0x	Request
xx1x	Request Response
xx2x	Advice
xx3x	Advice Response
xx4x	Notification
xx8x	Response acknowledgment
xx9x	Negative acknowledgment

Tabela 2.2.3 – Funções de Mensagem

A função da mensagem indica o propósito da mensagem no autorizador. O uso dessas funções pode variar entre diferentes sistemas. Usualmente, *requests* são usados para transações que necessitam de alguma autorização, uma transação de venda, por exemplo. Ou ainda uma transação de consulta, como por exemplo, uma consulta de sado de uma conta corrente.

Advice é a mensagem enviada para o emissor quando este não tem autonomia para autorizar ou não uma transação. Ou seja, o emissor é somente informado que uma transação foi realizada. Isso acontece normalmente quando uma transação já foi aprovada por uma outra instituição ou no caso de cartões com chip que podem ter algum tipo de processamento que permita a aprovação de transações no próprio chip.

Notificações podem ser usadas para enviar quaisquer tipos de notificações para o servidor. Por exemplo, uma aplicação rodando em um POS (*Point of Sale* – local aonde a transação é originada) que utilize uma rede GPRS (*General Packet Radio Service)* para se comunicar com o autorizador pode ficar um longo período de tempo sem acesso a nenhuma rede disponível. Quando a rede volta, a aplicação pode enviar uma mensagem para o servidor informando esse fato, para que um grupo de técnicos vá verificar o que está ocorrendo na rede GPRS da região.

O quarto e último dígito do MTI, mostrado na Tabela 2.4, indica quem originou a mensagem.

Posição	Origem
xxx0	Adquirente
xxx1	Adquirente Repeat
xxx2	Emissor
xxx3	Emissor Repeat
xxx4	Outro
xxx5	Outro Repeat

Tabela 2.2.4 – Origem da Mensagem

Na Tabela 2.4, *Repeat* significa que a mensagem que está sendo enviada já foi enviada antes, porém não houve resposta e nesse caso quem está originando a mensagem está reenviando.

A segunda parte da mensagem ISO 8583 é composta pelo *Bitmap* ou Mapa de Bits. Esse é o campo na mensagem que informa quais os campos (elementos de dados) estão presentes na mensagem e quais não estão. A mensagem pode ter até 2 bitmaps e cada *bitmap* é composto de 16 dígitos em hexadecimal onde cada dígito representa 4 campos da mensagem. O primeiro bit do primeiro *bitmap* informa sobre a presença do segundo *bitmap*. O número máximo de campos que uma mensagem ISO 8583 pode ter é de 127 campos.

A terceira parte da mensagem ISO 8583 é composta pelos elementos de dados e sub-elementos. Esses são os campos da mensagem onde os dados da transação trafegam. Esses campos carregam as mais diversas informações, e cada aplicação tem suas particularidades. Alguns exemplos das informações enviadas nesses campos são: data e hora da transação, valor da transação e NSU (Número Sequencial Único) da transação.

2.3 Empacotadores

Na prática uma mensagem ISO 8583 é uma sequência de bytes e para que essa sequência de bytes possa ser interpretada são utilizados os chamados empacotadores.

Um empacotador define quais campos podem estar presentes e como esses campos estão formatados na mensagem. Os empacotadores são necessários porque existem diferentes tipos de campos, e o tipo de cada campo não é enviado na mensagem.

A parte 1 da especificação da ISO 8583 define o tipo de alguns campos, porém, o padrão não obriga a seguir essa especificação, dessa forma todos os campos precisam ser informados nos empacotadores, mesmo os que têm seus tipos pré-definidos.

Os campos podem estar em 2 formatos: binário ou ASCII. Além disso, os campos podem ser numéricos ou alfanuméricos. Com relação ao tamanho dos campos, eles podem ser especificados no empacotador para terem um tamanho fixo ou variável. No caso de tamanho variável, é necessário especificar no empacotador quantos dígitos compõem o tamanho do campo.

1100202005000080000090000100001706580014003132373030303030

MTI: 1100

Bitmap em Hexadecimal: 2020050000800000

Bitmap em binário:
0010000000100000000001010000000000000001000000000000000000000000

Campo 3: 900001

Campo 11: 000017

Campo 22: 06580014 (06 é o tamanho do campo, ver Tabela 2.3.1)

Campo 24: 0031

Campo 41: 3237303030303030

Figura 2.2.1 – Exemplo de Mensagem ISO 8583

Na Figura 2.2.1 mostramos um exemplo de uma mensagem ISO 8583 e mostramos os campos da mensagem. A mensagem em si é a sequência de dígitos:

1100202005000080000090000100001706580014003132373030303030

Porém, aproveitamos a figura para mostrar como os campos na mensagem. Como pode ser visto na Figura 2.2.1 a primeira parte da mensagem é o MTI, que nesse caso é 1100.

A seguir vem o *bitmap*, que está em hexadecimal e para facilitar o entendimento mostramos em binário. Analisando o *bitmap* descobrimos que a mensagem possui somente um *bitmap*. Isso porque o primeiro bit do *bitmap* é 0. Se fosse 1 saberíamos que os 16 dígitos seguintes ao primeiro *bitmap* seriam um outro *bitmap*. Ainda analisando o *bitmap* vemos que a mensagem possui os campos 3, 11, 22, 24 e 41.

Agora que sabemos os campos presentes na mensagem precisamos saber o formato de cada campo. Como dito anteriormente essa informação não vem na mensagem. Porém, analisando o empacotador mostrado na Tabela 2.3.1 somos capazes de interpretar os campos na mensagem. Vale ressaltar que um empacotador pode definir campos que não aparecem na mensagem. A Tabela 2.3.1 mostra só um pedaço de um empacotador. Para simplificar mostramos nessa tabela somente os campos que aparecem na mensagem.

Os campos começam após o *bitmap* e o primeiro campo é o 3. Pelo empacotador mostrado na Tabela 2.3.1 esse campo é numérico e tem tamanho fixo em 6. Dessa forma os 6 primeiros dígitos após o *bitmap* são do campo 3. Da mesma forma temos o campo 11 após o campo 3.

No campo 22 muda um pouco, pois é um campo de tamanho variável. Na Tabela 2.3.1 vemos que para informar o tamanho do campo 22 são usados 2 dígitos. Portanto, os 2 dígitos após o campo 11 representam o tamanho do campo 22 na mensagem, que no caso é 06. Logo, os 6 dígitos a seguir são do campo 22.

O campo 24 segue a mesma ideia dos campos 3 e 11.

O campo 41 é um texto em ASCII, mas está em hexadecimal na mensagem. Para sabermos o valor do campo devemos utilizar a tabela ASCII [24] para obtermos o valor real

do campo. Fazendo a conversão de hexadecimal para caracteres temos que o valor do campo 41 é 27000000.

