CENTRO UNIVERSITÁRIO FEEVALE

JOÃO RODRIGO MATTE

Projeto de Alta Disponibilidade para Processamento de Transações Eletrônicas

Novo Hamburgo, novembro de 2009.

JOÃO RODRIGO MATTE

joaomatte@gmail.com

Projeto de Alta Disponibilidade para Processamento de Transações Eletrônicas

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso

Professor orientador: Edvar Bergman Araújo

Novo Hamburgo, novembro de 2009.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus por me proporcionar esta oportunidade.

Agradecer a todos que sempre estiveram do meu lado em todos os momentos.

Agradeço também ao meu orientador prof. MS. Edvar Bergman Araujo que sempre me ajudou e principalmente me incentivou neste trabalho.

RESUMO

Com a evolução da tecnologia e a facilidade de uso, os meios de pagamentos eletrônicos estão cada vez mais comuns nos dias de hoje. Por isto, os sistemas que processam essas informações são extremamente críticos e precisam estar em constante evolução para acompanhar o crescimento desta utilização. A indisponibilidade destes sistemas pode causar prejuízos incalculáveis. Desta forma, este projeto apresenta uma proposta e resultados da migração de um sistema de processamento de transações eletrônicas defasado para um novo sistema, totalmente remodelado focado em alta performance e principalmente em alta disponibilidade.

Palavras-chave: Alta disponibilidade. Escalabilidade. Cluster. Desempenho. Contingência.

ABSTRACT

The evolution of the technology and the facility of use, the ways of eletronics payments are nowadays more usual. In this way, the systems that process those informations are extremely critics and need to keep going in evolution for following the growing of this utilization. The unavailability of these systems might cause damage non mensurable. However, this project shows a migration proposal of a eletronic transactions processing system diferently to a new system, totally remodeled focused on high performance and mainly high availability.

Key-words: High availability, Scalability. Cluster. Performance. Contingency.

LISTA DE FIGURAS

Figura 1.1 – Processamento de uma transação eletrônica. 22
Figura 2.1 – Modelo atual da organização 24
Figura 4.1 – Proposta de Arquitetura 36
Figura 4.2 – Ambiente de aplicação clusterizada 38
Figura 4.3 – Conexões dos módulos. 43
Figura 4.4 – Visão macro do ambiente do SC. 44
Figura 5.1 – Diagrama de seqüência de uma transação eletrônica 51
Figura 5.2 – Proposta de base de dados para as transações 54
Figura 5.3 – Proposta de base de dados de consultas 61
Figura 6.1 – Volume transacional no novo ambiente 69

LISTA DE QUADROS

Quadro 5.1 – Estrutura da documentação 46
Quadro 5.2 – Descrição da tabela compra_padrao 55
Quadro 5.3 – Descrição da tabela cancelamento 57
Quadro 5.4 – Descrição da tabela desfazimento 57
Quadro 5.5 – Descrição da tabela consulta 58
Quadro 5.6 – Descrição da tabela sonda_operadora_padrao 59
Quadro 5.7 – Descrição da tabela sonda_captura 59
Quadro 5.8 – Descrição da tabela terminal_ultima_transacao 60
Quadro 5.9 – Descrição da tabela fechamento_diario_padrao 60
Quadro 5.10 – Descrição da tabela estabelecimento 61
Quadro 5.11 – Descrição da tabela estab_operadora 62
Quadro 5.12 – Descrição da tabela operadora 62
Quadro 5.13 – Descrição da tabela sonda 63
Quadro 5.14 – Descrição da tabela terminal 63

LISTA DE ABREVIATURAS E SIGLAS

SC – Sistema Concentrador

SA – Sistema Autorizador

TRN – Transação

NSU – Número seqüencial único

TPS – Transações por segundo

RAC – *Real Appication Cluster* (Cluster de Aplicação Real)

JVM – *Java Virtual Machine* (Máquina Virtual Java)

TCP/IP – *Transmission Control Protocol Internet Protocol* (Protocolo de Controle da Conexão na Internet)

API – *Application Programming Interface* (Interface de Programação de Aplicativos)

SUMÁRIO

1 TRANSAÇÕES ELETRÔNICAS - SISTEMAS ENVOLVIDOS 16
1.1 Sistema Concentrador 16
1.1.1 Processamento da transação 17
1.1.2 Resolução de Pendências 18
1.2 Sistema Autorizador 19
1.2.1 Processamento da Transação 19
1.2.2 Resolução de Pendências 20
1.3 Resumo do Funcionamento 21
2 SISTEMA CONCENTRADOR 23
2.1 Ambiente 24
2.2 Problemas 25
2.2.1 Problemas de comunicação 25
2.2.2 Problemas de memória 26
2.2.3 Problemas de lentidão com o banco de dados 26
2.2.4 Problemas de inflexibilidade e rigidez 27
2.2.5 Dificuldades na substituição de servidores 28
3 TECNOLOGIAS ENVOLVIDAS 29
3.1 *JAVA* 1.5 29
3.2 *Framework Spring* 30

3.3 *Framework Hibernate* 31
3.4 *Framework Apache Mina* 32
3.5 *Framework Quartz* 32
3.6 *Framework Ehcache* 33
3.7 *Oracle RAC - Oracle Real Application Clusters* 33
4 SOLUÇÃO PROPOSTA 35
4.1 Arquitetura de Hardware 35
4.2 Arquitetura do Software 38
4.2.1 Módulo Cache 38
4.2.2 Módulo Resolvedor 39
4.2.3 Módulo Processamento 40
4.2.4 Módulo Atendedor Genérico 40
4.2.5 Módulo de Conciliação de Transações 41
4.2.6 Módulo de Sonda 41
4.2.7 O Sistema como um todo 41
5 MODELAGEM DO SISTEMA 45
5.1 Estrutura dos documentos de modelagem 45
5.2 Detalhamento do Sistema 46
5.3 Detalhamento dos *Use-cases* 48
5.4 Diagrama de Sequência 50
5.5 Formato alternativo de detalhamento de *Use-Cases* 51
5.6 Modelo do diagrama entidade-relacional 54
5.6.1 Proposta de base de dados para armazenamento de transações 54
5.6.2 Proposta de base de dados para consultas administrativas 61
6 IMPLANTAÇÃO DO PROJETO 64
6.1 Desenvolvimento do projeto 64
6.2 Projeto Piloto 64

6.3 Produção 65
7 CONCLUSÃO 71
8 REFERÊNCIAS BIBLIOGRÁFICAS 72

INTRODUÇÃO

O mercado de cartões e pagamentos eletrônicos tem apresentado um grande crescimento nos últimos anos. Os cartões deixaram de ser utilizados em poucas operações nas agências bancárias e passaram a ser utilizados e aceitos nos mais diversos lugares do mundo. Algumas razões para isto são a segurança e a facilidade do uso. Além disto, muitas vantagens estão agregadas nas compras pagas com cartões, como sorteios instantâneos, premiações, milhas aéreas, entre outras. Mas, talvez a principal vantagem seja a possibilidade de parcelar uma compra pelo mesmo preço a vista. Isto acabou aumentando o consumo, a utilização e a difusão do uso do cartão. (Associação Brasileira de Cartões de Crédito e Serviços, 2009)

O pagamento eletrônico concorre diretamente com o uso do dinheiro em espécie e cheques, que eram as duas formas mais comuns de pagamento. É mais seguro as pessoas portarem um cartão, que em caso de perda ou roubo, pode ser facilmente bloqueado. Os estabelecimentos também são beneficiados com este tipo de pagamento, pois assim não correm o risco de receber cheques sem fundos ou dinheiro em espécie falso.

O crescimento do mercado de cartões deve muito a tecnologia que agiliza o processamento de todas as informações necessárias para a logística de uma transação eletrônica com cartão funcionar perfeitamente. Uma transação eletrônica envolve pelo menos quatro atores, são eles: o cliente; o estabelecimento que está vendendo produtos ou serviços (hotéis, postos de gasolinas, restaurantes, etc.); a empresa que captura as informações da compra (Getnet, Redecard, VisaNet); e por último a empresa que processa as informações da compra (Itaú, Banco do Brasil, Banrisul, etc.).

Este projeto está focado na captura das informações do pagamento eletrônico, ou seja, o sistema que está entre o estabelecimento e a empresa que processa as informações do pagamento efetuado pelo cliente. Este processo de validação de transações eletrônicas não é

tão simples como passar o cartão em um meio de captura, este é apenas o primeiro passo. Após isto, o meio de captura utiliza algum tipo de comunicação entre os diversos disponíveis, para enviar as informações necessárias para efetuar a transação à empresa que faz a captura das informações. Nesta fase, as informações são processadas, validadas e armazenadas para serem utilizadas no futuro. Tais validações são focadas no estabelecimento.

Seguindo o fluxo da transação, as informações são enviadas para a empresa de processamento, que também processa, valida e armazena informações, porém mais focada ao cliente portador do cartão. É nesta fase que é debitado o saldo do portador, criadas parcelas para compra parceladas, juros, etc. Após isto, as informações da transação e a resposta do processamento retornam para a empresa de captura que processa a resposta e repassa novamente ao meio de captura. Caso a compra tenha sido autorizada, o ticket será impresso e o meio de captura enviará uma confirmação de compra, que seguirá o mesmo fluxo anterior, ou seja, enviado para a empresa de captura e repassado para a empresa de processamento.

Justamente pela alta utilização e o crescimento de pagamentos eletrônicos o sistema chamado aqui de “Sistema Concentrador”, deve estar preparado para processar centenas de informações simultâneas com alta performance, baixo tempo de resposta e principalmente alta disponibilidade. A alta disponibilidade é vital para que uma transação eletrônica seja efetuada perfeitamente.

Uma das principais funções do Sistema Concentrador é manter as informações consistentes entre todos os envolvidos, ou seja, empresas de captura, operadoras de cartões e estabelecimentos. Cabe a ele resolver qualquer pendência, caso ocorra uma falha durante o progresso da transação. Quando este serviço fica indisponível acaba gerando prejuízos incalculáveis em cascata, ou seja, todos os atores envolvidos acabam sendo afetados financeiramente e ainda acaba causando a insatisfação do cliente.

O atual sistema concentrador da empresa de captura possui vários problemas e limitações. Este sistema foi projetado para processar apenas uma operadora de cartão e suas funcionalidades. Justamente por isto o sistema é pouco flexível e nada modulado. Com o passar do tempo, foi adicionado diversos novas operadoras de cartões. Atualmente, este sistema processa aproximadamente 30 operadoras, cada uma com funcionalidades distintas.

Este trabalho tem por finalidade desenvolver uma proposta de um novo sistema concentrador considerando características como: alta disponibilidade, escalabilidade, tolerância a falhas, processamento distribuído e balanceamento de carga.

A **Alta disponibilidade** é uma característica de um sistema informático resistente a falhas de software e energia, cujo objetivo é manter os serviços disponíveis o máximo de tempo possível. (Wikipédia, 2008).

É possível alcançar a escalabilidade horizontal ao adicionar mais máquinas ao ambiente atual, aumentando assim a capacidade do sistema como um todo. Para distribuir a carga de maneira otimizada para cada servidor, devem ser usadas técnicas de balanceamento de carga (*load balancing*) e de montagem de clusters. A arquitetura definida para a aplicação deve ser projetada tendo em vista que ela deverá funcionar sem problemas em um ambiente que pode escalar horizontalmente. Ao escalar de forma horizontal pode-se garantir flexibilidade, disponibilidade e confiabilidade para o ambiente e para a aplicação.

> Na sua forma mais básica, um cluster é um sistema que compreende dois ou mais computadores ou sistemas (denominados nodos) que trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma que os usuários do agrupamento de máquinas tenham a impressão de que somente um único sistema responde para eles, criando assim uma ilusão de um recurso único (computador virtual). (PITANGA, 2003, p. 14).

Tolerância a falhas consiste em um sistema que continue processando mesmo quando acontece alguma falha inesperada, ou seja, o objetivo é garantir a continuidade de um serviço, mesmo na presença de falhas de alguns componentes. As técnicas para garantir tolerância a falhas são caracterizadas pela redundância, onde os componentes (hardware ou software) ou informações são replicadas e estruturadas em grupo. (CIRNE, 2000).

Processamento distribuído é "um conjunto de processos executando em diferentes máquinas e interagindo através de mensagens, não havendo compartilhamento de memória física". (Instituto de Informática da UFRGS, 2008).

Balanceamento de carga possui como meta distribuir a carga entre os nós da rede ou do cluster. (PITANGA, 2003, p. 79).

Este trabalho está organizado em seis capítulos. O capítulo um abrange os principais sistemas envolvidos em um ambiente de processamento de transações eletrônicas. O capítulo dois aborda o Sistema Concentrador antigo da empresa de captura citando seus problemas. No capítulo três contém as principais tecnologias pesquisadas que serão utilizadas nesta proposta. Em seguida, o capítulo quatro aborda a solução do projeto. O Capítulo cinco

contém as informações da implantação do projeto em um ambiente real, seguido do capítulo seis que abrange a documentação utilizada para o desenvolvimento do projeto.

1 TRANSAÇÕES ELETRÔNICAS - SISTEMAS ENVOLVIDOS

Este capítulo aborda os dois principais sistemas envolvidos nas transações financeiras realizadas eletronicamente. Os sistemas são o sistema concentrador e o sistema autorizador. Para facilitar a explicação, a mesma será dividida em duas partes: o processamento das transações e a resolução de pendências.

Normalmente as transações eletrônicas são capturadas por uma empresa (Sistema Concentrador – SC) e processadas por outra (Sistema Autorizador – SA). Sendo assim, se faz necessário uma comunicação ágil utilizando o protocolo estabelecido entre ambas às empresas. Normalmente esse protocolo de comunicação é o TCP/IP padrão através de um link de internet dedicado.

1.1 Sistema Concentrador

O SC é o sistema responsável por validar, processar e manter as informações atualizadas e íntegras das transações eletrônicas que por ele são processadas. Tais validações são referentes à rede de transações.

Uma rede de transações é quando uma empresa possui diversos equipamentos eletrônicos capazes de capturar os dados de uma transação. Tais equipamentos localizam-se em diferentes locais referenciando-se à mesma central de processamento dos dados capturados, ou seja, ao mesmo SC. Sendo assim, a empresa constitui a sua própria rede de transações.

Antes de o sistema estar apto a processar pagamentos eletrônicos, uma série de cadastros são necessários. Primeiramente, é necessário efetuar os cadastros administrativos

contendo os dados dos clientes da organização, ou seja, os estabelecimentos que desejam transacionar na rede disponível. Basicamente tais cadastros consistem em manter os dados dos estabelecimentos atualizados, cadastro de meios de captura e operadoras. Um estabelecimento possui pelo menos um meio de captura e uma operadora aptos a transacionar na rede de transações.

Outro cadastro importante é o código do estabelecimento para o autorizador. Um SC trabalha com diversos sistemas autorizadores ao mesmo tempo. Desta forma, o estabelecimento cadastrado no SC com o código 0001 - Feevale, CNPJ: xxx possui outro cadastro no SA-1 onde seu código é 1000, já no SA-2 seu código é 2000. Então no momento da transação, cabe ao SC fazer o chamado “de-para” (de 0001 para 1000 no caso da transação ser autorizada pelo SA-1 ou de 0001 para 2000 no caso da transação ser autorizada pelo SA-2) do código do estabelecimento enviando o código conhecido do autorizador de acordo com o cadastro.

1.1.1 Processamento da transação

Uma transação inicia quando o meio de captura obtém os dados necessários para executar a transação. Em seguida, através de um meio de comunicação (ex.: internet) os dados são enviados ao SC que faz seu processamento e os encaminha para o SA. Após o processamento do SA, os dados voltam ao SC e logo depois ao meio de captura onde é impresso o *ticket*.

Detalhando um pouco mais este processo, destacam-se os seguintes estágios:

- **1º estágio –** pedido de compra: neste estágio o SC é responsável por armazenar as informações das transações financeiras e validar os dados administrativos da empresa do estabelecimento. A validação dos dados da transação, que consiste em verificar se todos os dados estão preenchidos corretamente. Há também a validação dos dados do terminal, estabelecimento e operadora. Com tudo validado, a transação é enviada ao SA e pode continuar sendo efetuada.
- **2º estágio –** resposta de compra: quando o SC recebe a resposta da transação do SA, este deve vincular os dados do autorizador ao pedido de compra

(estágio 1), atualizar a base de dados e retornar ao sistema solicitante os dados necessários para a transação continuar sendo efetuada.