Campo (Bit)	Tipo	Tamanho	Pad
3	Numérico	Fixo em 6	Sim
11	Numérico	Fixo em 6	Sim
22	Numérico	Variável com máximo em 16 e 2 dígitos para representar o tamanho	Não
24	Numérico	Fixo em 4	Sim
41	Alfa	Fixo em 8	Sim

Tabela 2.3.1 – Exemplo de Trecho de um empacotador

4 Ferramentas

No desenvolvimento das aplicações foram utilizados alguns *framewoks* com o objetivo de otimizar o desenvolvimento e tornar o código mais limpo. A linguagem utilizada no desenvolvimento foi Java [15] e a IDE utilizada foi o Netbeans [16].

Nesse capítulo, faremos uma abordagem dos *framewoks* utilizados, buscando dar ênfase no motivo pelo qual o *framework* foi utilizado e quais foram os benefícios na sua utilização.

4.1 JavaServer Faces

JavaServer Faces é o *framework* para desenvolvimento de aplicações web baseadas em Java. Esse *framework* foi desenvolvido pela empresa criadora do Java: Sun Microsystems. A primeira versão do *framework* foi lançada em 2004. A versão mais recente é a 2.0 e foi lançada no final de 2009. No desenvolvimento do *Gateway Manager* foi utilizada a versão a versão 1.2. Essa versão foi utilizada, e não a mais nova, pelo fato do desenvolvedor da aplicação já ter experiência com essa versão.

Utilizando um conceito baseado em componentes, o JavaServer Faces foi desenvolvido para simplificar o desenvolvimento de aplicações web. Os principais componentes do *framework* são: API para representação e gerenciamento de componentes de interface com o usuário (*tags*), definição da navegação pelas páginas do sistema, captura de eventos (clique em um *link* ou em um botão, por exemplo), conversão e validação de dados do lado do servidor (*server-side*), suporte a acessibilidade e a internacionalização, e por fim extensibilidade, que é a capacidade de adição de novos componentes desenvolvidos por usuários do *framework* [7].

A configuração do JavaServer Faces é feita no arquivo web.xml (*deployment descriptor)* da aplicação e em um arquivo especifico do *framework* chamado faces-config.xml. No web.xml registra-se o Faces Servlet e a URL que será mapeada para esse

servlet. E no arquivo faces-config.xml registam-se os componentes da aplicação que devem ser tratados pelo JSF e os fluxos de navegação entre as páginas da aplicação.

A utilização do JavaServer Faces permitiu o desenvolvimento do *Gateway Manager* de maneira organizada através de componentes e fluxos de navegação descritos no arquivo de configuração.

4.2 Facelets

Facelets é um *framework* que permite a utilização de *templates* no desenvolvimento de aplicações web baseadas em JavaServer Faces. A utilização de *templates* permite a reutilização de *layouts* no desenvolvimento de páginas web [8].

No desenvolvimento do *Gateway Manager* foi criado um arquivo de *template* onde estão definidos o cabeçalho, o menu e o rodapé. Todas as páginas do sistema utilizam esse *template*, o que permite reutilização de código e torna mais fácil a manutenção. Por exemplo: se houver necessidade de inclusão de um item no menu, apenas um arquivo deverá ser editado e a alteração irá se refletir em todas as outras páginas.

4.3 Spring

O Spring é um *framework* criado para tentar diminuir a complexidade no desenvolvimento de aplicações baseadas em Java, principalmente aplicações web. O núcleo do Spring *Framework* é baseado no princípio da Inversão de Controle (IoC). Aplicações que seguem o princípio da IoC utilizam configurações que descrevem as dependências entre os seus componentes. Cabe então ao *framework* de IoC satisfazer as dependências que foram configuradas. A 'inversão' significa que a aplicação não tem controle sobre sua estrutura, é responsabilidade do *framework* realizar essa tarefa [9].

A implementação da Injeção de Dependência do Spring coloca o foco no acoplamento fraco:os componentes do aplicativo devem conhecer o mínimo possível sobre outros componentes. A maneira mais fácil de alcançar acoplamento fraco em Java é desenvolver código baseado em interfaces. O conceito importante é que cada componente não sabe qual a implementação concreta que está usando, ele vê apenas uma interface. Como cada componente da aplicação é ciente apenas das interfaces dos outros componentes, podemos mudar a implementação dos componentes (ou grupos inteiros ou camadas de componentes), sem afetar os componentes que usam os componentes alterados. O núcleo de Injeção de Dependência do Spring usa as informações de um arquivo de configuração satisfazer as dependências entre os seus componentes.

O *framework* é composto de vários módulos, porém, no desenvolvimento do *Gateway Manager* foi utilizado somente a Injeção de Dependência e o módulo de controle de transações integrado ao Hibernate. O controle de transações promove uma maneira robusta da aplicação fazer operações que acessem o banco de dados, pois o Spring faz o gerenciamento das transações, incluindo *commits* e *rollbacks*.

4.4 Hibernate

O Hibernate é um *framework* desenvolvido para facilitar o desenvolvimento de aplicações em Java que fazem acesso a um banco de dados relacional. Atualmente já existe versão desse *framework* desenvolvida para .NET, chamada NHibernate. Basicamente o Hibernate permite que objetos Java sejam mapeados diretamente para o banco de dados.

Além disso, a configuração da conexão com o banco de dados é feita através de um arquivo de configuração, o que permite que os dados de conexão com o banco de dados sejam alterados sem que o código precise ser recompilado. E por dar suporte a vários SGBD's, utilizando o Hibernate é possível migrar uma aplicação de um SGBD para outro sem haver necessidade de alterações no código da aplicação [10].

4.5 Log4j

O Log4J é uma API desenvolvida pela *Apache Software Foundation* que torna mais fácil e mais limpa, em termos de linhas de código, a tarefa de logar os eventos que ocorrem no código. A configuração do Log4J é feita através de um arquivo de configuração, o que permite que o nível de log e as informações logadas sejam alteradas sem a necessidade de recompilar o código [11].

A utilização do Log4J no projeto permite monitorar o *Gateway* Conversor, pois todas as mensagens recebidas e envidas são gravadas em um arquivo de log.

4.6 JPOS

O JPOS é um *framework* desenvolvido em Java que permite o desenvolvimento de aplicações que utilizem o padrão ISO 8583 de uma maneira simples, pois possui componentes que abstraem a maior parte da complexidade do processamento de mensagens ISO 8583. Possui também componentes que auxiliam na tarefa de enviar e receber mensagens [12].

4.7 MySql

O SGBD (Sistema Gerenciador de Banco de Dados) utilizado no projeto foi o MySql. Não há nenhuma razão especial para ele ter sido escolhido, já que o requisito principal era utilizar um SGBD gratuito. Outras alternativas foram cogitadas, mas o MySql foi escolhido pelo simples fato do desenvolvedor desse trabalho ter mais experiência com este do que com outros [13].

4.8 Tomcat

O Tomcat é um *servlet container* de aplicações web desenvolvidas em Java, que utilizem as tecnologias Servlet e JavaServer Pages. Desenvolvido pela Apache Software Foundation, o Tomcat funciona como um servidor, onde pode-se fazer o *deploy* de aplicações web empacotadas em um *war* (Web Application Archive) [14].

O *Gateway Manager*, que é uma aplicação web, sendo assim, ela deve ser empacotada em um *war* que é colocado no Tomcat para que este disponibilize o acesso através da Internet.

5 Aplicação e Desenvolvimento

Neste capítulo as 4 aplicações desenvolvidas no projeto serão descritas. Duas delas, Disparador HTTP e Rebatedor ISO 8583, foram desenvolvidas com o único intuito de auxiliar nos testes. O resultado efetivo do trabalho de desenvolvimento são as aplicações *Gateway Manager* e *Gateway* Conversor.

5.1 Gateway Manager

O *Gateway Manager* é um sistema web, e portanto deve ser acessado utilizando-se um navegador de Internet, como Internet Explorer, por exemplo. Esse sistema provê uma interface onde é possível configurar os parâmetros de funcionamento do *Gateway* Conversor e as regras de conversão e roteamento de mensagens.

Existem 2 perfis de usuários no *Gateway Manager*, são eles: administrador e usuário comum. O perfil define o que o usuário pode fazer ou não no sistema e quais páginas ele pode acessar.

Funções do Administrador:

- Cadastrar Clientes
- Configurar parâmetros do *Gateway* Conversor
- Gerenciar empacotadores
  - Incluir
  - Editar
  - Exluir

Funções do Cliente:

- Alterar dados cadastrais
- Cadastro de Mensagens
- Configurar roteamento HTTP/ISO 8583

5.1.1 Login no sistema

Para acessar o *Gateway Manager* o usuário precisa fazer o login no sistema, digitando seu login e senha. A Figura 5.1.1 mostra a tela de login. Caso tenha esquecido a senha o usuário pode clicar no link *esqueci minha senha.* Nesse caso ele será direcionado para uma tela onde deve digitar o e-mail cadastrado no sistema para que uma nova senha seja enviada para o seu e-mail.

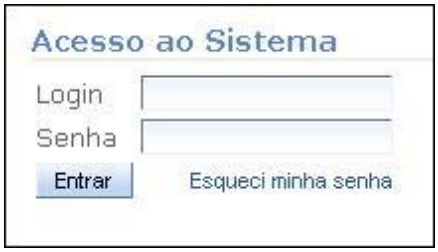

Figura 5.1.1 – Tela de login

5.1.2 Gerenciamento de Usuários

O gerenciamento de usuários é permitido somente para usuários do perfil administrador e inclui as funções de cadastro, listagem, detalhamento e exclusão de usuários.

No cadastro o usuário administrador deve digitar os dados conforme a Figura 5.1.2.1

Na listagem de usuários os usuários são exibidos em uma tabela conforme a Figura 5.1.2.2. Através da listagem de usuários é possível acessar informações de usuários e excluir usuários.

As informações de um usuário são acessadas através do link *Detalhes* e exibe informações do usuário conforme a Figura 5.1.2.3.

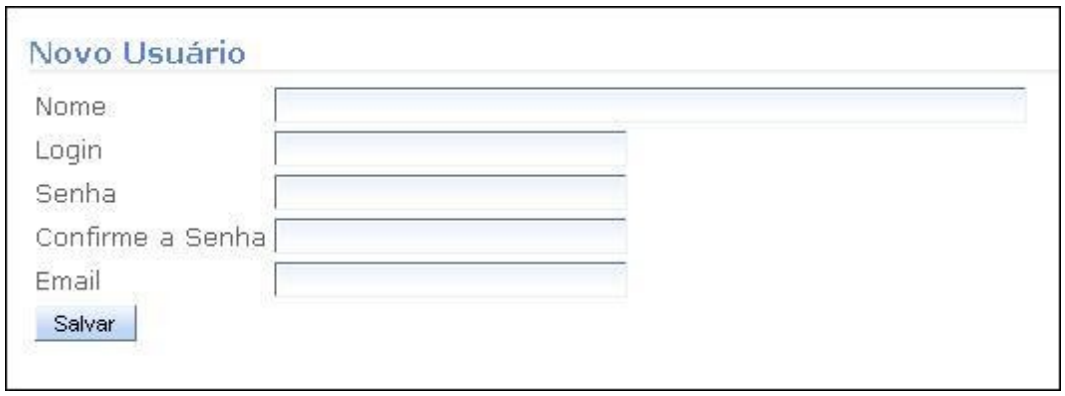

Figura 5.1.2.1 – Cadastro de Usuários

Figura 5.1.2.2 – Lista de Usuários

Figura 5.1.2.3 – Detalhes de Usuário

5.1.3 Configuração dos Parâmetros do Gateway Conversor

A configuração dos parâmetros do *Gateway* Conversor também só é permitida para usuários administradores. Os parâmetros configurados são: porta, tamanho inicial e máximo do *pool*, tempo de espera na fila, tamanho da fila e tamanho do cache de mensagens. A Figura 5.1.3 mostra a tela de configuração dos parâmetros do *Gateway* Conversor. A utilização dessas configurações no *Gateway* Conversor é discutida no item 5.2 deste capítulo.

Figura 5.1.3 - Configurações

5.1.4 Gerenciamento de empacotadores

O gerenciamento de empacotadores inclui as funções de inclusão, listagem, exclusão e edição de empacotadores. Essas funções estão disponíveis somente para usuários administradores.

No cadastro de um empacotador o usuário cadastra o nome e a descrição do empacotador, conforme a Figura 5.1.4.1. A Figura 5.1.4.2 mostra a listagem de empacotadores. Na listagem o usuário tem as opções de editar e excluir empacotadores. A exclusão só é permitida se não houver nenhuma mensagem cadastrada no sistema que use o empacotador.

Na edição de empacotadores o usuário pode alterar o nome e a descrição e incluir e excluir campos do empacotador. A Figura 5.1.4.3 mostra a tela de edição de empacotadores. Cada campo possui um identificador, que é o bit do *bitmap* que indica a presença do campo na mensagem.

Além disso, o campo tem uma descrição, um tamanho, um tipo e um *flag* dizendo se o campo tem *pad* ou não. A descrição serve apenas para auxiliar na interpretação do significado do campo, na Figura 5.1.4.3 é possível ver alguns exemplos de descrição de

campos. É importante ressaltar que essa descrição do campo não é utilizada na construção das mensagens.

O de pad diz se na hora de construir ou de ler uma mensagem ISO 8583 devemos completar o valor do campo com zeros a esquerda até que este tenha o tamanho exato informado no cadastro. Os tipos que podem ser escolhidos são: IFB_BITMAP, IFB_NUMERIC, IFB_LLNUM, IFB_CHAR, IFB_LLCHAR, IFB_LLLBINARY.

IFB significa que o campo é binário. O número de letras 'L' indica o número de bytes usados para representar o tamanho do campo na mensagem, ou seja, os campos do tipo IFB_LLNUM, IFB_LLCHAR e IFB_LLLBINARY podem ter um tamanho variável na mensagem. Já os campos IFB_CHAR e IFB_NUMERIC tem tamanho fixo na mensagem. Para esses campos de tamanho fixo o tamanho de pad não se aplica, pois serão colocados zeros a esquerda ou espaços em branco (conforme o tipo, CHAR ou NUMERIC) sempre que o valor passado na construção da mensagem for menor do que o valor especificado no empacotador.