- **3º estágio –** confirmação de compra: Quando o sistema concentrador recebe a confirmação da transação, este deve vincular os dados do autorizador à resposta de compra (estágio 2), atualizar a base de dados e retornar ao sistema solicitante os dados necessários para a transação financeira ser finalizada.

Os itens acima são de uma transação composta por três comunicações, onde a transação financeira ocorreu perfeitamente, ou seja, o *ticket* foi impresso. Basicamente, existem duas situações onde a transação financeira ocorre perfeitamente, porém, não ocorre a finalização da mesma. São elas:

- Quando o SC invalida a transação por alguma regra de negócio (estágio 1), exemplo “Estabelecimento inativo”.
- Quando o SA invalida a solicitação também por regras de negócio da operadora (estágio 2), exemplo “Cartão bloqueado”.

As transações que estiverem nas situações citadas acima são consideradas como transações que ocorreram perfeitamente e estão finalizadas. Para as transações que não foram finalizadas estas devem ser resolvidas de alguma maneira. Nenhuma transação financeira pode ficar pendente permanentemente dentro do SC.

1.1.2 Resolução de Pendências

O SC é responsável por resolver as pendências. Essa resolução pode ocorrer de duas maneiras:

- **Resolução de pendências on-line:** Através de um conjunto de regras de negócio, as transações financeiras ficam pendentes até a próxima transação do mesmo meio de captura na transação financeira anterior que não foi finalizada. Ou seja, sempre a transação atual resolve a transação financeira anterior, caso seja necessário. Quando ocorre uma resolução de pendência on-line, o SC pode ou não enviar a resolução para o sistema autorizador.

- **Resolução de pendências através de sondas:** Após a identificação do meio de captura o SC envia requisições de solicitação de status das transações que se encontram como “pendente” no sistema aos meios de captura e cabe a eles responderem os status das transações solicitadas. Com a resposta o SC toma a decisão de confirmar ou desfazer a transação financeira. O SC também fica responsável por gerar arquivos de conciliação. Tais arquivos são gerados e enviados diariamente ao autorizador contendo todas as transações efetuadas no dia anterior na rede de captura.

1.2 Sistema Autorizador

O SA é responsável por deixar ou não o portador do cartão efetuar uma transação financeira. O ciclo de utilização dos sistemas de autorização inicia quando são cadastrados novos clientes. Neste cadastro são armazenados todos os dados dos clientes, como data de vencimento da fatura, valor limite de compra, dependentes, etc.

O próximo passo é gerar os cartões individualmente para cada novo cliente. As regras dos sistemas de autorização já começam na própria geração dos cartões, pois neste momento é que são gerados cartões contendo o BIN específico para o cliente. BIN’s são os números dos cartões, estes números indicam o tipo de cartão que está sendo disponibilizado.

1.2.1 Processamento da Transação

O SA é estimulado quando recebe do SC uma solicitação de processamento contendo os dados de uma transação eletrônica para serem processados. Após o seu processamento, o SA retorna o resultado para o SC solicitante. São 2 os estágios do processamento:

- **1º estágio –** pedido de compra: O SA recebe as informações da transação do SC. Então entra em ação para validar os dados do cartão, como validade, desbloqueado, dados preenchidos na transação, etc. Após estas validações, o

autorizador passa a validar os dados do cliente juntamente com as regras de negócios da operadora do cartão. Ou seja, será validado o saldo do cliente, a senha digitada, entre outras validações que diferem de acordo com cada operadora. Se todos os dados foram validados, o saldo do cliente será sensibilizado, a transação financeira será armazenada como uma transação realizada com sucesso e o autorizador responderá ao solicitante (SC).

- **2º estágio –** confirmação de compra: Quando o sistema autorizador recebe a confirmação da transação, este deve atualizar o status do pedido de compra (estágio 1) para confirmada.

1.2.2 Resolução de Pendências

Outras funcionalidades necessárias em sistemas de autorização são processos de resoluções de pendências. Ou seja, caso uma transação por algum motivo tenha ficado pendente, esta precisa ser resolvida. Existem dois processos normalmente utilizados, um é efetuando a troca de arquivos de conciliação entre os sistemas concentradores e sistemas de autorização. O outro processo é através de sondas entre os sistemas de autorização com o sistema concentrador. A seguir detalhes destes 2 processos:

- **Arquivos de conciliação**: no processo de conciliação o SA processa diariamente arquivos gerados pelo SC, tais arquivos contêm todas as transações financeiras efetuadas no dia anterior. Como o SC é o que mantém as informações mais integras, o SA atualiza os status das transações de acordo com as informações dos arquivos.
- **Sondas**: no processo de resolução de pendências através de sondas o SA solicita através de transações administrativas o status de todas as suas transações pendentes ao sistema concentrador, quando recebe o retorno do concentrador, o autorizador finaliza suas transações com o status recebido.

1.3 Resumo do Funcionamento

Resumindo a explicação apresentada neste capítulo, uma transação financeira inicia quando um meio de captura é acionado em qualquer estabelecimento. Quando o meio de captura lê as informações dos cartões, este executa as perguntas conforme o tipo do cartão lido. Essas perguntas ficam dentro do próprio meio de captura. Com todas as perguntas respondidas, este deve requisitar a transação através da internet comunicando-se a um endereço, tal endereço refere-se à empresa que captura as informações das transações. O meio de captura fica conectado aguardando a resposta por um tempo configurável em sua aplicação.

A empresa de captura, por sua vez, valida a requisição recebida, registra e processa os dados para enviar via internet à requisição de autorização a empresa autorizadora. Também neste estágio é controlado um tempo de aguardo da resposta do autorizador. Quando o autorizador responde, o SC recebe e processa a resposta do autorizador, vinculando à requisição de compra do meio de captura a resposta vinda do autorizador.

Logo após, o SC responde ao meio de captura que processa a resposta e se tudo estiver correto, este deve imprimir o ticket do cliente. Depois da impressão do ticket, o meio de captura deve enviar a confirmação da transação ao SC, que irá processá-la e encaminhá-la para o sistema autorizador.

Neste cenário, pode-se observar que o SC é o único que processa três informações durante uma única transação, é por este motivo que este sistema tem as informações mais integras frente aos outros envolvidos. A figura a seguir ilustra o funcionamento de uma transação eletrônica.

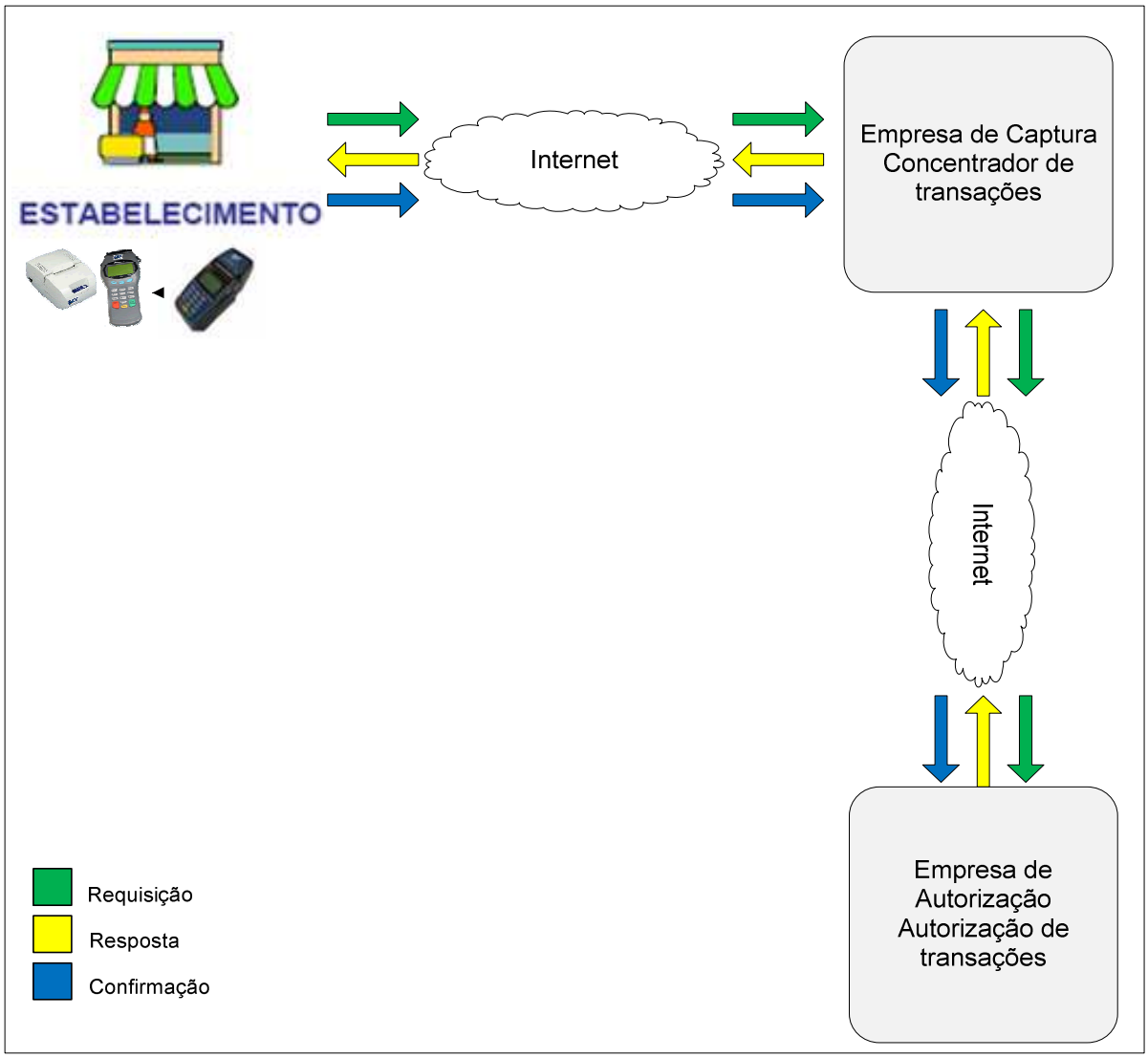

Figura 1.1 – Processamento de uma transação eletrônica.

Fonte: do autor.

2 SISTEMA CONCENTRADOR

O sistema concentrador surgiu no momento em que houve a necessidade da organização de centralizar as informações das transações trafegadas em sua rede. Antes do sistema concentrador, tais informações eram armazenadas em arquivos de *logs* e algumas em banco de dados através da utilização de *procedures*.

Com o crescimento da utilização da rede de transações, começaram a surgir os problemas administrativos das transações financeiras. Diante de diversas dificuldades foi projetado o primeiro SC de informações sobre as transações eletrônicas da organização. O Sistema foi projetado e implementado em pouco tempo apenas para suprir a necessidade acima citada.

Com o mercado eletrônico financeiro em constante evolução, logo surgiram administradoras de cartões querendo utilizar a rede de captura da organização. O fato era que o sistema não estava projetado para suportar diversas operadoras simultâneas. Por causa deste fato, hoje o sistema encontra-se saturado e no seu limite, tornando cada vez mais difícil e praticamente inviável tentar inserir novas operadoras para ele processar, visto que, pode acabar afetando todos os outros clientes nele existente.

2.1 Ambiente

O ambiente interno transacional da organização é composto por três camadas, são elas: sistemas de roteamento, sistemas concentradores e banco de dados onde são armazenadas as informações necessárias de acordo com cada produto ou cliente.

Os sistemas roteadores de transações também conhecidos como *gateway* tem a função de encaminhar as transações eletrônicas para seus devidos processamentos dentro da organização, no caso deste trabalho, ao sistema concentrador. Segue figura que ilustra o atual ambiente do SC.

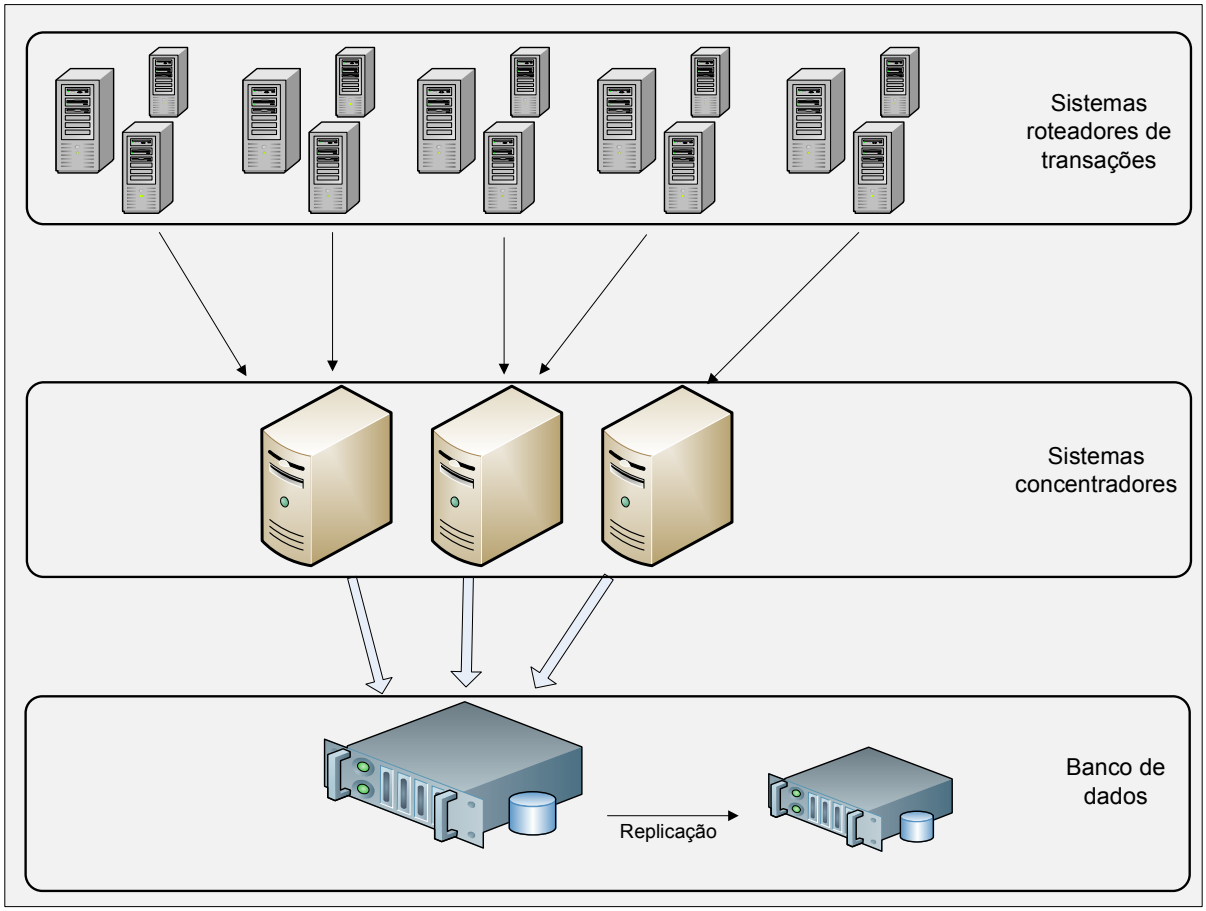

Figura 2.1 – Modelo atual da organização

Fonte: do autor.

O ambiente de produção atual do sistema concentrador são três servidores com as seguintes configurações:

- Memória: 8 Gb.
- Processador: Intel Xeon CPU 5160 @ 3.00 GHz.
- Sistema Operacional: Red Hat Enterprise Linux ES release 4 (Nahant Update 4).

2.2 Problemas

Os principais problemas do atual sistema concentrador se devem ao fato de sua arquitetura de desenvolvimento não estar preparada para suportar diversos tipos de transações eletrônicas com diferentes operadoras de cartões simultaneamente. Os problemas mais graves identificados são:

1. Problemas de comunicação (limitação do número de conexões suportadas);

2. Estouro de memória da JVM;

3. Lentidão do banco de dados;

4. Inflexibilidade na arquitetura de desenvolvimento e gerenciamento o que acaba causando sérios riscos em suas atualizações;

5. Dificuldade na troca de servidores em caso de indisponibilidade.

As próximas seções detalham estes problemas.

2.2.1 Problemas de comunicação

O sistema concentrador abre uma porta socket para cada tipo de transação eletrônica. Desta forma, requisição de compra usa uma porta, resposta de compra usa outra porta, confirmação de compra outra porta. Levando em consideração que atualmente ele processa aproximadamente 35 operadoras de cartões e para cada operadora existem em média 8 transações de diferentes tipos, o concentrador acaba abrindo um número considerável de

portas *socket's*. Além disto, o sistema chega a processar em horários de maior volume cerca de 40 transações por segundo.