Figura 5.1.4.1 – Cadastro de Empacotador

Figura 5.1.4.2 – Lista de empacotadores (Administrador)

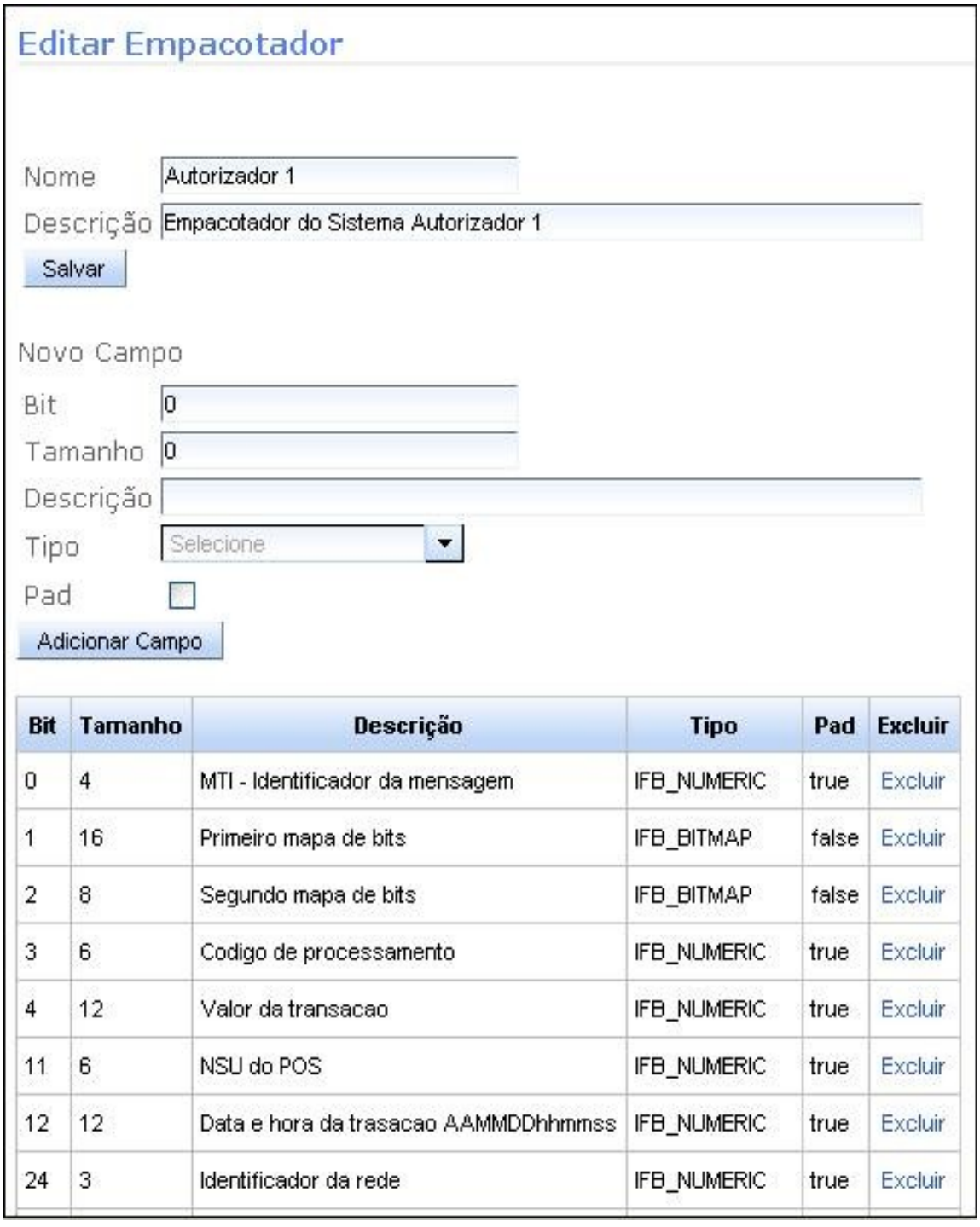

Figura 5.1.4.3 – Edição de Empacotador


Editar Mensagem
Descrição
Consulta Adiantamento
MTI
1600
Cód. Processamento
900001
IP do Host
localhost
Porta do Host
804
Bit
Tamanho
Tipo
Descrição
Parâmetro HTTP
1
16
IFB_BITMAP
Primeiro mapa de bits
-
2
8
IFB_BITMAP
Segundo mapa de bits
-
4
12
IFB_NUMERIC
Valor da transacao
valor
11
6
IFB_NUMERIC
NSU do POS
nsu
12
12
IFB_NUMERIC
Data e hora da trasacao AAMMDDhhmmss
dthora
24
3
IFB_NUMERIC
Identificador da rede
redeId
34
3
IFB_LLNUM
CVV
cvv
35
37
IFB_LLNUM
Trilha 2 do cartao
trilha2
39
2
IFB_NUMERIC
Codigo de resposta
codResp
41
8
IF_CHAR
Codigo do terminal
termId
42
15
IFB_NUMERIC
Identificacao do estabelecimento
estabId
43
40
IF_CHAR
Nome do Estabelecimento
estabName
48
4
IFB_LLCHAR
Versao da aplicacao
appVersion

Figura 5.1.4.4

5.1.5 Gerenciamento de Mensagens

O gerenciamento de mensagens inclui as funções de inclusão, listagem, exclusão e edição de mensagens. Essas funções estão disponíveis somente para usuários do tipo comum.

Para incluir uma nova mensagem o usuário deve selecionar qual empacotador será usado para empacotar a mensagem no formato ISO 8583. O usuário deve selecionar o empacotador através de uma lista de empacotadores, conforme a Figura 5.1.5.1.

Após selecionar o empacotador o usuário é direcionado para uma tela onde ele fará a configuração de roteamento e de conversão da mensagem. Nessa tela o usuário cadastra uma descrição da mensagem, o MTI, o código de processamento e o IP e porta do *host* para o qual a mensagem será enviada após ser convertida para o formato ISO 8583. O código de processamento não é um campo definido pelo padrão ISO 8583, mas na prática é o campo que identifica a mensagem. Isso porque, podem haver mais de uma mensagem com o mesmo MTI, e nesses casos o código de processamento é que define como a mensagem deve ser tratada.

Além disso, as regras de conversão dos parâmetros da requisição HTTP para os campos da mensagem ISO 8583 também são cadastradas nessa tela. Um exemplo dessa tela é mostrado na Figura 5.1.5.2.

Figura 5.1.5.1 – Lista de empacotadores (Usuários)

Figura 5.1.5.2 – Edição de Mensagem, configuração das regras de conversão e roteamento

5.2 Gateway Conversor

O *Gateway* Conversor é responsável por fazer a conversão e o roteamento das mensagens conforme as configurações feitas no *Gateway Manager*. O *Gateway* Conversor e o *Gateway Manager* acessam a mesma base de dados, dessa forma o *Gateway* Conversor recupera as informações gravadas pelo *Gateway Manager* e as utiliza tanto no seu funcionamento interno (*pool* de *threads*, tamanho da fila, etc...) quanto na conversão e roteamento das mensagens.