Isto acaba gerando problema no gerenciamento do sistema o que leva ele a não conseguir mais processar e ficar indisponível. Neste caso, para que o sistema concentrador volte a funcionar é necessário reiniciar o mesmo.

2.2.2 Problemas de memória

O SC atual possui um grave problema de memória que ocorre de forma descontrolada em qualquer horário. Normalmente, acontece em horário de pico onde o seu processamento é mais exigido. Tal problema gera indisponibilidade do sistema, o que acaba automaticamente gerando também uma "dor de cabeça" aos clientes e usuários dos produtos que o SC processa. Com isto a imagem da empresa acaba ficando exposta de uma forma negativa.

Os problemas de memória se devem ao fato de limitações da própria versão dos softwares utilizados no desenvolvimento da aplicação. Com a estrutura atual não foi possível solucionar este problema. Então para contornar de forma paliativa, o sistema é reiniciado diariamente durante a madrugada, onde o volume de transações eletrônicas é mais baixo para evitar problemas mais graves durante o período em que o sistema é mais exigido. Com esta ação é possível minimizar o número de registros de problemas com as situações citadas.

2.2.3 Problemas de lentidão com o banco de dados

O problema de lentidão do SC com o banco de dados se dá ao fato do crescimento mau planejado da arquitetura da atual aplicação. Os clientes e produtos foram sendo desenvolvidos dentro da aplicação e inicialmente estava tudo funcionando corretamente. Porém com o aumento do volume transacional e obviamente o aumento de informações no banco de dados, com o passar do tempo as consultas às tais informações passaram a cada vez ficar mais demoradas, o que acaba impactando diretamente em uma transação eletrônica que ocorre em segundos.

Para tentar eliminar o problema de lentidão de acesso ao banco de dados, são mantidos diversos dados em memória. Dentre elas, informações administrativas que são utilizadas nas validações durante as transações eletrônicas e também as próprias transações. Esta ação acaba impactando no problema de “falta de memória” citado acima. Outro problema em manter os dados das transações em memória ocorre por não se tratar de uma aplicação com memória compartilhada. Então existe uma mesma aplicação sendo executada em três diferentes máquinas. Cada aplicação possui o seu gerenciamento de memória, então as transações que ocorreram no servidor 1, somente existem em memória nele mesmo. Então se a transação não for concluída com sucesso, pelas regras de negócio e resolução de pendência, esta situação acaba gerando problemas justamente porque os dados da transação não estão compartilhados em todos os servidores.

As estruturas das tabelas do banco de dados foram crescendo de forma desorganizada, o que acaba impactando quando o número de registros é alto. Então outra ação tomada para diminuir os riscos de problemas inesperados no sistema é o expurgo dos registros trimestralmente. Neste expurgo é removido do banco de dados de produção cerca de 90% dos registros. Tais registros são copiados para uma base de dados de backup.

Esta prática influência consideravelmente no tempo de resposta do sistema concentrador em suas tarefas. Levando em consideração que o tempo de resposta em média é em torno de 20 milissegundos. Porém, cada vez que é executado o expurgo dentro do SC, acaba gerando uma indisponibilidade de aproximadamente 30 minutos.

2.2.4 Problemas de inflexibilidade e rigidez

A inflexibilidade do sistema se dá ao fato de este não ter sido projetado para atender diversos clientes, mas mesmo assim estar atendendo. O sistema foi crescendo de acordo com as necessidades da organização, porém não foram levadas em consideração as boas práticas de desenvolvimento e principalmente a estrutura não modular do software.

O principal problema neste fato é a inflexibilidade na arquitetura do sistema de acordo com cada cliente, ou seja, em qualquer atualização, todos os clientes são impactados no mínimo em tempo de indisponibilidade, que este talvez seja o menos crítico.

Sendo um único sistema para diversos clientes, em qualquer problema inesperado ou manutenção planejada onde é necessário no mínimo reiniciar o sistema, acaba impactando todos os clientes que utilizam seus serviços.

O risco é muito alto nas atualizações pelo fato do sistema concentrador atender mais de 30 operadoras de cartões diferentes, porém utilizando serviços incomuns para todas, ou seja, uma pequena alteração pode impactar no ambiente transacional causando problemas que não são possíveis de mapeá-los.

2.2.5 Dificuldades na substituição de servidores

Ao acontecer uma falha de hardware que afete o SC, encontra-se uma grande dificuldade em disponibilizar novamente o ambiente ou até mesmo substituir o hardware com defeito por outro estável. Como mostrado na figura 1, os sistemas roteadores de transações comunicam-se diretamente com os sistemas concentradores. Isto acaba gerando problemas no momento em que qualquer um dos servidores do SC ficar indisponível. Pois este chaveamento de troca de endereço deve ser feito manualmente em cada máquina que executam os sistemas de roteadores das transações alterando seus apontamentos para os novos endereços.

Com a atual estrutura, não é possível ter este processo automatizado devido a limitações estruturais e regras de negócios do SC. Este é um problema antigo e sem solução, pois o impacto e risco de tentar resolver na atual versão do SC é muito alto.

3 TECNOLOGIAS ENVOLVIDAS

As tecnologias para o desenvolvimento deste projeto foram pesquisadas de acordo com a necessidade do sistema, pesquisou-se o que era mais interessante de acordo com as necessidades também da organização.

3.1 *JAVA* 1.5

Em 1991 iniciou-se o projeto na empresa *SUN Microsystems* de criação do ambiente JAVA, porém este ficou sem sucesso até 1995 quando foi incorporado aos *browsers* (navegadores) populares da época, quando realmente foi desenvolvido um ambiente para utilização em múltiplas plataformas. *JAVA* tornou-se uma linguagem de programação interpretada orientada a objetos. (HTML Staff, 2009).

Atualmente *JAVA* é um sucesso no desenvolvimento de softwares. *JAVA* possui como diferencial seu próprio interpretador, chamado de *JAVA Virtual Machine* (Máquina Virtual *JAVA*). *JAVA* permite que os códigos fontes sejam escritos em qualquer aplicativo independente do Sistema Operacional e executados em qualquer equipamento que possua a JVM interpretando os códigos fontes. (HTML Staff, 2009).

A versão 1.5 do *JAVA* é uma versão bastante consolidada no mercado e também homologada para ser utilizadas em conjunto com produtos da *Oracle.* Por isto não está sendo proposto utilizar alguma versão mais atual do *JAVA.* (Sun Microsystems, 2009)

Dentre as muitas vantagens do *JAVA* as interessantes para este projeto são:

Orientação a objetos: o que permite a herança e a reutilização de códigos diminuindo o tempo de desenvolvimento e manutenção do software.

Alto desempenho: utilizando seu código nativo e outros recursos de auto desempenho como *multithreading* e compilação *just-in-time* seu desempenho na execução de processos se torna superior a outras linguagens de programação conhecidas no mercado.

API: *Application Programming Interface* (ou Interface de Programação de Aplicativos) são blocos de códigos já prontos que podem ser facilmente encontrado e utilizado nas aplicações JAVA, o que facilita muito o desenvolvimento de novos *softwares*.

Outras características do JAVA são: possui um ambiente de desenvolvimento gratuito; multiplataforma; portabilidade; escalabilidade; confiabilidade; flexibilidade; desempenho; capacidade de reutilização; segurança; linguagem popular, sendo mais fácil à contratação de recursos (e-articles, 2009).

3.2 *Framework Spring*

Trata-se de um *framework open source* dividido em módulos que podem ser usados separadamente ou em conjunto conforme a necessidade do projeto. Tem como base os padrões de projeto inversão de controle e injeção de dependência. Facilita a modularização do mapeamento objeto-relacional quando integrado ao *Hibernate*, controlando a abertura e o fechamento de sessões. Sua arquitetura é baseada principalmente em arquivos de configuração XML. (Wikipédia, 2009).

Inversão de controle é uma das características do *framework Spring* onde o controle dos métodos desenvolvidos pelo usuário são gerenciados pelo *Spring*, não sendo invocados no código fonte da aplicação como em projetos ou linguagens que não utilizam a inversão de controle. Com isto a responsabilidade de gerenciamento das instâncias dos objetos fica sob a responsabilidade do *framework*. Desta forma o desenvolvedor invoca os métodos necessários, o *Spring* os gerencia e o fluxo retorna novamente para o desenvolvedor. (Imasters, 2009).

Algumas características que serão bastante utilizadas no projeto que são fornecidas pelo *Spring* são o gerenciamento do ciclo de vida dos objetos; gerenciamento do método de criação dos objetos; controle de transação; recursos de programaçao orientada a aspectos

(*A.O.P.*); disponibilização de Serviços Remotos; integrações com *EhCache, Hibernate, e Quartz*;

3.3 *Framework Hibernate*

Hibernate é um *framework* de persistência de dados. Um *framework* de mapeamento de objetos-relacional para objetos *JAVA*, facilitando a integração do modelo objeto da linguagem *JAVA* com o modelo relacional do banco de dados. Ao utilizar *hibernate*, não é necessário pensar em tabelas do banco de dados e sim em objetos, relacionando os próprios objetos de acordo com as necessidades. (Wikipédia, 2009).

Então sua principal característica é a facilidade na transformação dos objetos *JAVA* em tabelas de dados. Outra vantagem do *hibernate* é que utilizando este *framework* é possível eliminar muitos códigos de comando ao banco de dados, ou seja, os comandos mais básicos, já estão implementados e não é necessário ser descritos pelo desenvolvedor, somente os comandos mais específicos da aplicação onde exige um grau maior de complexibilidade na hora do desenvolvimento. (Linhares, 2009).

O *Hibernate* utiliza o *HQL (Hibernate Query Language)* que é o dialéto do *Hibernate* com o *SQL (Structured Query Language)*. Não é necessário utilizar somente *HQL* quando se utiliza *Hibernate*, também é possível utilizar *SQL* com *Hibernate*. A vantagem do *HQL* ao se utilizar *Hibernate* é que este possui uma linguagem totalmente orientada a objetos, onde é possivel utilizar herança, polimorfismo, encapsulamento entre outras caracteristicas da linguagem *JAVA.* (Hibernate.org, 2009).

Outra vantagem do *Hibernate* que está disponível é o uso de *cache* conforme a sua configuração que mantém o objeto em memória para uma reutilização onde os seus próximos acessos serão mais rápido que o primeiro. O *Hibernate* também possui consultas sob demanda, isto significa que os objetos são carregados conforme a necessidade do software. Este processo reduz o processamento do banco de dados e também o trafego de rede, pois evita a troca de informações desnecessárias. (Hibernate.org, 2009).

As principais características do *Hibernate* utilizadas no projeto são: gerenciamento do modelo de persistência; definições de entidades; definições de *cache*.

3.4 *Framework Apache Mina*

O *framework Apache MINA* (*A Multi-purpose Infrastructure for Network Applications*) serve para aplicações em rede, desenvolvido em *JAVA*, com um conjunto de *API* para capturar eventos assincronamente, facilitando o desenvolvimento de aplicações que requer conectividade, com uma alta performace e alta escalabilidade. Com o *MINA* é possível separar a lógica do protocolo de comunicação da lógica de conectividade.

É possível desenvolver com o *MINA* um servidor de protocolo específico, sem grandes dificuldades, já que este possui suas *API's* completas para o desenvolvimento. Ou seja, no desenvolvimento de um *software* é necessário se preocupar com a lógica de desenvolvimento e protocolo de comunicação e não com a forma que a comunicação irá acontecer. (Rigoni, 2008)

Testes de performance mostram o poder do seu processamento de conectividade.

3.5 *Framework Quartz*

O *framework Quartz* é um ótimo auxiliar na implementação de agendamento de tarefas, trabalha com conceito de *jobs* e *triggers*. Utilizado para notificar ou executar tarefas pré-agendadas em seus componentes. Um *framework* simples e de fácil configuração além de ter integração com outros sistemas. (JavaFree.org, 2009)

Quartz utiliza o conceito de *Cron Expressions*, que são utilizados na configuração do agendamento. *Cron Expressions* são *strings* de configuração que podem ser divididas em: mês, dia do mês, dias da semana, horas, minutos e segundos, com esta configuração é possível atingir todos os níveis de agendamento necessário. (Hotwork, 2009).

3.6 *Framework Ehcache*

Cache é um mecanismo auxiliar que contribui com o tempo de processamento de busca e acesso aos dados em determinados dispositivos, neste caso, o banco de dados, agindo de maneira onde uma vez que os dados são solicitados, tais dados são armazenados em memória *RAM* e a partir do segundo acesso aos mesmos dados, a busca não é executada no banco de dados, e sim na memória, assim atingindo uma performance superior em relação a buscas dos dados no banco de dados.

Resumindo, se os dados solicitados estão em memória, utiliza os dados já existentes, se não, busca no banco de dados e armazena-os em memória para uma utilização futura. Desta forma, o *framework EhCache* aumenta a performance do sistema.

O *Framework EhCache* é de simples utilização e a partir de arquivos de configurações é possível desfrutar deste aspecto. As configurações do *EhCache* são por regiões, ou seja, cada tipo de objeto possui a sua própria região e suas próprias configurações individuais, deixando uma ampla gama de customizações por objetos. Além disto o *EhCache* utiliza conceitos de *LRU (Last recently used), LFU (Least frequently used) e FIFO (Fist in first out)* (EhCache.org, 2009).

LRU (Last recently used) utiliza um mecanismo de datas de controle de acesso e através desta informação o objeto mais antigo é removido do *cache* de acordo com a configuração da região do objeto.

LFU (Least frequently used) um mecanismo que utiliza a freqüência de acesso ao objeto para remove-lo quando necessário, removendo sempre os menos acessados.

FIFO (Fist in first out) utiliza um mecanismo de fila, onde o primeiro objeto a entrar, também é o primeiro objeto a sair, quando necessário. (Pereira, 2009).

3.7 *Oracle RAC - Oracle Real Application Clusters*

A *Oracle* surgiu há quase 30 anos e atualmente é uma potência mundial no mercado de TI, conhecida e utilizada no mundo todo, presente em 98 das 100 empresas citadas na lista

"Fortune 100" (lista das 100 maiores empresas do mundo). Uma empresa com muita experiência no mercado mundial, bastante consolidada tornou-se a principal fornecedora de *softwares* para gerenciamento de informações independente do mundo. Um dos seus produtos de grande sucesso é o *Oracle Real Application Cluster*. (Oracle Brasil, 2005).

> "O relatório Forrester, elaborado pelo analista sênior Noel Yuhanna, avalia as demandas das empresas que operam ambientes de computação intensiva, bem como os benefícios do uso do Oracle Real Application Clusters para atender a esses requisitos específicos. De acordo com estudo, trata-se "da solução mais inovadora e confiável do mercado". Os clientes citam a sua capacidade de ampliar, adaptar e melhorar a performance, conforme os aplicativos empresariais vão se expandindo. Segundo a Oracle, mais de 5.400 são clientes que utilizam o produto" (Oracle, 2005).

O *Oracle Real Application Clusters* também conhecido como *Oracle RA*C fornece a funcionalidade de vários computadores utilizarem a mesma base de dados simultaneamente, sendo assim, fornece um cluster de banco de dados. O *Oracle RAC* foi desenvolvido para suportar alta disponibilidade, tolerância a falhas e escalabilidade. Então ao utilizar o *Oracle RAC* é possível criar a base de dados em *cluster*, com isto, obtenha-se alta disponibilidade, tolerância a falhas além do melhor gerenciamento da base de dados e outras vantagens da versão. Também com o *Oracle RAC*, é possível expandir a capacidade da base de dados adicionando quantos *hardwares* for necessário, ou seja, escalabilidade horizontal, sem precisar alterar a aplicação, com a opção de escalabilidade é possível reduzir custos com *hardwares* utilizando um número maior de servidores com valores mais acessíveis. (Olimpus, 2009).

4 SOLUÇÃO PROPOSTA

Este capítulo contém a solução proposta para um novo ambiente do SC. Inicialmente aborda a arquitetura de hardwares para o projeto explicando qual a forma proposta para buscar alta disponibilidade a nivel de *hardwares* e rede. Logo após, este capítulo aborda a arquitetura de *software* desenvolvida para atingir o máximo de disponibilidade.

Com a soma dos conjuntos da arquitetura de *hardware* e arquitetura do *software*, busca-se atingir indisponibilidade zero no SC perante o usuário final do cartão.

4.1 Arquitetura de Hardware

A solução proposta busca obter uma maior flexibilidade no SC de forma que seja possivel diminuir ao máximo o tempo de indisponibilidade do ambiente. Para obter melhores resultados, propõe-se aqui a entrada de um *hardware* Roteador *Cisco* entre os sistemas roteadores de transações e o SC. Além disto, o SC deverá ser desenvolvido em *cluster* utilizando memória compartilhada entre os servidores. O banco de dados também utilizará tecnologia de *cluster* para aumentar assim a disponibilidade do mesmo. A figura 4.1 ilustra tal ambiente.

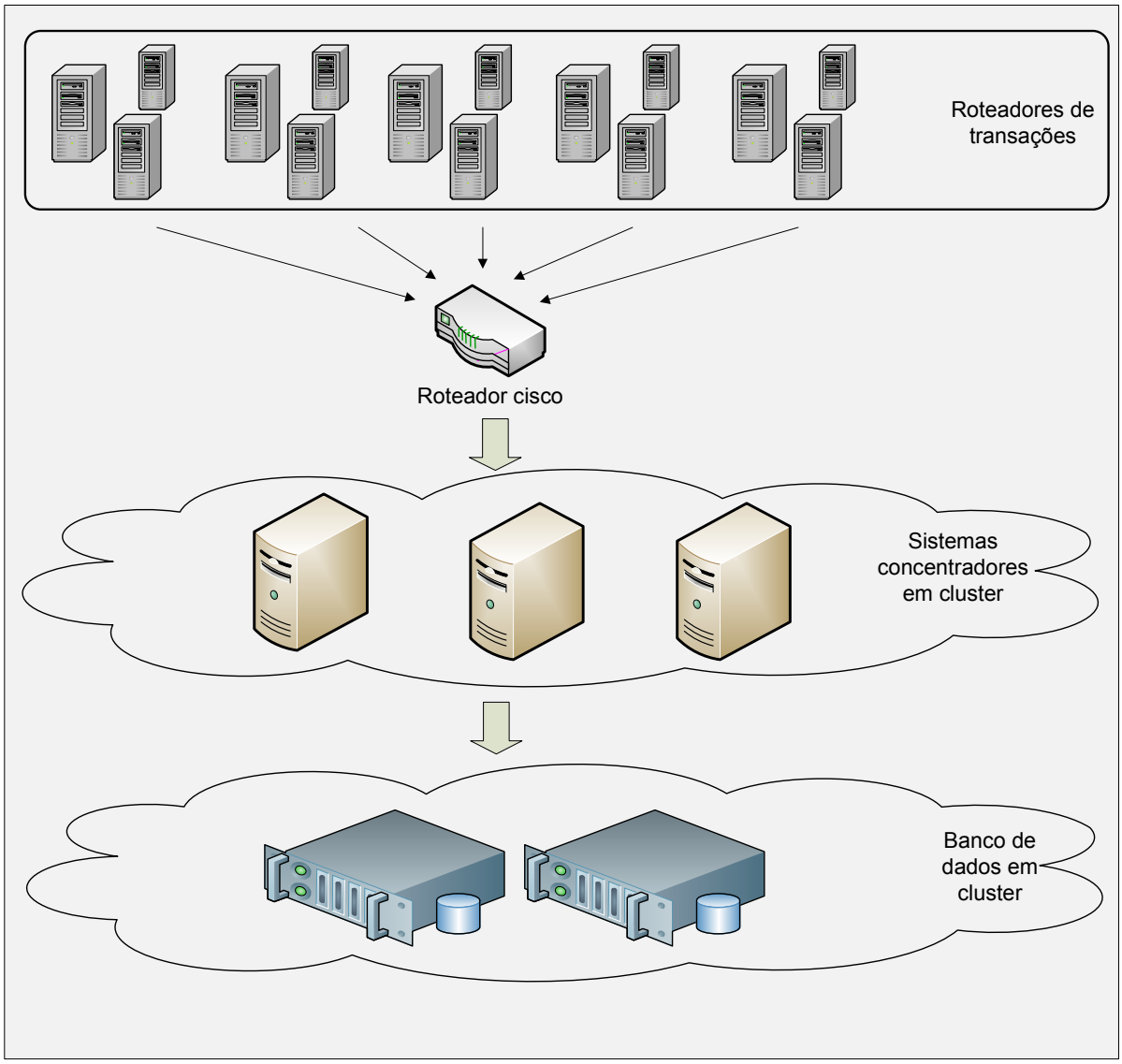

Figura 4.1 – Proposta de Arquitetura

Fonte: do autor.

Com a entrada de um roteador dinâmico da *Cisco* entre os sistemas de roteamento de transações e os sistemas concentradores a rede transacional fica mais segura e é possível eliminar a indisponibilidade caso ocorra algum problema em qualquer servidor dos sistemas concentradores, exceto se todos os servidores pararem de responder.

O maior ganho da utilização de um roteador dinâmico no ambiente transacional da organização se dá ao fato que o roteador tem a inteligência de mapear a melhor rota de rede diminuindo o tempo de entrega de pacotes e este também consegue identificar quando algum endereço IP não está disponível. Sendo assim, utilizando regras pré-definidas o roteador é

capaz de identificar quando um ou mais dos servidores dos sistemas concentradores não estão respondendo e automaticamente enviar os pacotes contendo as transações para o sistema concentrador que estiver funcionando perfeitamente.

Segundo Rodrigo Uchoa (2008), diretor de novos negócios da Cisco no Brasil, o ganho com redes inteligentes deve ser considerável atualmente nas empresas, como segue:

> A inteligência de rede é que vem diferenciando os roteadores e switches, avalia o diretor de novos negócios da Cisco Brasil, Rodrigo Uchôa. Com as tecnologias disponíveis, a rede consegue reconhecer as aplicações e os usuários e, a partir daí, diferenciar os pacotes de dados. Os ganhos são consideráveis no que diz respeito à produtividade.
>
> Uchôa cita como exemplo uma empresa do ramo financeiro que triplicou o número de transações por minuto, passando de 1,5 mil para mais de 4,5 mil, otimizando a aplicação responsável pelas operações.

Como a proposta do novo SC é ser mantido em um ambiente clusterizado utilizando a memória do sistema compartilhada entre os diversos servidores, não tem problema nenhum a transação financeira iniciar em um servidor e terminar em outro, caso o roteador envie parte de uma transação para cada servidor do SC. Existe também o ganho de ser possível fazer um balanceamento de carga entre os servidores.

Com o SC desenvolvido com tecnologias que disponham de *cluster*, o ganho de performance e confiabilidade aumentou bastante, visto que, com esta arquitetura é possível aumentar o processamento do sistema, apenas acrescentado *hardwares*, tantos quantos forem necessários, sem a necessidade de alterações sistêmicas.

A mesma proposta serve também para o banco de dados, onde será inicialmente utilizado 2 *hardwares* para suportar o banco *Oracle 10g* em *cluster*, o chamado *Oracle RAC*.

Como o sistema concentrador para funcionar perfeitamente depende apenas do banco de dados e dele mesmo, com a proposta de manter os dois em *cluster*, a segurança e confiabilidade estarão bastante presentes no modelo apresentado.

4.2 Arquitetura do Software

A figura 4.2 apresenta a arquitetura do sistema de forma macro:

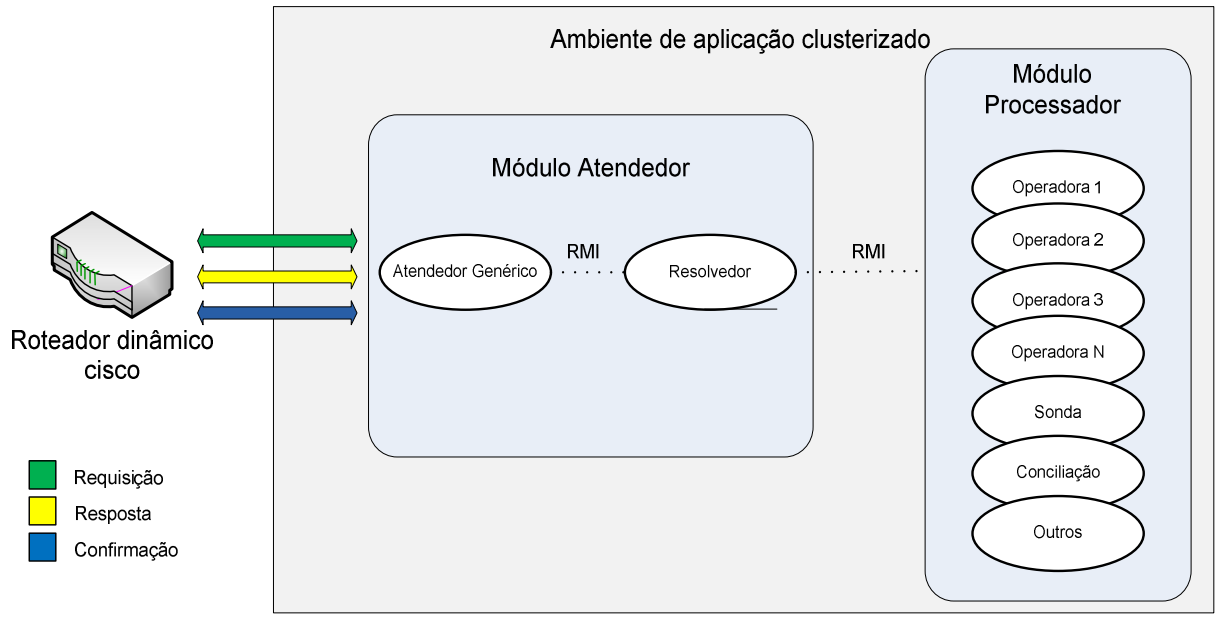

Figura 4.2 – Ambiente de aplicação clusterizada

Fonte: do autor.

Este sistema foi projetado de maneira modular para atingir flexibilidade nos momentos de alteração de ambiente de produção, seja este momento uma atualização ou implantação de um novo módulo ao projeto. Os módulos serão detalhados nas próximas seções.

4.2.1 Módulo Cache

O módulo *cache* é responsável por guardar os dados mais acessados do sistema, de forma que esses dados são armazenados em memória evitando assim o acesso ao banco de dados para consultá-los. O módulo *cache* deve ser o primeiro modo a ser iniciado no sistema, visto que, todos os outros módulos dependem dele para ter um funcionando normal. Então o

módulo *cache* armazena os dados mais acessados e também as transações no momento que elas vão sendo executadas.

O *cache* também é responsável por trocar informações entre os servidores que fazem parte do *cluster*. Através de um serviço *RMI* (Invocação Remota de Métodos) disponibilizado pelo módulo *cache* que os outros módulos conseguem acessar os dados em memória do sistema.

4.2.2 Módulo Resolvedor

Módulo responsável pelo acesso ao banco do SC. Deve ser o segundo módulo a ser iniciado no sistema, então o resolvedor se conecta no *cache* através da porta *RMI* que o *cache* disponibilizou.

Os dados utilizados no resolvedor, sempre são compartilhados com o *cache*, estes dois módulos precisam trabalhar sempre em conjunto, uma vez que o *cache* já possui o dado necessário, o resolvedor não acessa o banco de dados e utiliza a informação que está em memória, caso a informação ainda não estejam em memória, o resolvedor acessa o banco de dados e disponibiliza tal informação para o módulo *cache*.

Este módulo tem a responsabilidade de receber objetos *JAVA* do Atendedor Genérico e encaminhá-los para os sistemas de processamento da operadora específica em questão. Também é de sua responsabilidade a verificação e execução dos processos de resolução de pendência on-line.

Então a ordem do sistema é a seguinte: o *cache* é responsável pelo compartilhamento dos dados em memória. O módulo resolvedor alimenta e utiliza as informações do módulo *cache*. Por este motivo ao iniciar o sistema o módulo *cache* obrigatoriamente precisa ser o primeiro a ser iniciado, logo após o módulo resolvedor, que se conecta ao *cache*.

4.2.3 Módulo Processamento

Os processadores são os módulos de cada cliente do sistema, ou seja, com já citados o *cache* e o resolvedor, que sozinhos não fazem o processamento de uma transação, mas quando em conjunto se transformam em uma ferramenta muito efetiva na execução.

Os processadores são os módulos que contém as regras de negócio de cada cliente, ou seja, para todo e qualquer cliente deve ser criado um novo módulo processador buscando a exclusividade dentro do sistema para que no futuro, seja possível manipulá-lo individualmente, como alterações pontuais em regras de negócios por cliente diminuindo muito os possíveis impactos no projeto como um todo. Isto também facilita na hora de uma manutenção especifica, visto que apenas será atualizado o sub módulo necessário, não deixando nenhum outro também indisponível.

Então seguindo a ordem de funcionamento do sistema por completo, inicia-se o módulo *cache*, o módulo resolvedor e em seguida o módulo processador, que se conecta no módulo resolvedor para acesso aos dados, independentemente se os dados estão em memória ou não, os módulos processadores apenas precisam dos dados e quem tem que fornecer é o módulo resolvedor.

Desta forma, a arquitetura do sistema tornou-se mais modular. Com isto foi possível obter flexibilidade nas atualizações que sempre foi uma grande dificuldade com o sistema na versão antiga.

4.2.4 Módulo Atendedor Genérico

O módulo atendedor deve ser iniciado por último, logo após os processadores. O módulo atendedor assim como os módulos processadores, também deve se conectar no módulo resolvedor, uma vez que, o módulo atendedor é responsável por receber todas as transações do sistema, ou seja, o atendedor é a porta de entrada para iniciar o processamento.

O módulo atendedor tem a responsabilidade de receber o pacote socket e transformá-lo em objetos *JAVA*, após isto o atendedor encaminha o objeto *JAVA* para o resolvedor e fica aguardando sua resposta. Quando o resolvedor responde ao atendedor, este apenas transforma o objeto *JAVA* em um pacote e o responde via *socket*.

4.2.5 Módulo de Conciliação de Transações

Responsável por efetuar a geração de arquivos diários contendo todas as transações do dia anterior por operadora. Cada operadora possui seu *layout* próprio, cabe a este sistema identificar e processar no horário previamente configurado no sistema. Essas configurações são individuais de cada arquivo que o sistema deve gerar.

4.2.6 Módulo de Sonda

Responsável por gerenciar a resolução de pendências via sonda. Tal gerenciamento basicamente é o envio de transações aos estabelecimentos solicitando o status das transações que estão pendentes no sistema concentrador. A forma e intervalo de envio é configurado no módulo da sonda de operadora por operadora.

Este módulo é responsável apenas por solicitar uma resposta do estabelecimento, não é de sua responsabilidade processar a resposta do estabelecimento, isto fica a cargo do módulo processador, visto que a resposta seguirá o mesmo fluxo, iniciará pelo sistema atendedor, passará pelo resolvedor até chegar ao módulo da operadora especifico da resposta de sonda.

4.2.7 O Sistema como um todo

O módulo atendedor abre uma porta *socket* padrão *TCP/IP* e funciona como um servidor que fica aguardando as requisições. Uma vez que recebe uma requisição, este faz um

pequeno processamento que consiste em transformar o pacote recebido no *socket* para um objeto *JAVA*, uma vez feito isto, este chama o módulo resolvedor e aguarda sua resposta.

Quando o módulo resolvedor recebe uma requisição do módulo atendedor, este tem a responsabilidade de encaminhá-la ao processador ao qual pertence, ou seja, este deve identificar qual o cliente do sistema que deve processar a requisição recebida. Quando o resolvedor encaminha a requisição ao seu devido processamento, este também fica aguardando a resposta do processamento.

Quando a requisição chega ao módulo processador, tal módulo faz todo o seu processamento de acordo com as configurações e regras de negócio exclusivamente do cliente. Neste momento existe a integração entre os módulos para a utilização do *cache*, tanto para a utilização dos dados já existente quanto à inserção de novos dados.

Após o processamento do módulo processador, este responde ao módulo resolvedor que estava aguardando a sua resposta. Quando o módulo resolvedor recebe a resposta do módulo processador, o resolvedor também responde ao módulo atendedor, que estava aguardando a sua resposta.

Quando o módulo atendedor recebe a resposta do módulo resolvedor, este tem a responsabilidade de transformar o objeto *JAVA* recebido em pacote *socket* e responde-lo ao sistema que o solicitou tal processamento. Ou seja, o atendedor funciona como um servidor passivo.