Para o acesso ao banco de dados foi desenvolvida uma biblioteca que abstrai a interface com o banco de dados. Esse biblioteca contém as classes Java que fazem o mapeamento para as tabelas do banco de dados e com as classes que fazem os acessos (*queries*, *inserts* e *updates*) ao banco de dados [10]. Essa biblioteca é utilizada tanto pelo *Gateway* Conversor como pelo *Gateway Manager*. Portanto, não houve replicação de código nas aplicações. No desenvolvimento dessa biblioteca foi utilizado o padrão DAO [17].

O *Gateway* Conversor é um aplicativo Java empacotado em um arquivo Jar que deve ser executado por linha de comando [15]. Ao iniciar sua execução o *Gateway* Conversor busca do banco de dados os parâmetros de configuração: porta, tamanho inicial do *pool*, tamanho máximo do *pool*, tempo de espera na fila, tamanho da fila e tamanho do cache de mensagens. Ou seja, os parâmetros que foram configurados no *Gateway Manager* pelo usuário administrador.

O *Gateway* Conversor pode receber tratar várias requisições concorrentemente. Para trabalhar em concorrência o *Gateway* Conversor possui um *pool* de *threads*. O uso de um *pool* de *threads* permite que o *Gateway* Conversor trate requisições concorrentemente sem o overhead de ter que instanciar um novo objeto sempre que receber uma nova requisição. Com isso, conseguimos um desempenho melhor do que se fossemos colocar todas as requisições em uma fila e ir tratando uma por uma de forma sequencial.

O tamanho inicial do *pool* e o tamanho máximo do *pool* configuram as dimensões do *pool*.

O tempo de espera na fila é o tempo (em milissegundos) que um requisição pode ficar esperando para ser atendida. As requisições tem que ficar esperando quando o número de requisições que o *Gateway* Conversor está tratando é maior que o tamanho máximo do *pool*.

O tamanho da fila define quantas requisições podem ficar esperando ao mesmo tempo para serem executadas. É nessa fila que as requisições esperam antes de serem executadas, e caso o tempo de espera seja ultrapassado, a requisição é retirada da fila e não é mais tratada.

O tamanho do cache de mensagens é utilizado para configurar o cache de mensagens do *Gateway* Conversor. Esse cache serve para que o *Gateway* Conversor não tenha que buscar do banco de dados o packager e as regras de conversão toda vez que receber uma nova requisição. Quando o cache está cheio e o *Gateway* Conversor tenta buscar uma mensagem que ainda não está no cache, ele tem que retirar uma mensagem do cache para incluir a nova mensagem. A mensagem que é excluída do cache é aquela que não é utilizada a mais tempo. Para fazer esse controle utilizamos uma fila. Na primeira posição da fila temos sempre a mensagem que não é usada há mais tempo e no final da fila temos a última mensagem utilizada. Quando uma mensagem que já está no cache é utilizada tiramos ela da fila e colocamos de volta no final da fila. Da mesma forma, uma mensagem que acabou de entrar no cache vai para o final da fila.

Com esses parâmetros o *Gateway* Conversor instancia e configura um objeto do tipo *HttpServer*. Com exceção do parâmetro tamanho do cache de mensagens, que é utilizado para instanciar o objeto *Cache*. Em seguida o *Gateway* Conversor passa a agir como um servidor HTTP e fica escutando na porta que foi configurada.

O *Gateway* Conversor espera receber requisições HTTP onde o método seja GET [3]. Quando recebe uma nova requisição o *Gateway* Conversor verifica se nos parâmetros do GET vieram o MTI, o código do usuário e o código de processamento. Caso esteja faltando algum desses parâmetros um erro é retornado.

Após validar a presença desses parâmetros o *Gateway* Conversor busca o usuário no banco de dados pelo código do usuário enviado na requisição HTTP. Caso o usuário não seja encontrado na base de dados retorna um erro pro cliente.

Em seguida o *Gateway* Conversor busca as regras de roteamento e o empacotador no cache, caso não encontre ele busca esses dados no banco de dados. Só então as regras de conversão são aplicadas, ou seja, os parâmetros da requisição HTTP são colocadas em uma mensagem ISO 8583 que em seguida é empacotada e enviada para o host de destino.

Após enviar a mensagem o *Gateway* Conversor aguarda o recebimento da resposta para então aplicar novamente as regras de conversão e montar a resposta para o cliente. A resposta é montada no mesmo formato que os parâmetros da requisição HTTP chegaram.

5.3 Disparador HTTP e Rebatedor ISO 8583

Para a realização dos testes do *Gateway* Conversor foram desenvolvidas 2 aplicações: Disparador HTTP e Rebatedor ISO 8583.

O Disparador HTTP envia requisições HTTP para o *Gateway* Conversor. O método enviado na mensagem é o GET [3]. Para configurar o Disparador HTTP utilizamos um arquivo de propriedades passado como parâmetro no momento da execução. Através desse arquivo podemos configurar os parâmetros enviados na requisição, a URL do *Gateway* Conversor e o número de transações enviadas por segundo.

O Rebatedor ISO 8583 simula um servidor que recebe e responde mensagens no formato ISO 8583. A configuração do Rebatedor também é feita por arquivo de propriedades. Para simular o processamento que ocorre em servidores reais, definimos um parâmetro que indica o tempo que o Rebatedor deve aguardar antes de responder a mensagem. Isso porque, se o Rebatedor simplesmente recebesse uma mensagem e respondesse logo em seguida, esse processo seria tão rápido que não simularia um servidor real.

Tanto o Disparador HTTP quanto o Rebatedor ISO 8583 gravam um arquivo de log com dados das mensagens enviadas e recebidas. Dessa forma, podemos analisar a performance do *Gateway* Conversor. Pois, analisando os logs podemos ver as mensagens trafegando entre o Disparador, o *Gateway* Conversor e o Rebatedor.

6 Experimentos

Nesse capítulo iremos apresentar os resultados dos experimentos realizados. Antes de iniciar os experimentos, realizamos diversos testes para validar o funcionamento do *Gateway* Conversor. Através de depuração do código e de análise de log verificamos que o mesmo estava funcionando da forma esperada, realizando as conversões e os roteamentos corretamente. Após esses testes iniciamos a preparação do ambiente para os experimentos.

6.1 Preparando o ambiente

Para a realização dos experimentos configuramos 2 empacotadores no *Gateway Manager* e cadastramos 4 usuários do tipo cliente. Para cada usuário cadastramos 3 tipos de mensagens, cada uma com suas próprias regras de conversão e de roteamento. Portanto, cadastramos um total de 12 mensagens, sendo que 6 foram configuradas para utilizar um empacotador e 6 para utilizar outro empacotador.

Na execução de cada experimento utilizamos 4 disparadores HTTP, 2 rebatedores ISO 8583 e 1 *Gateway* Conversor. Cada disparador foi configurado através do seu arquivo de propriedades para atuar como um dos clientes cadastrados anteriormente no *Gateway Manager*. Ou seja, no arquivo de propriedades configuramos o código do cliente e os campos enviados nos *requests*, conforme as regras de conversão cadastradas no *Gateway Manager*.