A figura 4.3 a seguir mostra a forma que os módulos se conectam dentro do SC.

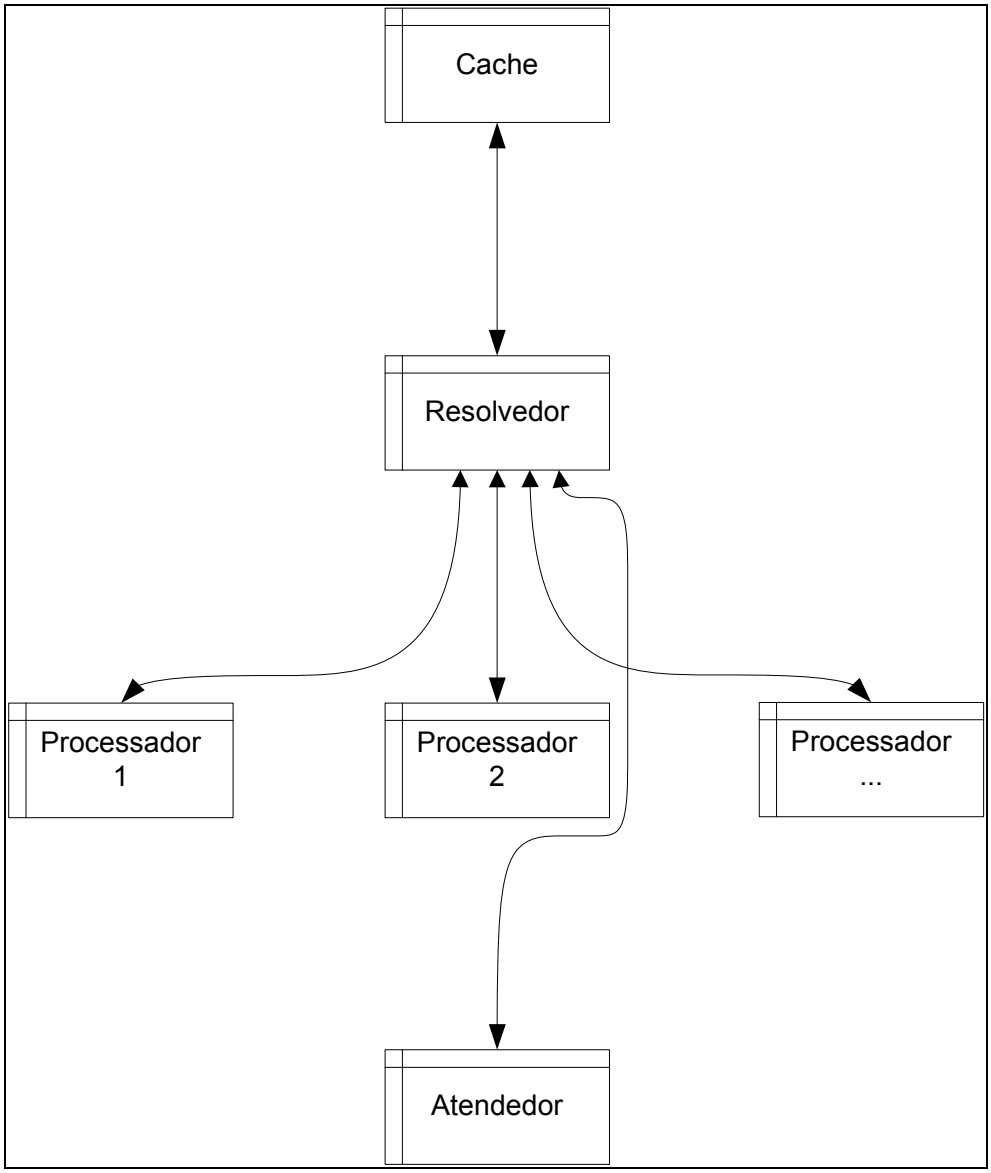

Figura 4.3 – Conexões dos módulos.

Fonte: do autor.

Os módulos citados neste capítulo estão englobados dentro do sistema concentrador na forma de cluster, conforme figura 4.4 com visão macro do ambiente.

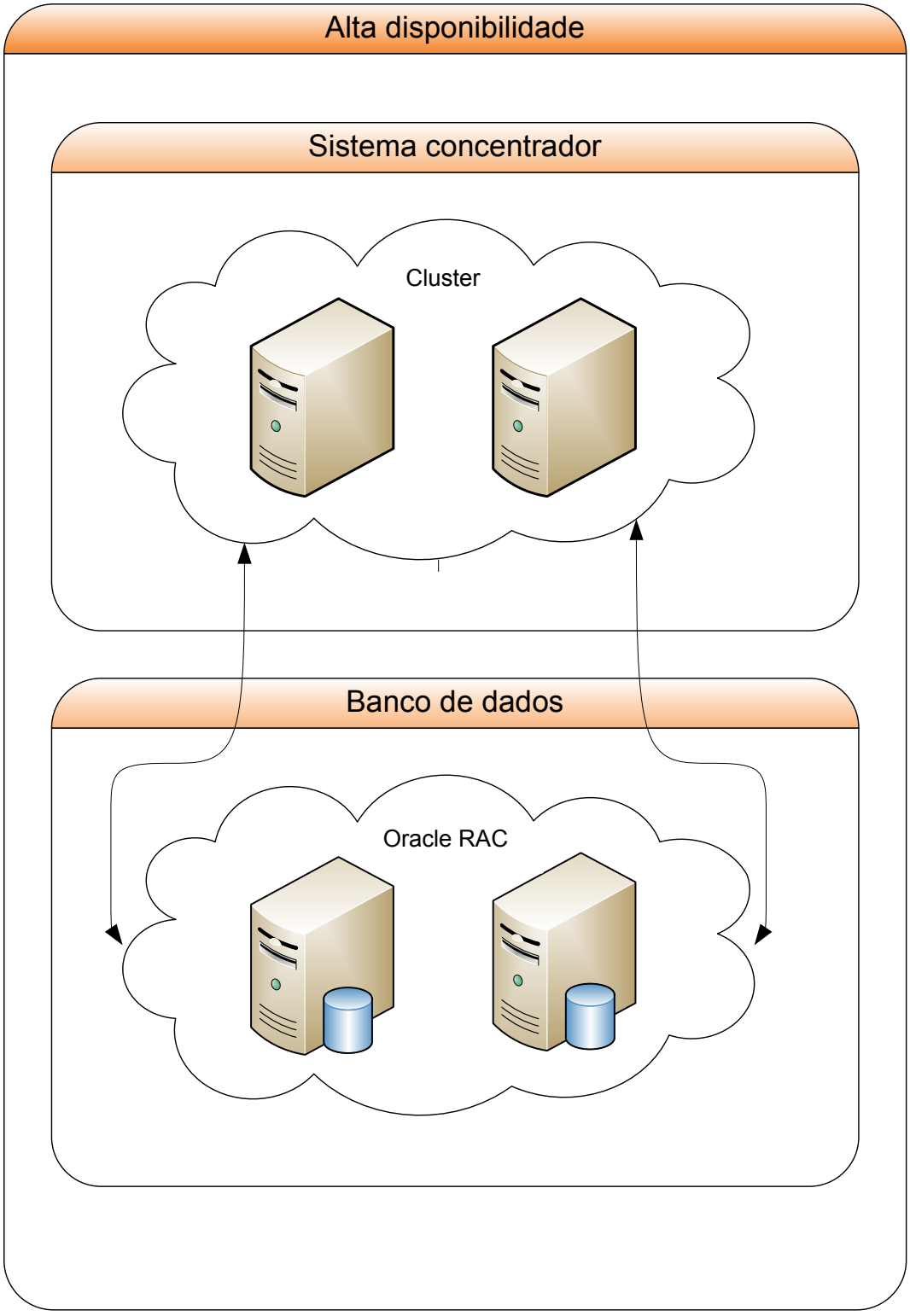

Figura 4.4 – Visão macro do ambiente do SC.

Fonte: do autor.

5 MODELAGEM DO SISTEMA

Este capítulo aborda a modelagem utilizada na fase de análise do sistema para o seu futuro desenvolvimento. A modelagem está da seguinte forma: um documento de visão geral do sistema e vários outros documentos, um para cada módulo planejado. Sendo assim, cada módulo possui a sua documentação específica.

5.1 Estrutura dos documentos de modelagem

Após pesquisas foi adotado um padrão de documentação que abrange todas as necessidades do projeto. Este padrão de documentação ficou adaptado para as necessidades do projeto da forma que segue. Para todos os módulos, foi criado um documento de especificação que tinha a sua estrutura baseada em capítulos e subcapítulos conforme estrutura apresentada no quadro 5.1

Capítulo	**Subcapítulos**
1 Introdução	1.1 Finalidade.
	1.2 Escopo.
	1.3 Definições e abreviaturas.
	1.4 Referências.
	1.5 Detalhamento do Sistema.

2 Espeficicação de requisitos	2.1 Requisitos funcionais.
	2.2 Requisitos não-funcionais.
3 Descrição dos *use-cases* e atores	3.1 *Use cases.*
	3.2 Atores.
4 Diagrama geral de *use-cases* do sistema	4.1 Diagrama geral de *use-cases* do sistema.
5 Detalhamento das *use-cases*	5.1 Detalhamento das *use-cases.*
6 Comportamento dinâmico	6.1 Diagramas de sequências.
7 Comportamento estático	7.1 Diagrama entidade-relacionamento.
8 Camadas e pacotes	8.1 Diagrama de Componentes.
9 Testes	9.1 Teste de classe.
	9.2 Testes de *stress.*
	9.3 Testes de funcionalidades.

Quadro 5.1 – Estrutura da documentação

Fonte: do autor.

Aqui serão apresentados os principais conteúdos e diagramas da documentação utilizada no projeto, abrangindo um trecho de cada módulo para o desenvolvimento.

Estes exemplos foram extraídos de forma parcial da análise do projeto, ou seja, aqui não contém todo o conteúdo do projeto, apenas trechos para exemplificar o conteúdo e estrutura do padrão de análise aqui citado.

5.2 Detalhamento do Sistema

Neste exemplo contém os dados do detalhamento do sistema da documentação realizada para a visão geral do sistema concentrador.

1.5 Detalhes do Sistema

1.1.1 Serviços do Sistema Concentrador

O Sistema Concentrador é responsável direto pelo atendimento das seguintes operações principais:

1. Transação de Compra (também chamada de 1.a perna);
2. Retorno da Transação de Compra (também chamada de 2.a perna);
3. Confirmação de Compra (também chamada de 3.a perna);

Além destas transações, que fazem parte do fluxo ótimo, o Sistema Concentrador também atende a outras operações de cancelamento e desfazimento:

4. Cancelamento de Transação de Compra (1.a perna do cancelamento);
5. Retorno do Cancelamento de Transação de Compra (2.a perna do cancelamento);
6. Confirmação do Cancelamento (também chamada de 3.a perna);
7. Desfazimento da Transação de Compra (uma perna apenas);

Sobre o desfazimento pode-se elencar as seguintes situações em que este acontece:

- POS não recebeu retorno da transação (problemas de queda de linha ou timeout, por exemplo) e então ele envia o desfazimento, ou seja, uma requisição de desfazimento de compra;
- Último NSU de uma transação ok é diferente daquele que está gravado no Sistema Concentrador, e então o sistema gera um desfazimento da transação, enviando inclusive para a operadora.

O Sistema Concentrador ainda provê serviços de consulta de saldo:

8. Verificação de Saldo (1.a perna);
9. Retorno de Verificação de Saldo (2.a perna);

E serviço de sondagem para a Operadora:

10. Sondagem da Operadora (1.a perna);

11. Retorno da Sondagem da Operadora (2.a perna);

Obs.:

- Mesmo fluxo da transação de compra. Precisa retornar status da transação, por isso as duas pernas.
- Primeiro verifica com Resolvedor de Pendências se transação está pendente. Se tiver pendente, apenas responde para a operadora este status. Se não está, acessa o Atendedor de Requisições que acessa o Módulo Processador para ver se a transação está confirmada ou desfeita.

O Sistema Concentrador também executa sondagens:

12. Sondagem do Concentrador (uma perna apenas);

Obs.: Periodicamente é feita sondagem, caso expire o tempo determinado, então a transação pendente é removida da memória;

1.1.2 Responsabilidades do Sistema Concentrador

O Sistema Concentrador é responsável:

- Pelo registro das transações na captura;
- Pela resolução de pendências;
- Pela interface com o Sistema de Autorização;
- Pela conciliação com Sistemas de Autorização externos;
- Pelas validações da captura;

5.3 Detalhamento dos *Use-cases*

Do módulo atendedor foi extraído o trecho do caso de uso de resolução de pendências on-line que se referência ao item 5.1 do padrão de modelagem aqui já citado.

Nome da Use Case	*UC Resolver Pendência on-line*
Descrição	*Identificar e resolver uma operação pendente on-line*
Requisitos Associados	*ERaF0001.2 – Resolver Pendências On-line*
Pré Condições	*Existir uma transação pendente a ser resolvida*
Pós Condições	***Inexistente***
Atores	*GatewayIn*

Fluxo Principal	
Ações Recebidas	**Ações Realizadas**
1. Os atores GatewayIn inicia a fluxo principal (ou fluxo ótimo).	
	2. O sistema Atendedor Genérico recebe a operação atual. Nesta operação contém também informações que identificam a ultima operação OK efetuado no mesmo terminal. *3. O sistema Atendedor Genérico envia o **terminal** e **estabelecimento** para o sistema Resolvedor de Pendências que retorna se existe alguma operação pendente.*
*4. Recebe uma resposta **positiva** do sistema Resolvedor de Pendências e dispara processo de resolução.*	
	5. Dispara um processo de resolução de pendência e em paralelo o processo da operação atual.

Fluxo Alternativo - Pendência do mesmo cartão atual	
Ações Recebidas	**Ações Realizadas**
1. Os atores GatewayIn iniciam o fluxo principal (ou fluxo ótimo).	

	2. O sistema Atendedor Genérico recebe a operação atual. Nesta operação contém também informações que identificam a ultima operação OK efetuado no mesmo terminal. *3. O sistema Atendedor Genérico envia o **terminal** e **estabelecimento** para o sistema Resolvedor de Pendências que retorna se existe alguma operação pendente.*
*4. Recebe uma resposta **positiva** do sistema Resolvedor de Pendências*	
	5. Dispara um processo de resolução de pendência.
6. Recebe a resposta que a operação de resolução de pendência foi concluída.	
	7. Dispara o processo da operação atual

Tabela 5.2 - Fluxo de Eventos da Use Case Resolver Pendência on-line

5.4 Diagrama de Sequência

Neste exemplo, foi extraído o diagrama de seqüência de uma transação eletrônica processada dentro do SC apresentada no quadro 5.1. Tal exemplo está no documento do módulo resolvedor no capítulo 6.1.