Cada rebatedor foi configurado para utilizar um dos empacotadores cadastrados no *Gateway Manager.* Dessa forma, cada um deles foi responsável por receber 6 tipos de mensagens diferentes.

O *Gateway* Conversor recebia as mensagens de todos os disparadores e fazia o roteamento para um dos rebatedores conforme o tipo da mensagem.

O número de clientes, de empacotadores, de mensagens e de rebatedores foi escolhido arbitrariamente, mas acreditamos que esse cenário seja suficiente para avaliar o

desempenho de execução do *Gateway* Conversor, onde este atende vários clientes concorrentemente e faz o roteamento das mensagens para mais de um servidor.

Foram utilizados 7 computadores nos experimentos, sendo 4 rodando disparadores HTTP, 1 rodando o *Gateway* Conversor e 2 rodando os rebatedores ISO 8583. A configuração dos computadores era a seguinte: Intel(R) Core 2 Duo, 2.93GHz, 4 GB de RAM, Sistema Operacional Windows 7 de 32 bits.

6.2 Execução dos experimentos

Foram realizados um total de 63 experimentos. Em cada experimento o *Gateway* Conversor recebeu 1000 requisições, sendo 250 de cada disparador. O objetivo dos experimentos foi analisar a capacidade do *Gateway* Conversor de tratar múltiplas requisições simultâneas. Foram utilizadas 9 configurações diferentes nos parâmetros do *Gateway* Conversor, sendo que 5 delas variavam em todos os parâmetros (linhas de 1 a 5 na Tabela 6.1) e 4 tinham apenas um parâmetro alterado em relação a configuração que apresentou os melhores resultados (linhas de 6 a 9 na Tabela 6.1). Ao fazer isso, nosso objetivo foi analisar quais parâmetros têm mais influência no desempenho do *Gateway* Converor. A Tabela 6.1 mostra as configurações utilizadas. Chamaremos as configurações de C1, C2, C3 e assim por diante até C9, para podermos referenciar cada uma delas posteriormente.

Configuração	Cache de Mensagens	*Pool* de *Threads* Inicial	*Pool* de Threads Máximo	Tempo de espera (*seg*)	Tamanho da Fila
C1	4	16	64	3	64
C2	8	32	128	5	128
C3	16	64	256	10	256
C4	24	128	512	15	512
C5	32	256	1024	20	1024
C6	4	256	1024	20	1024
C7	32	16	64	20	1024
C8	32	256	256	3	1024
C9	32	256	256	20	64

Tabela 6.1 – Configurações do Gateway Conversor

Note que C6, C7, C8 e C9 são variações da C5. Onde, C6 varia no tamanho do Cache, C7 varia no tamanho inicial e máximo do *Pool* de *Threads*, C8 varia no tempo de espera e C9 varia no tamanho da fila.

Com cada uma das configurações foram realizados 7 experimentos, que variavam no número de requisições por segundo. Em cada experimento cada disparador enviou 250 requisições HTTP, distribuídas ao longo do tempo conforme a configuração de frequência de envio. A Tabela 6.2 mostra as frequências de requisições por segundo utilizadas.

Identificador da frequência	Requisições por segundo por disparador	Total de requisições por segundo
1	50	200
2	100	400
3	200	800
4	300	1200
5	500	2000
6	700	2800
7	1000	4000

Tabela 6.2 – Número de requisições por segundo

A coluna do meio da Tabela 6.2 mostra o valor que era configurado em cada disparador antes da execução do experimento e a coluna direita mostra a frequência de requisições por segundo que o *Gateway* Conversor recebia em cada experimento. O identificador da frequência será utilizado para referenciarmos uma determinada frequência quando estivermos analisando os resultados.

6.3 Resultados

Após a execução dos experimentos iniciamos a análise dos resultados através dos logs dos disparadores. Através dessa análise coletamos 2 dados: número de respostas recebidas com sucesso e tempo médio de resposta. O tempo de resposta é o tempo entre o envio da requisição HTTP e o recebimento da resposta pelo disparador.

Com esses dados pudemos fazer comparações entre as configurações utilizadas e analisar quais parâmetros influenciaram mais o desempenho do *Gateway* Conversor. O termo desempenho será utilizando no texto para referenciar o número de respostas recebidas com sucesso pelos disparadores e o tempo transcorrido entre o envio da requisição e o recebimento da resposta. Quanto maior o número de respostas recebidas com sucesso, melhor o desempenho do *Gateway* Conversor. E quanto menor o tempo transcorrido entre o envio da requisição e o recebimento da resposta, melhor o desempenho do *Gateway* Conversor. Utilizaremos também o termo "tempo de resposta" para referenciar o tempo entre o envio de uma requisição e o recebimento da resposta.

A Figura 6.1 mostra o gráfico das respostas recebidas com sucesso. No eixo horizontal temos o identificador da frequência de envio de requisições e no eixo vertical temos o número de respostas recebidas com sucesso.

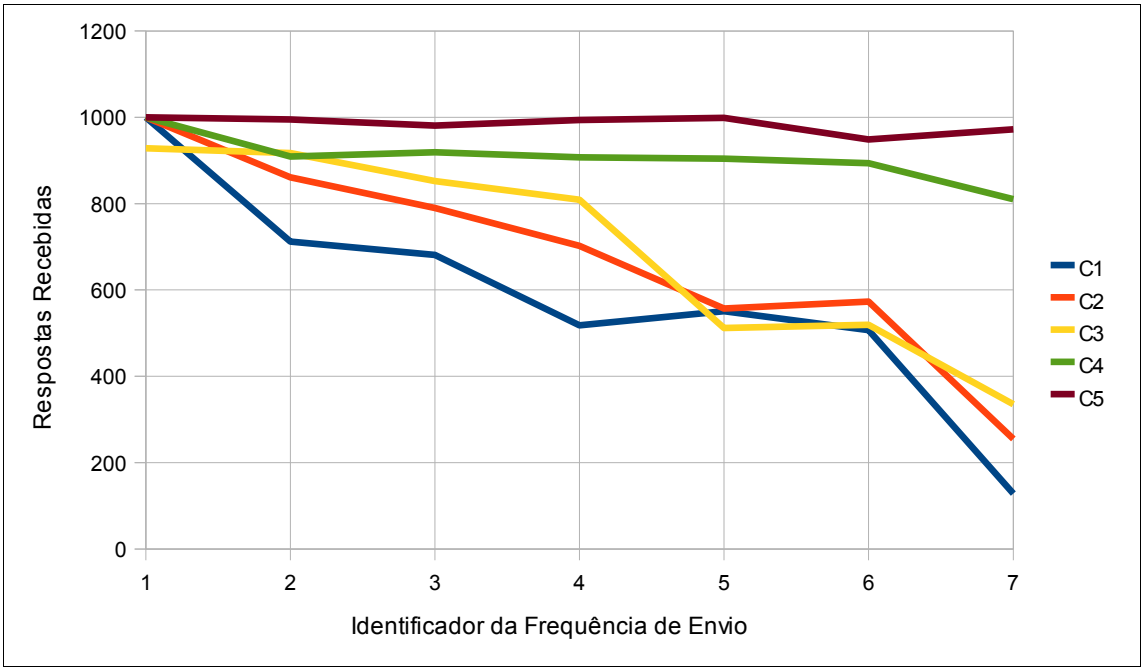

Figura 6.1 – Gráfico de respostas recebidas

Podemos ver que utilizando a configuração C5 o *Gateway* Conversor teve o melhor desempenho e com as configurações C1, C2 e C3 o desempenho foi caindo conforme o número de requisições por segundo foi aumentando. Enquanto que, com as configurações C4 e C5 o desempenho se manteve praticamente constante. Porém, acreditamos que se aumentássemos ainda mais o número de requisições por segundo, o desempenho com essas configurações iria cair gradativamente.

A Figura 6.2 mostra a comparação entre os tempos de respostas com as configurações C4 e C5. A ideia de fazer essa comparação surgiu após a analise do gráfico da Figura 6.1, pois, analisando o gráfico vimos que com as configurações C4 e C5 o *Gateway* Conversor teve um desempenho bem melhor do que com as outras configurações. Dessa forma surgiu a ideia de comparar também o tempo médio entre o envio da requisição e o recebimento da resposta.

No gráfico da Figura 6.2 temos no eixo horizontal o identificador da frequência de envio de requisições e no eixo vertical o tempo médio de resposta em milissegundos.

Podemos ver que o tempo médio de resposta foi aumentando conforme aumentava a frequência de envio de requisições, porém, o desempenho foi melhor com a configuração C5, pois o tempo médio de resposta se manteve menor.

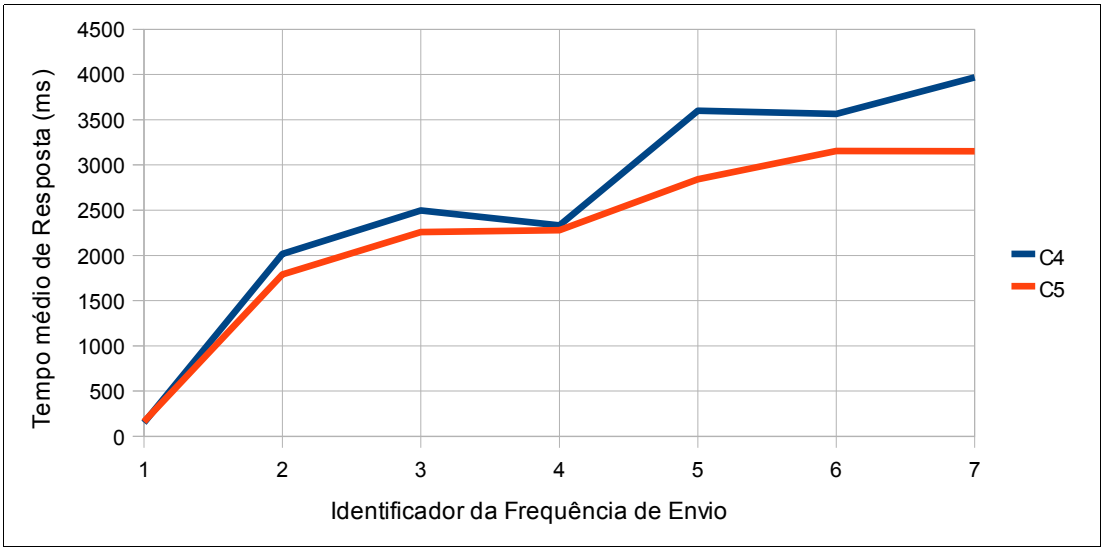

Figura 6.2 – Gráfico do tempo médio de resposta

Nesse ponto dos experimentos chegamos a conclusão que o desempenho do *Gateway* Conversor aumenta conforme os valores dos parâmetros de configuração aumentam. Isso porque, com a configuração C5 tivemos os melhores resultados, e essa configuração tem valores maiores nos parâmetros de configuração do que as outras.

Decidimos então, fazer uma analise da influência de cada parâmetro no desempenho do *Gateway* Conversor. Para isso, criamos as configurações C6, C7, C8 e C9, com os parâmetros configurados conforme a Tabela 6.1. Os resultados desses experimentos são mostrados nas figuras 6.3 e 6.4.

A Figura 6.3 mostra o gráfico das respostas recebidas com sucesso. No eixo horizontal temos o identificador da frequência de envio de requisições e no eixo vertical temos o número de respostas recebidas com sucesso. Analisando esse gráfico chegamos a algumas conclusões. A primeira é que o desempenho do *Gateway* Conversor depende do

conjunto de parâmetros, pois, como podemos ver no gráfico o desempenho com a configuração C5 ainda foi o melhor.

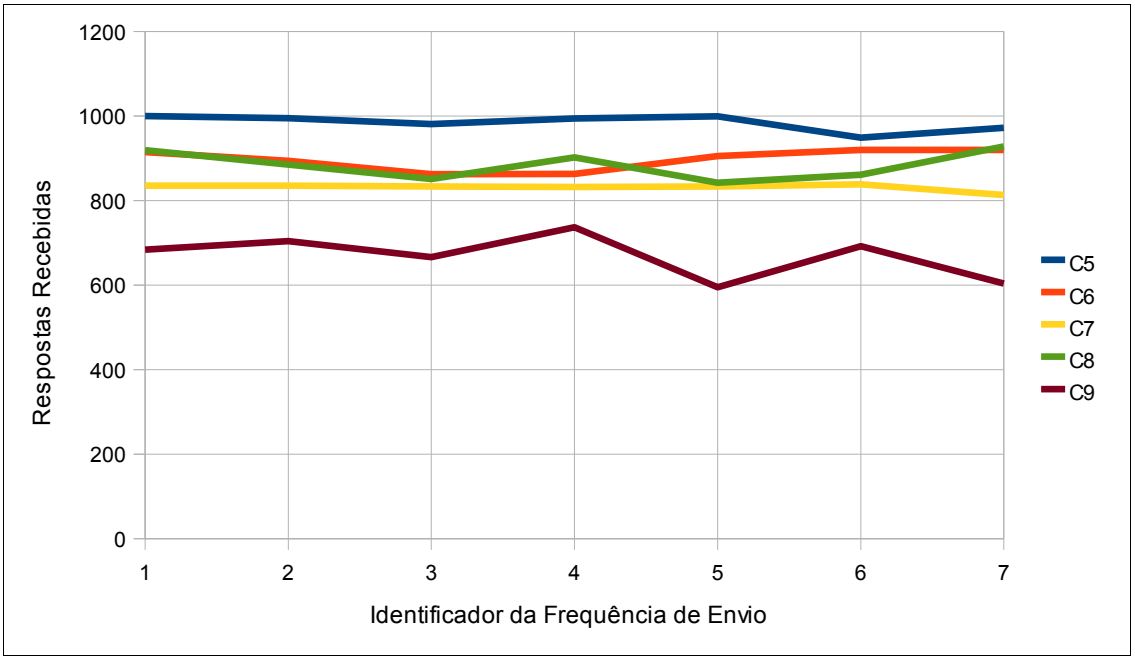

Figura 6.3 – Gráfico de respostas recebidas

Podemos notar também que com as configurações C6 e C8 o desempenho foi muito parecido. Isso mostra que o tamanho do cache e tempo de espera na fila influenciaram de forma semelhante o desempenho do *Gateway* Conversor.