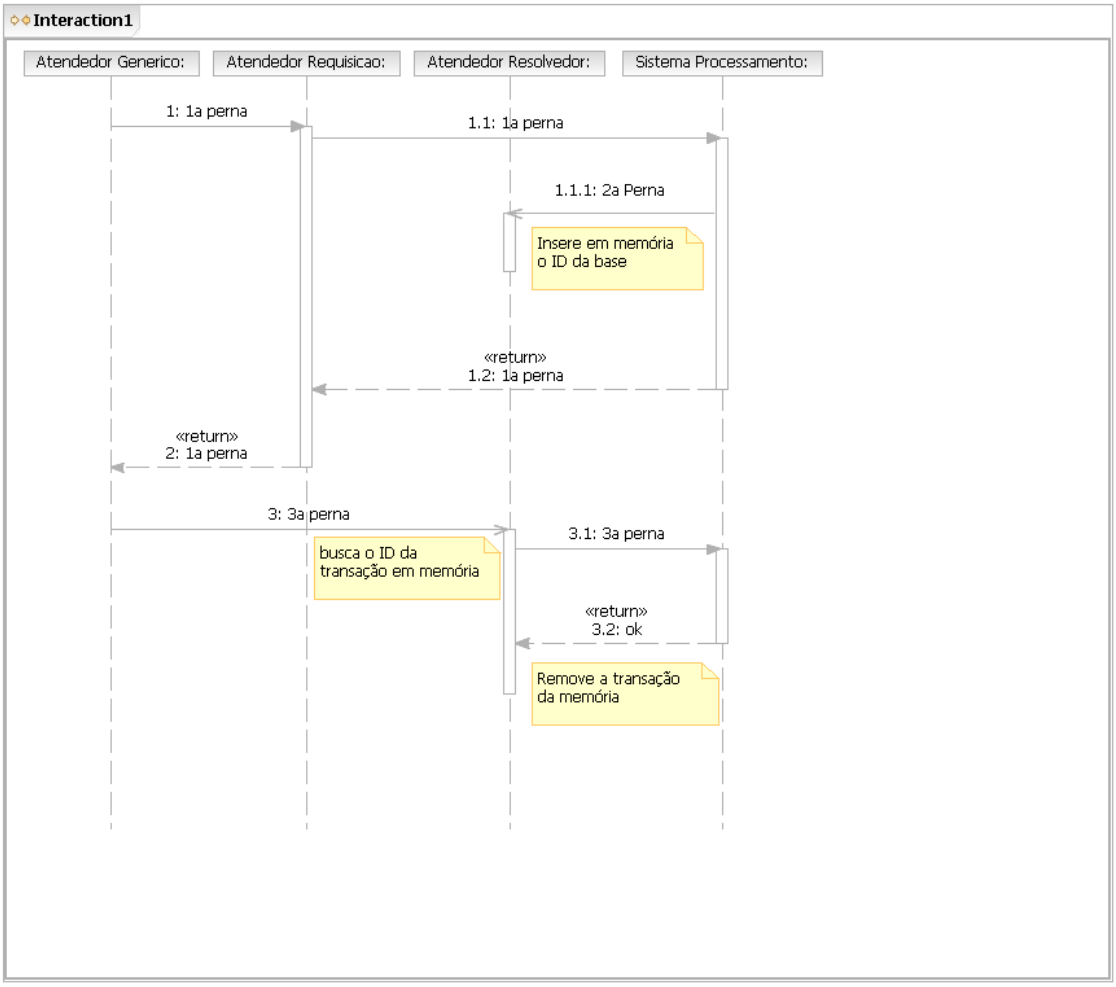

Figura 5.1 – Diagrama de seqüência de uma transação eletrônica

5.5 Formato alternativo de detalhamento de *Use-Cases*

Apesar de toda a documentação do projeto seguir o padrão apresentado neste capítulo, no momento da entrega da documentação a empresa parceira para o desenvolvimento do projeto, houve adaptações da documentação para se enquadrar no padrão de documentação utilizado pela empresa, então, além da documentação já criada, foi também desenvolvdo *Use cases* no seguinte modelo.

Requisito Origem:
4.1. 01-ProjetoNovoSistemaCaptura.doc
4.1.8. 05-SistemaProcessamento.doc - UC Transacionar Operação (Processar Requisição)

1 Caso de Uso: Processar Requisição

1.1 Descrição simplificada

Este UC é um serviço que tem a responsabilidade de receber uma operação (1.a perna) do componente AtendedorRequisição, processar, autorizar junto a operadora (interna ou externa) e encaminhar a resposta.

1.2 Atores envolvidos

- Gateway In;
- Gateway Out;
- Aplicação Get (TEF Discado);

1.3 Evento de Início

- Recebimento de transação (1.a perna) de requisição do Atendedor Requisição;

2 Pré-condições

- Não se aplica;

3 Pós-condições

- Devolver retorno (2.a perna);

4 Fluxo de Eventos

4.1 Cenário Principal - Processar Requisição

1. Ator envia transação para aplicação;
2. Componente AtendedorRequisição inicia o fluxo principal;
3. Módulo de Processamento recebe a operação;
4. Módulo de Processamento carrega a operadora respectiva à transação;
5. Componente da operadora gera o NSU SisCap, com as seguintes regras:
    1. Gera-se um NSU por operadora, iniciando a partir do número 1;
    2. Este código deve ser reiniciado na primeira transação do dia;
6. Componente da operadora valida o estabelecimento verificando:
    1. Se estabelecimento não é nulo;
    2. Se estabelecimento está no buffer; se não estiver pesquisa no DB;
7. Com o estabelecimento recuperado valida:
    1. Se estabelecimento está ativo;
    2. Se estabelecimento está credenciado para a operadora;
8. Componente valida terminal verificando:
    1. Se terminal não é nulo;
    2. Se estabelecimento contém o terminal;
9. Faz o “DePara” entre o código do estabelecimento na GetNet e o código do estabelecimento na operadora.
10. Registra a operação no DB (1.ª perna).
11. Comunica com o autorizador.
12. O Sistema Processamento recebe a resposta do autorizador.
13. O Sistema Processamento altera o status da operação.
14. Monta e encaminha a resposta ao Sistema Atendedor Requisição.
15. Monta e encaminha a chave [1] para o Sistema Resolvedor de Pendência.
16. Por fim, responde ao Sistema Autorizador (retorno da 1.a perna);
17. Fim do cenário.

4.1.1 Fluxos alternativos

1. Ator envia uma resolução de pendência.
    a. Realizar cenário 1, a partir do item 10 (Registra a operação no DB).
2. Ator envia uma resposta de sonda (3.a perna).

---

[1] Chave do buffer de transações pendentes (*Generic Key*): Código do Estabelecimento, Número do Terminal, Data da Captura, NSU da Captura, e Código da Operadora

a. Realizar cenário 1, a partir do item 10 (Registra a operação no DB).

4.1.2 Exceções

1. Validação de alguma, ou todas, das regras falham.
   a. Sistema apresenta mensagem de erro correspondente, se possível indicando a causa da falha;
   b. Sistema não persiste os dados na base até que o ator tenha corrigido a causa da falha e envie solicitação novamente;

4.1.3 Exceções

Não aplicável.

5 Requisitos Especiais

Não aplicável.

6 Requisitos Relacionados

05-SistemaProcessamento.doc - ERaF0001.1 – Serviço de Registro dos Sistemas

05-SistemaProcessamento.doc - ERaF0001.2 – Transacionar Operação

7 Leiaute da Tela

Não aplicável.

8 Especificações técnicas

Ver documentos:

01-ProjetoNovoSistemaCaptura.doc

05-SistemaProcessamento.doc

5.6 Modelo do diagrama entidade-relacional

5.6.1 Proposta de base de dados para armazenamento de transações

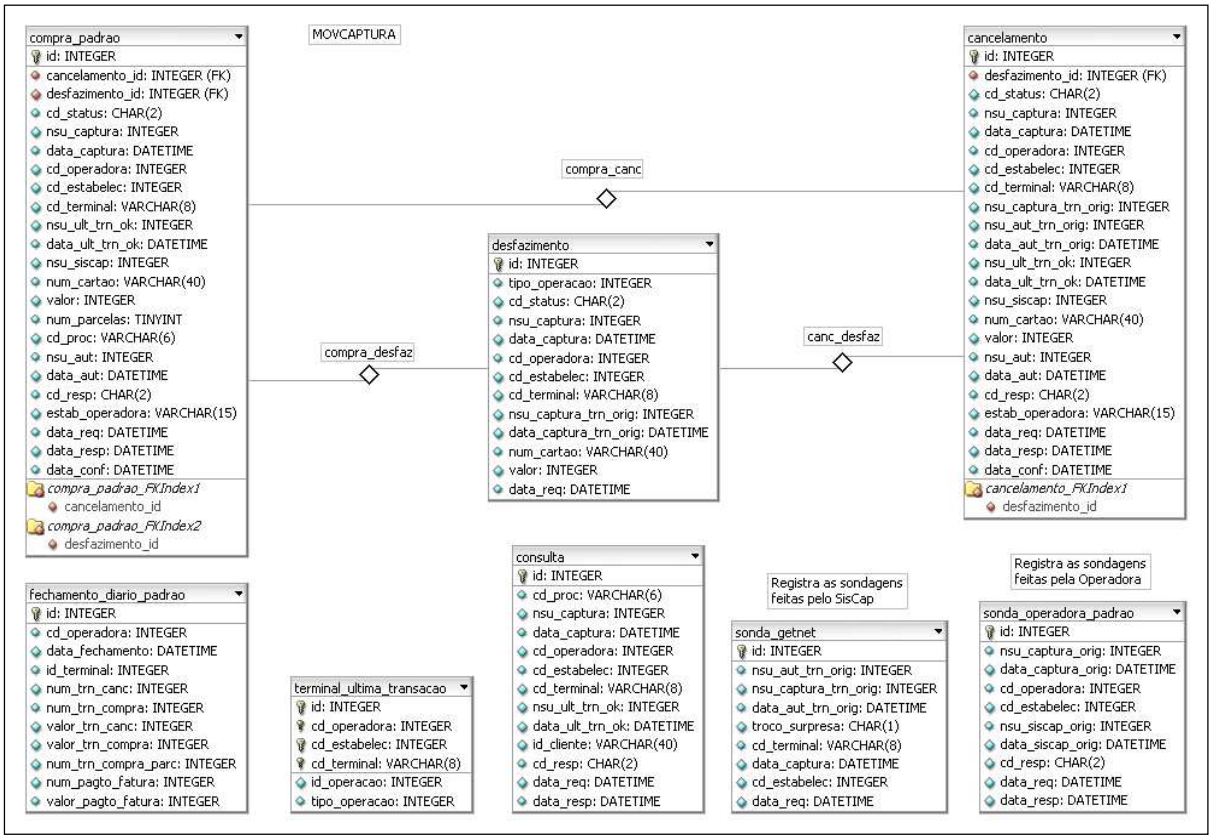

Figura 5.2 – Proposta de base de dados para as transações

Fonte: do autor.

Dicionário de dados

Nome da Tabela:		**COMPRA_PADRAO**	
Nro.	**Nome do Campo**	**Tipo**	**Descrição**
1	Id	INTEGER(PK)	Código identificador da transação
2	cancelamento_id	INTEGER(FK)	FK para a tabela cancelamento (relação 0..1)

3	desfazimento_id	INTEGER(FK)	FK para a tabela desfazimento (relação 0..1)
4	cd_status	CHAR(2)	Código do status atual da transação
5	nsu_captura	INTEGER	NSU do ponto de venda da captura atual
6	data_captura	DATETIME	Data e hora do ponto de venda da captura atual
7	cd_operadora	INTEGER	Código da operadora
8	cd_estabelec	INTEGER	Código do estabelecimento
9	cd_terminal	VARCHAR(8)	Código do terminal
10	nsu_ult_trn_ok	INTEGER	NSU da última transação ok
11	data_ult_trn_ok	DATETIME	Data e hora da última transação ok
12	nsu_siscap	INTEGER	NSU gerado pelo Sistema de Captura
13	num_cartao	VARCHAR(40)	Número do cartão
14	Valor	INTEGER	Valor da transação (valor inteiro, sem formatação)
15	num_parcelas	TINYINT	Número de parcelas (1 = à vista, utilizado p/ EDI)
16	cd_proc	VARCHAR(6)	Código do processamento (utilizado p/ EDI)
17	nsu_aut	INTEGER	NSU gerado pelo Sistema de Autorização
18	data_aut	DATETIME	Data e hora da autorização
19	cd_resp	CHAR(2)	Código da resposta da transação
20	estab_operadora	VARCHAR(15)	Código “DePara” entre o código do concentrador e o autorizador
21	data_req	DATETIME	Data e hora do recebimento da requisição (SC - 1.a perna)
22	data_resp	DATETIME	Data e hora da devolução da resposta (SC - 2.a perna)
23	data_conf	DATETIME	Data e hora do recebimento da confirmação (SC - 3.a perna)

Quadro 5.2 – Descrição da tabela compra_padrao

Fonte: do autor.

Nome da Tabela:		**CANCELAMENTO**	
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	id	INTEGER(PK)	Código identificador da transação
2	desfazimento_id	INTEGER(FK)	FK para a tabela desfazimento (relação 0..1)
3	cd_status	CHAR(2)	Código do status atual da transação
4	nsu_captura	INTEGER	NSU do ponto de venda da captura atual
5	data_captura	DATETIME	Data e hora do ponto de venda da captura atual
6	cd_operadora	INTEGER	Código da operadora
7	cd_estabelec	INTEGER	Código do estabelecimento
8	cd_terminal	VARCHAR(8)	Código do terminal
9	nsu_captura_trn_orig	INTEGER	NSU da Captura da transação original
10	nsu_aut_trn_orig	INTEGER	NSU da Autorização da transação original
11	data_aut_trn_orig	DATETIME	Data e Hora da Autorização da transação original
12	nsu_ult_trn_ok	INTEGER	NSU da última transação ok
13	data_ult_trn_ok	DATETIME	Data e hora da última transação ok
14	nsu_siscap	INTEGER	NSU do Sistema de Captura
15	num_cartao	VARCHAR(40)	Número do Cartão
16	Valor	INTEGER	Valor da transação (valor inteiro, sem formatação)
17	nsu_aut	INTEGER	NSU gerado pelo Sistema de Autorização
18	data_aut	DATETIME	Data e hora da autorização
19	cd_resp	CHAR(2)	Código da resposta da transação
20	estab_operadora	VARCHAR(15)	Código “DePara” entre o código Concentrador e o da Autorizador
21	data_req	DATETIME	Data e hora do recebimento da requisição (SC - 1.a

			perna)
22	data_resp	DATETIME	Data e hora da devolução da resposta (SC - 2.a perna)
23	data_conf	DATETIME	Data e hora do recebimento da confirmação (SC - 3.a perna)

Quadro 5.3 – Descrição da tabela cancelamento

Fonte: do autor.

Nome da Tabela:		**DESFAZIMENTO**	
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	Id	INTEGER(PK)	Código identificador da transação
2	tipo_operacao	INTEGER	Identificador do tipo de operação (Compra = 0 ou Cancelamento = 1)
3	cd_status	CHAR(2)	Código do status atual da transação
4	nsu_captura	INTEGER	NSU do ponto de venda da captura atual
5	data_captura	DATETIME	Data e hora do ponto de venda da captura atual
6	cd_operadora	INTEGER	Código da operadora
7	cd_estabelec	INTEGER	Código do estabelecimento
8	cd_terminal	VARCHAR(8)	Código do terminal
9	nsu_captura_trn_orig	INTEGER	NSU da captura da transação original
10	data_captura_trn_orig	DATETIME	Data da captura da transação original
11	num_cartao	VARCHAR(40)	Número do Cartão
12	Valor	INTEGER	Valor da transação (valor inteiro, sem formatação)
13	data_req	DATETIME	Data e hora do recebimento da requisição

Quadro 5.4 – Descrição da tabela desfazimento

Fonte: do autor.

Nome da Tabela:	**CONSULTA**		
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	Id	INTEGER(PK)	Código identificador da transação
2	cd_proc	VARCHAR(6)	Código de processamento
3	nsu_captura	INTEGER	NSU do ponto de venda da captura atual
4	data_captura	DATETIME	Data e hora do ponto de venda da captura atual
5	cd_operadora	INTEGER	Código da operadora
6	cd_estabelec	INTEGER	Código do estabelecimento
7	cd_terminal	VARCHAR(8)	Código do terminal
8	nsu_ult_trn_ok	INTEGER	NSU da última transação ok
9	data_ult_trn_ok	DATETIME	Data e hora da última transação ok
10	id_cliente	VARCHAR(40)	Identificação do cliente (CPF, caso Serasa p.ex. ou número do cartão)
11	data_req	DATETIME	Data e hora do recebimento da requisição

Quadro 5.5 – Descrição da tabela consulta

Fonte: do autor.

Nome da Tabela:	**SONDA_OPERADORA_PADRAO**		
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	id	INTEGER(PK)	Código identificador da transação
2	nsu_captura_orig	INTEGER	NSU da captura original
3	data_captura_orig	DATETIME	Data e hora da captura original
4	cd_operadora	INTEGER	Código da operadora
5	cd_estabelec	INTEGER	Código do estabelecimento
6	nsu_siscap_orig	INTEGER	NSU do Sistema de Captura original
7	data_siscap_orig	DATETIME	Data e hora do Sistema de

			Captura original
8	data_req	DATETIME	Data e hora do recebimento da requisição
9	cd_resp	CHAR(2)	Código da resposta da transação

Quadro 5.6 – Descrição da tabela sonda_operadora_padrao

Fonte: do autor.

Nome da Tabela:	**SONDA_CAPTURA**		
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	Id	INTEGER(PK)	Código identificador da transação
2	nsu_aut_trn_orig	INTEGER	NSU autorizador da transação de origem
3	nsu_captura_trn_orig	INTEGER	NSU da captura da transação original
4	data_aut_trn_orig	DATETIME	Data e Hora da Autorização da transação original
5	troco_surpresa	CHAR(1)	Flag para o Sistema Troco Surpresa
6	cd_terminal	VARCHAR(8)	Código do terminal
7	data_captura	DATETIME	Data e hora do ponto de venda da captura atual
8	cd_estabelec	INTEGER	Código do estabelecimento
9	data_req	DATETIME	Data e hora do recebimento da requisição

Quadro 5.7 – Descrição da tabela sonda_captura

Fonte: do autor.

Nome da Tabela:		**TERMINAL_ULTIMA_TRANSACAO**	
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	id	INTEGER(PK)	Código identificador da transação
2	cd_operadora	INTEGER(PK)	Código da operadora

3	cd_estabelec	INTEGER(PK)	Código do estabelecimento
4	cd_terminal	VARCHAR(8)(PK)	Código do terminal
5	id_operacao	INTEGER	Código identificador da operação *
6	tipo_operacao	INTEGER	Identificador do tipo de operação (Compra = 0 ou Cancelamento = 1)

Quadro 5.8 – Descrição da tabela terminal_ultima_transacao

Fonte: do autor.

* chave controlada via sistema para identificar a compra ou cancelamento (Não é FK para ser mais rápida a inserção)

Nome da Tabela:	**FECHAMENTO_DIARIO_PADRAO**		
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	Id	INTEGER(PK)	Código identificador da transação
2	cd_operadora	INTEGER	Código da operadora
3	data_fechamento	DATETIME	Data do fechamento
4	id_terminal	INTEGER	Código identificador do terminal
5	num_trn_canc	INTEGER	Número transações cancelamento
6	num_trn_compra	INTEGER	Número transações compra
7	valor_trn_canc	INTEGER	Valor transações cancelamento
8	valor_trn_compra	INTEGER	Valor transações compra
9	num_trn_compra_parc	INTEGER	Número transações compra parcelada
10	num_pagto_fatura	INTEGER	Número pagamento fatura
11	valor_pagto_fatura	INTEGER	Valor pagamento fatura

Quadro 5.9 – Descrição da tabela fechamento_diario_padrao

Fonte: do autor.

5.6.2 Proposta de base de dados para consultas administrativas

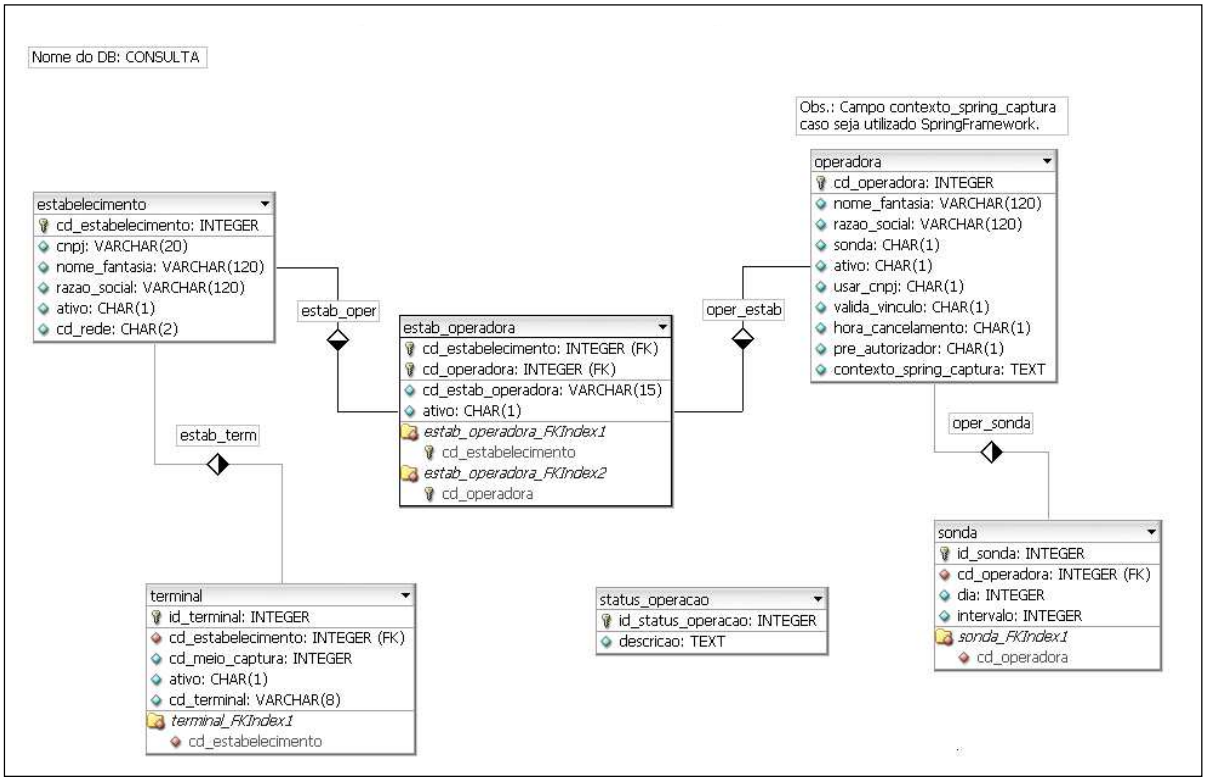

Figura 5.3 – Proposta de base de dados de consultas

Fonte: do autor.

Dicionário de dados

Nome da Tabela:		**ESTABELECIMENTO**	
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	cd_estabelecimento	INTEGER(PK)	Código do Estabelecimento
2	Cnpj	VARCHAR(20)	CNPJ
3	nome_fantasia	VARCHAR(120)	Nome fantasia
4	razao_social	VARCHAR(120)	Razão social
5	Ativo	CHAR(1)	flag

Quadro 5.10 – Descrição da tabela estabelecimento

Fonte: do autor.

Nome da Tabela:	**ESTAB_OPERADORA**		
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	cd_operadora	INTEGER(FK)	Código da Operadora
2	cd_estabelecimento	INTEGER(FK)	Código do Estabelecimento
3	cd_estab_operadora	VARCHAR(15)	Código do Estabelecimento na Operadora
4	ativo	CHAR(1)	flag para ativo ou inativo

Quadro 5.11 – Descrição da tabela estab_operadora

Fonte: do autor.

Nome da Tabela:	**OPERADORA**		
Núm	**Nome do Campo**	**Tipo**	**Descrição**
1	cd_operadora	INTEGER(PK)	Código da Operadora
2	nome_fantasia	VARCHAR(120)	Nome fantasia
3	razao_social	VARCHAR(120)	Razão social
4	conciliacao	CHAR(1)	flag usado na Captura Padrão
5	ativo	CHAR(1)	flag para ativo ou inativo
6	usar_cnpj	CHAR(1)	Se utiliza CNPJ na transação
7	valida_vinculo	CHAR(1)	flag para validar é permitido ou não cancelar mais de uma vez a transação (true = cancelar uma única vez)
8	hora_cancelamento	CHAR(1)	flag para validar a hora no cancelamento
9	pre_autorizador	CHAR(1)	flag para chamar o pré autorizador...
10	contexto_spring_captura	TEXT	Caso utilizado SpringFramework

Quadro 5.12 – Descrição da tabela operadora

Fonte: do autor.

Nome da Tabela:	**SONDA**		
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	id_sonda	INTEGER(PK)	Id da sonda
2	cd_operadora	INTEGER(FK)	Código da Operadora (FK para tab. Operadora)
3	dia	INTEGER	Dia da conciliação (Ordinal)
4	intervalo	INTEGER	Intervalo de tempo entre as sondagens (em minutos)

Quadro 5.13 – Descrição da tabela sonda

Fonte: do autor.

Nome da Tabela:	**TERMINAL**		
Núm.	**Nome do Campo**	**Tipo**	**Descrição**
1	id_terminal	INTEGER(PK)	Id do terminal
2	cod_estabelecimento	INTEGER(FK)	Código do Estabelecimento
3	cod_meio_captura	INTEGER	Código do meio de captura
4	Ativo	CHAR(1)	flag para ativo ou inativo
5	cod_terminal	VARCHAR(8)	Código do terminal

Quadro 5.14 – Descrição da tabela terminal

Fonte: do autor.

6 IMPLANTAÇÃO DO PROJETO

6.1 Desenvolvimento do projeto

Depois de efetuada a análise do sistema, iniciou-se o desenvolvimento do mesmo. Tal desenvolvimento foi efetuado por uma empresa externa que se reportava e seguia as regras de desenvolvimento. O andamento do desenvolvimento foi monitorado diariamente.

O sistema foi desenvolvido conforme as tecnologias já citadas e nos testes todos os objetivos foram atingidos, superando as expectativas. Foram desenvolvidos diversos processos automatizados para testar o ambiente transacional onde foi processada mais de 500 transações por segundo durante horas seguidas.

Foram 18 máquinas utilizando sistema operacional Windows configurados para abrir 60.000 portas *sockets* enviando transações para dois servidores com as configurações mapeadas acima e os resultados foram bastante satisfatórios e acima do esperado.

6.2 Projeto Piloto

Primeiramente o projeto foi implantado com status de piloto, já na arquitetura proposta, ou seja, 2 máquinas em *cluster* com o *Oracle Application Server* versão 10.1.1.3 como servidor de aplicações e mais 2 máquinas para banco de dados utilizando *Oracle RAC* versão 10G.

Já estavam desenvolvido os seguintes módulos, módulo *cache*, módulo resolvedor, 2 módulos de processamento e o módulo atendedor. Durante 4 meses o sistema ficou

processando apenas uma operadora efetuando em média 10 mil transações dia. Neste período o sistema não apresentou nenhuma falha grave, apenas foram efetuados ajustes finos no sistema e também no ambiente.

Com o sucesso do projeto piloto, foi desenvolvido outro módulo processador para processar uma operadora cuja as transações são capturadas através de um aparelho celular. Como o sistema está bastante modulado e flexível, não foi difícil colocar mais esta funcionalidade dentro do sistema, visto que foi necessário apenas desenvolver o módulo processamento da mesma, pois os módulos auxiliares (*cache*, resolvedor e atendedor) já estão prontos, ainda como piloto. O volume de transações não aumentou muito e o sistema continuou funcionando perfeitamente.

Seguindo na mesma linha, foi incluída uma operadora no módulo aqui chamado de processador padrão, tal módulo já estava desenvolvido e foi apenas cadastrar o cliente no banco de dados e começar a monitorar o comportamento do sistema. Os resultados foram ótimos e o sistema passou a efetuar em média 18 mil transações dia.

Com o sistema processando um volume maior de transações, foi avaliado o seu comportamento por mais 3 meses. Com o sucesso do projeto piloto, este foi aceito para entrar com status de produção.

6.3 Produção

Após a fase de projeto piloto, foi implementada uma operadora cujo o volume transacional esperado era em torno de 100 mil transações dia. Foi efetuada a análise de implementação da operadora dentro do novo sistema concentrador, visto que, este cliente não utilizaria o módulo de captura padrão. Porém os outros módulos seriam reutilizados para o desenvolvimento. Após a fase de desenvolvimento concluída, o projeto foi homologado durante 21 dias e entrou em produção.

Durante há primeira semana, a operadora estava em status de piloto, isto significa que, apenas alguns estabelecimentos estavam aptos a transacionar tal cliente. Em torno de 50 estabelecimentos que possuíam meio de captura liberado a transacionar. Todos os resultados

obtidos foram satisfatórios, então foram liberados mais 1000 estabelecimentos. Os resultados continuaram satisfatórios.

Tudo parecia perfeito, então liberou-se todos os estabelecimentos do cliente para transacionar, aproximadamente 5000 estabelecimentos. O sistema passou a processar em média 50 mil transações por dia, porém o volume transacional aumentava dia após dia. Com isto, iniciou um ciclo de problemas até então desconhecidos causando indisponibilidade para o cliente final, ou seja, o usuário do cartão.

O sistema simplesmente parava de funcionar sem nenhum motivo, cada queda do sistema causava uma indisponibilidade de aproximadamente 10 minutos. Conforme o volume de transação aumentava, diminuía o tempo que o sistema ficava ativo sem problemas, chegando ao ponto em que ficava em média 25 horas no ar.

Após uma analise dos dados foi verificado que processando em média 180 mil transações dia, o sistema ficaria aproximadamente 24 horas no ar. Isto gerou muitos problemas internos na organização onde se cogitou inclusive uma troca de tecnologia.

O sistema não tinha nenhum problema explícito, parava de processar *logando* apenas que a conexão *RMI* foi recusada entre os *conteiners* do servidor de aplicação. Porém monitorando o número de portas que estavam abertas durante uma queda do sistema, verificou-se que este número era bastante baixo, normalmente em torno de 2000 portas. Verificou-se também, que 90% destas portas estavam com status de "*time_wait*", que significa que o sistema operacional está aguardando para fechá-la. Além disto, 95% das portas em *time_wait* eram portas gerenciadas pelo produto *Oracle Application Server*, ou seja, não era possível interferir neste processo.

Iniciou-se uma investigação dos problemas referente ao número de portas, pois este era o único caminho que o sistema concentrador nos indicava ao *logar* conexão recusada. Por outro lado, o sistema operacional pode abrir até 64.512 portas, então as informações eram conflitantes. Pensando que o problema era nas conexões entre os módulos do sistema e/ou entre os dois servidores de aplicações, então se passou a utilizar apenas uma máquina para processar as transações, ou seja, o *cluster* foi desativado.

Após a desativação do *cluster*, o sistema passou a sobreviver mais tempo ativo. Porém, o problema não tinha sido resolvodo, apenas mascarado e mesmo assim o sistema

continuava a cair, agora com uma freqüência menor e não calculável, ou seja, às vezes ficava 100 horas ativo e outras 40 horas.

Com a intenção de sanar o problema, foi montado um ambiente contendo o sistema concentrador igual ao de produção e vários testes foram reproduzidos. Porém o problema não ocorreu em nenhum momento. Mesmo ao reprocessar os *logs* do ambiente produtivo o sistema se comportava perfeitamente. A diferença era que as transações eram disparadas por robos, ao invés do processo normal de uma transação.

Em paralelo a isto, diversas pesquisas foram feitas e foi encontrado um *bug* na versão do *Oracle Application Server* 10.1.1.3. O *bug* encontrado referenciava as portas "*time_wait*" do produto. Seguindo nas pesquisas encontramos também um *bug* na versão do sistema operacional utilizado *RedHat* 4 *update* 5, *bug* também referente ao gerenciamento das portas em "*time_wait*".

Como as tentativas de reproduzir o problema que ocorria em produção não obtiveram sucesso, outra estratégia foi exigida, montar um novo ambiente com as versões mais novas de sistema operacional e também do servidor de aplicação da Oracle.

Após instalar os produtos em hardwares novos o ambiente foi novamente homologado. Os testes foram satisfatórios e o novo ambiente substituiu o já antigo ambiente do sistema concentrador.

O novo ambiente entrou em produção sem problemas, foi monitorado o número de portas e realmente as novas versões sanaram os problemas, existindo em média 80 portas em "*time_wait*", o que é normal dentro da situação do sistema.

Mas, mesmo assim, o sistema voltou a cair e o único erro gerado nos *logs* era de conexão recusada. Isto continuava gerando informações conflitantes, todavia que no sistema operacional estava sendo utilizando um total de aproximadamente 300 portas onde este suportaria mais de 64 mil e mesmo assim o sistema quando caia gerava erros de conexões.

Apesar de nesta fase o sistema estar novamente em *cluster* entre duas máquinas e se comportando melhor que no ambiente antigo, pois não ficava indisponível com tanta freqüência, o problema ainda não tinha sido resolvido.

Novamente em pesquisas descobrimos que o *Oracle Application Server* gerencia seus *conteiners* e quando este julga necessário, os reinicia. O que pode ser fatal no projeto,

visto que, alguns *conteiners* (módulo *cache* e resolvedor) são pré-requisitos dentro da estrutura do sistema.

A inteligência artificial do *OAS* estava causando sérios problemas. Por padrão o *OAS* monitora a cada 20 segundos seus *conteiners*, a monitoração é através de *pings*, e se o *conteiner* não responder, o *OAS* reinicia o mesmo automaticamente.

Após essa descoberta, passou-se a monitorar os reinícios dos *conteiners* e foi descoberto também que o *container* se reiniciava sem nenhuma intervenção humana, ou seja, era o recurso do *Oracle RAC* que os reiniciava. Nos *logs* do *OAS*, apenas era escrito que o *conteiner* foi parado e iniciado, o motivo que é o importante, não era escrito.