Com a configuração C7 o desempenho foi o segundo pior, mostrando que as configurações do *Pool* de *Threads* tem mais influencia do que o tamanho do cache e o tempo de espera da fila.

Porém, o resultado que mais nos chamou a atenção foi o desempenho do *Gateway* Conversor utilizando a configuração C9. Como pode ser visto na Figura 6.3 o desempenho com essa configuração foi muito abaixo do que com as demais.

Portanto, concluímos que o tamanho da fila é o parâmetro que tem mais impacto no desempenho. Acreditamos que isso ocorra porque a fila de espera é a porta de entrada do

Gateway Conversor. É onde as requisições ficam esperando para serem atendidas, ou seja, é o ponto onde uma requisição pode ser rejeitada de imediato.

Pensando nos outros parâmetros podemos entender melhor: se uma mensagem não está no cache, o *Gateway* Conversor acessa o banco da dados para buscar as informações de regras de conversão e de roteamento; se não há *threads* disponíveis no *pool*, o *Gateway* Conversor cria uma nova *thread*; o tempo de espera na fila pode ter um impacto maior quando o *Gateway* Conversor demora para receber a resposta da mensagem ISO 8583, o que não ocorreu nos experimentos, pois os rebatedores demoravam no máximo 200 milissegundos para responder a mensagem ISO 8583.

Entretanto, uma requisição só começa a ser tratada depois que está na fila. Se não há espaço para ela na fila não há nada que o *Gateway* Conversor possa fazer. Ao contrário do que acontece no caso de não encontrar um determinado objeto no cache ou não haver *threads* disponíveis no *pool*, onde o *Gateway* Conversor tem uma alternativa e pode tratar a mensagem.

7 Conclusões

Nesse trabalho desenvolvemos um *Gateway* Conversor capaz de converter mensagens HTTP em mensagens ISO 8583 e um sistema web, denominado *Gateway Manager*, responsável por prover uma interface para a realização das configurações do *Gateway* Conversor e das regras de conversão e roteamento de mensagens. Para auxiliar nos experimentos desenvolvemos um disparador de mensagens HTTP e um rebatedor de mensagens ISO 8583.

Durante todo o desenvolvimento tivemos a preocupação de criar programas fáceis de serem configurados. No caso do Disparador HTTP e do Rebatedor ISO 8583 utilizamos arquivo de propriedades, e no caso do *Gateway* Conversor utilizamos o *Gateway Manager*. Isso nos permitiu alterar o funcionamento de cada programa sem haver a necessidade de alterar código. Outra preocupação foi gravar todas as ações dos programas em arquivos de log, pois foi através da análise desses arquivos que validamos o funcionamento correto do *Gateway* Conversor e analisamos seu desempenho. Além disso, os arquivos de log foram muito úteis para corrigirmos erros no código durante o desenvolvimento.

Através da análise dos resultados obtidos nos experimentos concluímos que o desempenho do *Gateway* Conversor depende do conjunto de parâmetros configurados para o seu funcionamento e não somente de um parâmetro específico. Concluímos também que a alteração nos parâmetros do *Gateway* Conversor influencia tanto na capacidade de tratar requisições simultâneas quanto na velocidade para tratar essas requisições. Por fim, concluímos que o parâmetro que tem mais influência no desempenho do *Gateway* Conversor é o tamanho da fila de espera.

Como trabalho futuro incluímos a extensão do Gateway Conversor para trabalhar com mais protocolos e não somente HTTP e ISO 8583. Um candidato a ser incluído no *Gateway* Conversor é o padrão TISS (Troca de Mensagens em Saúde Suplementar), que é um padrão estabelecido para pela ANS (Agência Nacional de Saúde) para registro e intercâmbio de dados entre operadoras de planos saúde. Nesse caso o *Gateway* Conversor receberia requisições HTTP e converteria para o padrão TISS.

8 Referências Bibliográficas

[1] A. Henry-Labordère and V. Jonack. *SMS and MMS Interworking in Mobile Networks*, 1ª ed, Artech House Publishers, 2004

[2] D. Williams. *Pro PayPal E-Commerce*, 1ª edição, ed. Apress, 2007

[3] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice. HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*, 1ª edição, ed. Addison-Wesley Professional, 2001

[4] ISO/TC 68/SC 6, *ISO 8583-1:2003, Financial transaction card originated messages - Interchange message specifications - Part 1: Messages, data elements and code values,* 2007

[5] ISO/TC 68/SC 6, *ISO 8583-2:1998, Financial transaction card originated messages - Interchange message specifications - Part 2: Application and registration procedures for Institution Identification Codes (IIC)*, 2007

[6] ISO/TC 68/SC 6, *ISO 8583-3:2003, Financial transaction card originated messages - Interchange message specifications - Part 3: Maintenance procedures for messages, data elements and code values,* 2007

[7] C. Schalk, E. Burns and J. Holmes, *JavaServer Faces: The Complete Reference*, 1ª edição, ed. McGraw-Hill Osborne Media, 2006

[8] B. Aranda and Z. Wadia, *Facelets Essentials: Guide to JavaServer Faces View Definition Framework*, 1ª edição, ed. Apress, 2008

[9] J. Machacek, J. Ditt, A. Vukotic and A. Chakraborty, *Pro Spring 2.5*, 1ª edição, ed. Apress, 2008

[10] C. Bauer and G. King, *Java Persistence with Hibernate*, 2ª edição, ed. Manning Publications, 2006

[11] S. Gupta, *Pro Apache Log4J*, 2ª edição, ed. Apress 2005

[12] http://www.jpos.org/ consultado em 27 de Julho de 2010

[13] http://www. mysql.com consultado em 27 de Julho de 2010

[14] http://tomcat.apache.org/ consultado em 27 de Julho de 2010

[15] http://java.sun.com/ consultado em 27 de Julho de 2010

[16] http://netbeans.org/ consultado em 27 de Julho de 2010

[17] D. Alur, J. Crupi and D. Malks, *Core J2EE Patterns*, 2ª edição, ed. Prentice Hall, 2003

[18] J. Keogh, *J2ME: The Complete Reference,* 1ª edição, ed. McGraw-Hill, 2003

[19] http://www.smsxchange.com/main/default.asp consultado em 03 de Agosto de 2010

[20] http://www.tm4b.com/ consultado em 03 de Agosto de 2010

[21] https://www.paypal.com/cgi-bin/webscr?cmd=_payflow-gateway-overview-outside consultado em 03 de Agosto de 2010

[22] http://www.authorize.net/ consultado em 03 de Agosto de 2010

[23] F.J. Kurose, e W.K Ross, Redes de computadores e a Internet, 3ª edição, ed. Addisson Wesley, 2006

[24] http://www.ans.gov.br/portal/site/_hotsite_tiss/materia.htm, consultado em 07 de Setembro de 2010, compatível somente com Internet Explorer