Alteramos as propriedades do *OAS* para que a sua monitoração seja executada a cada 600 segundos e se ocorrer 600 problemas seguidos, ai sim o *OAS* pode reiniciar o *containers*, ou seja, colocamos números absurdos para a realidade, visto que existem as monitorações da empresa que pegariam um possível problema em um tempo bem menor, mas esta ação foi para que o *OAP* não tome nenhuma atitude em cima dos *conteiners*. Já para os módulos pré-requisitos, foi configurado para que não haja monitoração sobre eles.

Agora com todas essas alterações no ambiente produtivo, atingiu-se mais de 500 horas sem nenhuma indisponibilidade no sistema concentrador.

O projeto ainda não esta concluído, esta sendo testado a entrada de um VIP (IP virtual) entre os dois nós dos servidores de aplicações, onde a meta é atingir indisponibilidade zero para o cliente final.

O VIP é a abreviação de IP Virtual que consiste em um compartilhamento de múltiplos domínios ou múltiplos servidores. O VIP elimina a dependência de um servidor sobre a interface de rede. Pacotes recebidos são enviados ao endereço virtual, porém todos os pacotes trafegam através da interface de rede real.

Neste momento, a intenção é deixar o sistema em cluster rodando em 2 servidores onde o VIP executará um balanceamento de carga entre eles. Em caso de indisponibilidade em qualquer um dos servidores, cabe ao VIP a responsabilidade do chaveamento das transações entre os servidores. Neste caso, o servidor ativo assume toda a responsabilidade do processamento das transações até que o outro servidor volte a responder normalmente.

Com este ambiente consolidado, é possível obter vantagens também nas atualizações do sistema que ficaram transparentes para o ambiente como um todo, sendo primeiro atualizado um servidor, depois o outro.

Esta é uma ação imediata, porém ao final do projeto teremos um balanceamento de carga controlado pelo VIP e servidores de contingência em outra rede.

Neste período em que o sistema concentrador esta no ar, percebemos que algumas alterações serão necessárias no ambiente proposto para conseguir processar o volume de transações esperado após a migração de todas operadoras e desativação do sistema antigo. No mínimo este ambiente deverá ser multiplicado por três para ter mais segurança, visto que o volume transacional deve chegar próximo de 500 mil transações dia.

Na figura 6.1 é possivel analisar o volume transacional desde a fase piloto do projeto até a fase de produção. A figura mostra o aumento do volume de transações a partir do terceiro mês, onde foi acrescentados novos produtos. Atualmente o volume transacional está bastante estável com média de aproximadamente 410 mil transações processadas por mês.

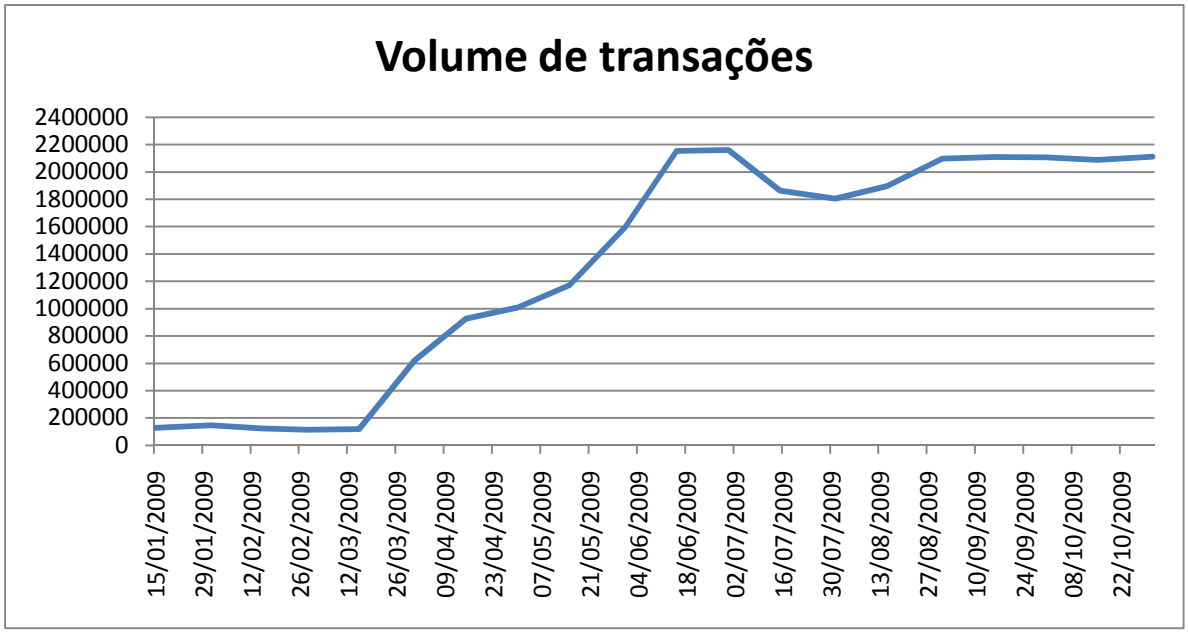

Figura 6.1 – Volume transacional no novo ambiente

Fonte: do autor.

Atualmente o novo SC não tem apresentado problemas e baixissima indisponibilidade ao cliente final. Enquanto isto, o SC antigo, está processando

aproximadamente 250 mil transações mês com no mínimo uma indisponibilidade por dia de aproximadamente 3 minutos, que é o procedimento preventivo de reinicio do sistema no periodo noturno, onde o volume transacional de sua rede é consideravelmente menor. Além deste 3 minutos, eventualmente ocorre indisponibilidade por problemas conhecidos já citados no capítulo 2 deste trabalho.

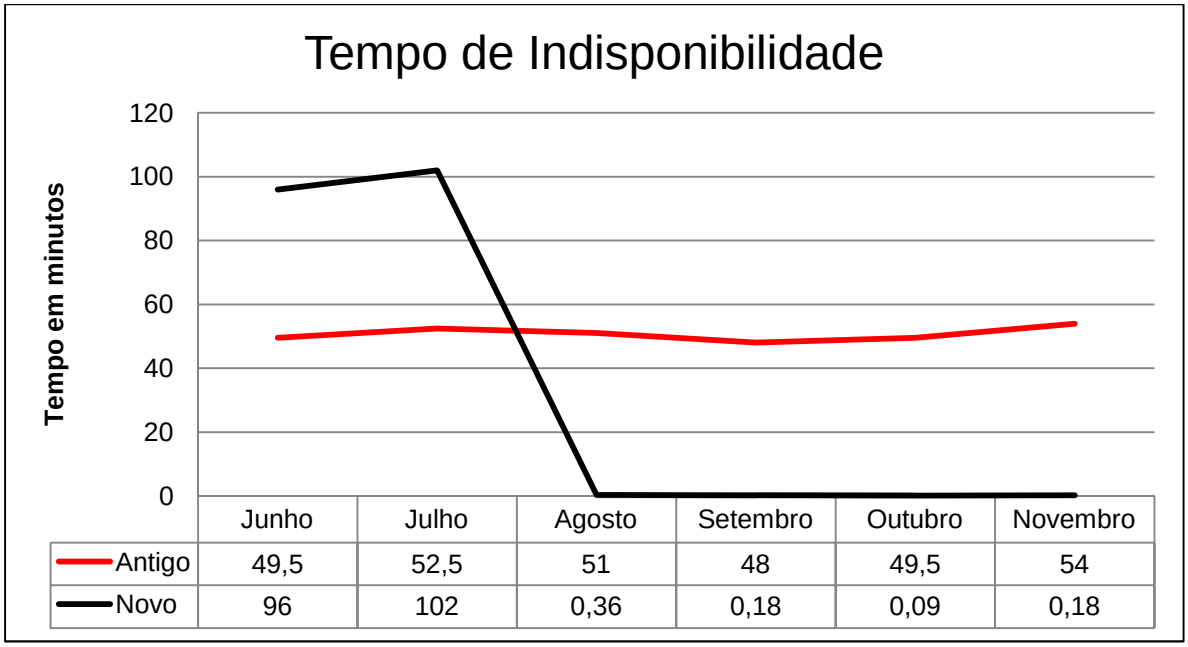

Figura 6.2 – Tempo de Indisponibilidade dos Sistemas

Fonte: do autor.

O gráfico acima mostra o resultado do trabalho aplicado, onde é possivel facilmente visualizar o aumento da disponibilidade do novo Sistema em comparação aos meses anteriores e também quando comparado com o Sistema antigo.

7 CONCLUSÃO

Uma vez concluído grande parte da proposta aqui apresentada, já com a aplicação sendo executada em ambiente de produção é possivel apresentar o seguinte conjunto de conclusões:

Não é viável trabalhar no projeto do sistema concentrador antigo, pois este possui graves problemas na sua estrutura. O tempo de desenvolvimento e testes no Sistema antigo até sua implantação é muito grande em relação a migração para o novo Sistema, além do risco da alteração no Sistema antigo que pode afetar diretamente todos os produtos que por ele transacionam. No atual ambiente transacional da organização, não é possivel correr este risco.

Com o desenvolvimento do projeto do novo SC atingiu-se as metas de eliminar os problemas existentes no SC antigo. Com o novo SC é possivel ter flexibilidade para controle dos módulos individualmente, controle de servidores ativos/passivos. Também é possivel obter indisponibilidade zero no momento de uma manutenção ou implantação de um novo produto. Além disto, o novo SC consegue processar o dobro de transações diariamente com um tempo menor de resposta quando conparado com a antiga versão.

Sendo assim, conclui-se que este projeto obteu grande sucesso, visto que, após algum tempo de utilização e ajustes, o sistema atingiu todos os seus objetivos propostos. Após a consolidação e sucesso do projeto, o sistema concentrador tornou-se tornando-se modelo de projeto para futuras implementações da organização.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALAPATI, Sam R. **Expert Oracle Database 10*g* Administration.** New York, NW, USA: Apress Books, 2005.

ALECRIM, Emerson (2004) "**Cluster: Principais conceitos**". Disponível em http://www.infowester.com/cluster.php. Acessado em: 06 de set. 2008.

Associação Brasileira de Cartões de Crédito e serviços. Disponível em http://www.abecs.org.br. Acessado em: 27 de ago. 2008.

AULT, Mike; TUMMA, Madhu. **Oracle 10g Grid & Real Application Clusters**. Kittrell, North Carolina, USA: Rampant TechPress, 2004.

CIRNE, Liliane Dantas; MACÊDO, Raimundo José de Araújo. **Uma abordagem para tolerância a falhas em JAVA através de comunicação em grupo** In: Anais do 18º Simpósio Brasileiro de Redes de Computadores**.** Disponível em http://www.lbd.dcc.ufmg.br:8080/colecoes/sbrc/2000/007.pdf. Acesso em: 09 de set. 2008.

E-ARTICLES, ***JAVA* e suas vantagens**. Disponível em http://e-articles.info/t/i/5006/l/pt/. Acesso em: 05 de set. 2009.

EHCACHE, ***Ehcache – User Guide***. Disponível em http://ehcache.org/EhcacheUserGuide.html. Acesso em: 01 de nov. 2009.

EHCACHE, ***Project Guide for Oracle RAC Implementation***. Disponível em http://www.oracle.com/technology/pub/articles/haskins-rac-project-guide.html. Acesso em: 22 de out. 2009.

HART, Matthew; JESSE, Scott. **Oracle Database 10*g*** High Availability with RAC, Flashback & Data Guard. Emeryville, California, USA: McGraw-Hill/Osborne, 2005.

HIBERNATE.ORG, ***Documentation Overview***. Disponível em https://www.hibernate.org/5.html. Acesso em: 3 de out. 2009.

HotWork Solution, ***Quartz User Guide***. Disponível em http://hotwork.sourceforge.net/hotwork/manual/quartz/quartz-user-guide.html#Introdução. Acesso em: 11 de nov. 2009.

HTML STAFF, **História da linguagem *JAVA***. Disponível em http://www.htmlstaff.org/ver.php?id=4383. Acesso em: 05 de set.2009.

HTML STAFF, **Vantagens da linguagem *JAVA***. Disponível em http://www.htmlstaff.org/ver.php?id=4384. Acesso em: 05 de set. 2009.

IMasters, **Spring Framework: Introdução**. Disponível em http://imasters.uol.com.br/artigo/4497/JAVA/spring_framework_introducao/. Acesso em: 5 de out.2009.

Instituto de Informática da UFRGS, **Processamento distribuído**. Disponível em http://www.inf.ufrgs.br/gpesquisa/procpar/dist.html. Acessado em: 29 de ago. 2008.

IPNews, **Inteligência de rede traz bons resultados e abre inúmeras possibilidades.** Disponível em http://www.ipnews.com.br/voip/infra-estrutura/qualidade/inteligencia-de-rede-traz-bons-resultados-e-abre-in-meras-possibilidades.html Acessado em 20 de nov.2008.

JAVA Anywhere, ***Hibernate***. Disponível em http://JAVAwora.blogspot.com/2007/08/hibernate.html. Acesso em: 01 de nov.2009.

JAVA FREE, ***Quartz***. Disponível em http://JAVAfree.uol.com.br/wiki/Quartz. Acesso em: 13 de nov.2009.

JUN, RICARDO, ***Spring Framework* - O que é afinal de contas e para onde ele vai?**. Disponível em http://blog.globalcode.com.br/2009/09/spring-framework-o-que-e-afinal-de.html. Acesso em: 01 de nov. 2009.

Linhares, Maurício , **Introdução ao *Hibernate***. Disponível em http://www.guj.com.br/content/articles/hibernate/intruducao_hibernate3_guj.pdf. Acesso em: 01 de out.2009.

LONEY, Kevin. **Oracle 10*g*: o manual do DBA /** Kevin Loney, Bob Bryla; tradução de DocWare Traduções Técnicas**.** Rio de Janeiro: Campus, 2005.

Olimpus Tecnologia, ***Oracle Rac***. Disponível em http://www.olimpius.com.br/index.php?option=com_content&task=view&id=84&Itemid=125. Acesso em: 11 de set. 2009.

ORACLE Brasil – **Oracle Applicarion Server**, Disponível em http://www.oracle.com/global/br/appserver/index.html. Acessado em: 05 de set. 2008.

ORACLE Brasil. **Banco de dados Oracle**. Disponível em http://www.oracle.com/global/br/database/index.html. Acessado em: 05 de set. 2008.

Oracle Brasil (2005) " **Estudo indica que Oracle o Real Application Clusters (RAC) é a solução mais inovadora do mercado**". Disponível em http://www.oracle.com/global/br/corporate/press/2005_nov/rac_solucao_inovadora.html. Acessado em: 11 de out. 2009.

Oracle, ***Real Application Cluster***. Disponível e http://www.oracle.com/lang/pt/database/rac_home.html. Acesso em: 10 de outubro de 2009

PEREIRA, Rodrigo Máron, ***Ehcache - Ehcaching na web***. Disponível e http://knol.google.com/k/rodrigo-máiron-pereira/ehcache/3ta539a1fivcy/5#. Acesso em: (de nov. 2009.

PITANGA, Marcos. **Computação em cluster:** o estado da arte da computação. Rio (Janeiro: Brasport Livros e Multimídia Ltda, 2003. 322p.

PITANGA, Marcos. **Construindo supercomputadores com linux.** 2ª edição. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2004. 292p.

Requisitos não funcionais. 2008. Disponível em http://ladoservidor.com/node/12. Acesso em: 09 de setembro de 2008.

RIGONI, RONALDO, **Trabalhando com *Flex e XML Sockets* com *Apache Mina***. Disponível em http://www.ronaldorigoni.com.br/trabalhando-com-flex-e-xml-sockets-com-apache-mina/?lang=pt. Acesso em: 06 de nov. 2009.

SUN Microsystems, ***JAVA Technology Reference***. Disponível em http://JAVA.sun.com/reference/index.jsp#documentation. Acesso em: 16 de out. 2009.

Wikipedia, **Sistema de alta disponibilidade**. Disponível em http://pt.wikipedia.org/wiki/Sistema_de_alta_disponibilidade. Acesso em: 10 de set. 2008.

Wikipedia, **Spring Framework**. Disponível em http://pt.wikipedia.org/wiki/Spring_Framework. Acesso em: 31 de set. 2009.

Wikipedia, ***Hibernate***. Disponível em http://pt.wikipedia.org/wiki/Hibernate. Acesso em: 01 de nov.2009.