Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

Mestrado em Engenharia Informática e de Computadores
Arquitetura de Sistemas Distribuídos
2013/2014
Relatório do Projeto Final

Nome	Nº de Aluno	E-mail
Rui Miranda	A32342	a32342@alunos.isel.pt
David Coelho	A21359	a21359@alunos.isel.pt
Frederico Ferreira	A7066	a7066@alunos.isel.pt

DEETC, Fevereiro de 2014

1 INDEX

ASIVeste, Lda.

2 INTRODUÇÃO

O presente relatório descreve os aspetos mais relevantes da implementação de um hipotético Sistema de Gestão de uma empresa focada na comercialização de vestuário, descrita para o projeto final da Disciplina de "Arquitetura de Sistemas de Informação" do Mestrado de Informática e Computadores do ISEL.

O relatório está organizado de forma a apresentar primeiro os requisitos e o glossário dos termos usados (negócio ou contexto, técnicos e da implementação), seguindo depois com os capítulos requeridos pelo enunciado do trabalho.

Os ambientes pretendidos foram simulados numa única máquina e onde oportuno são assinaladas as principais diferenças para um possível sistema real.

Neste protótipo, o foco foi nos conceitos, tecnologia e técnicas usadas na cadeira sendo o restante desenvolvimento de acordo com uma aproximação "good enough", isto é, apenas o necessário para os demonstrar.

O relatório está organizado de forma a apresentar primeiro os requisitos e o glossário dos termos usados (negócio ou contexto, técnicos e da implementação), seguindo depois com os capítulos requeridos pelo enunciado do trabalho.

3 REQUISITOS

A ASIVeste, Lda é uma empresa de venda direta ao público de vestuário de Senhora, Criança, Homem e Desportista, que usa uma rede de lojas para apresentação, demonstração e venda de produtos, mas cuja entrega é feita remotamente a partir dos seus armazéns.

A lista seguinte apresenta os requisitos extraídos direta ou indiretamente do enunciado.

3.1 REQUISITOS FUNCIONAIS

3.1.1.1 Fornecedores

Os registos de Fornecedores são geridos (criados, alterados e removidos) na sede.

3.1.1.2 Produtos na Sede

Os produtos são geridos (criados, alterados, removidos) na Sede. Entre a Sede e as Lojas é usado o Código de produto como identificador do produto.

Um produto só tem um fornecedor.

3.1.1.3 Produtos nas Lojas

Nas lojas os produtos necessitam apenas de ter o Código e a quantidade mínima em Stock. Considerando que há venda a clientes, considera-se também necessário dispor a sua designação e o preço de venda.

3.1.1.4 Listagem de Produtos

Na sede será possível fazer listagens de stock de todos os produtos.

3.1.1.5 Venda de Produtos

O processo de venda de produtos, é iniciado na loja com o registo da venda e a morada de entrega e é complementado no armazém com a Expedição da Venda para a morada do comprador. Não há entrega de produtos nas lojas, que apenas possuem amostras.

A aplicação usada pelo armazém é a da sede, pelo que as vendas terão de fluir das lojas para a sede.

Na venda deverá ser validada quantidade em stock e rejeitada se não existir stock suficiente para a satisfazer (razão pela qual o stock é mantido atualizado nas lojas).

3.1.1.6 Procedimentos de Gestão mais comuns

Os procedimentos de gestão mais comuns, para o conjunto total das lojas são realizados na sede.

3.1.1.7 Encomendas a fornecedores

As Encomendas a fornecedores são realizadas na Sede, sempre que o stock de um produto desce abaixo do stock mínimo.

As Encomendas de fornecedores são rececionadas na Sede.

3.1.1.8 Receção de Encomendas a fornecedores

Na receção de produtos encomendados a fornecedores deverá ser atualizado o stock na sede e no grupo de lojas respetivo de forma síncrona.

3.2 REQUISITOS NÃO FUNCIONAIS

3.2.1.1 Arquitetura Física dos Centros

O negócio está distribuído por 3 centros geograficamente separados: Sede e 2 Grupos de Lojas. Cada um dos centros é composto por um servidor aplicacional e um SGBD. Os grupos de lojas são independentes entre si.

3.2.1.2 Aplicação das Lojas

Cada loja tem um ou vários computadores pessoais onde corre a lógica de apresentação da aplicação de vendas, que está ligada ao servidor aplicacional central do grupo de lojas a que pertence.

3.2.1.3 Autonomia das Lojas

As lojas, para efeitos da colocação de vendas, devem poder funcionar com total autonomia da sede e do outro conjunto de lojas.

4 GLOSSÁRIO

Lista de termos usados na descrição do negócio e da tecnologia

ASIVESTE	Referência à totalidade da organização ou da aplicação (em função contexto em que é usada)
LHD	Conjunto de Lojas de Vestuário de Homem e Desportista
LSD	Conjunto de Lojas de Vestuário de Senhora e Criança
Venda	Define o processo de aquisição de um artigo / produto por um cliente. No sistema designa a entidade que suporta esse processo. A venda associa um (ou vários) produto a um cliente.
Encomenda	A encomenda representa a colocação de uma compra de um produto a um fornecedor.
Fornecedor	O fornecedor é uma entidade externa que produz ou disponibiliza um artigo ou produto. No modelo considerado é único por produto.
Produto	Produto é um artigo tipificado e que pode ser vendido de forma disjunta num conjunto de lojas
Fila de Mensagens	Entidade computacional usada na gestão de mensagens de vendas, que permite um desacoplamento temporal entre produtor e consumidor de mensagens.

Lista de termos usados na descrição do negócio e da tecnologia

ASIVESTE	Referência à totalidade da organização ou da aplicação (em função contexto em que é usada)
LHD	Conjunto de Lojas de Vestuário de Homem e Desportista
LSC	Conjunto de Lojas de Vestuário de Senhora e Criança

Lista de termos usados na implementação

ASIVesteSede	Projeto com a aplicação web usada na Sede
ASIVesteLoja	Projeto com a aplicação web usada nas Lojas
QueueVendas	Projeto usado para a propagação das vendas das Lojas para a Sede, ou descrição da interface paras as filas de mensagens de vendas
VendasPublisher	“console application” usada na inserção de Vendas na fila de mensagens, usada apenas para debug.
VendasReceiver	“console application” usada na extração de Vendas da fila de mensagens. Insere os registos nas tabelas da Sede.

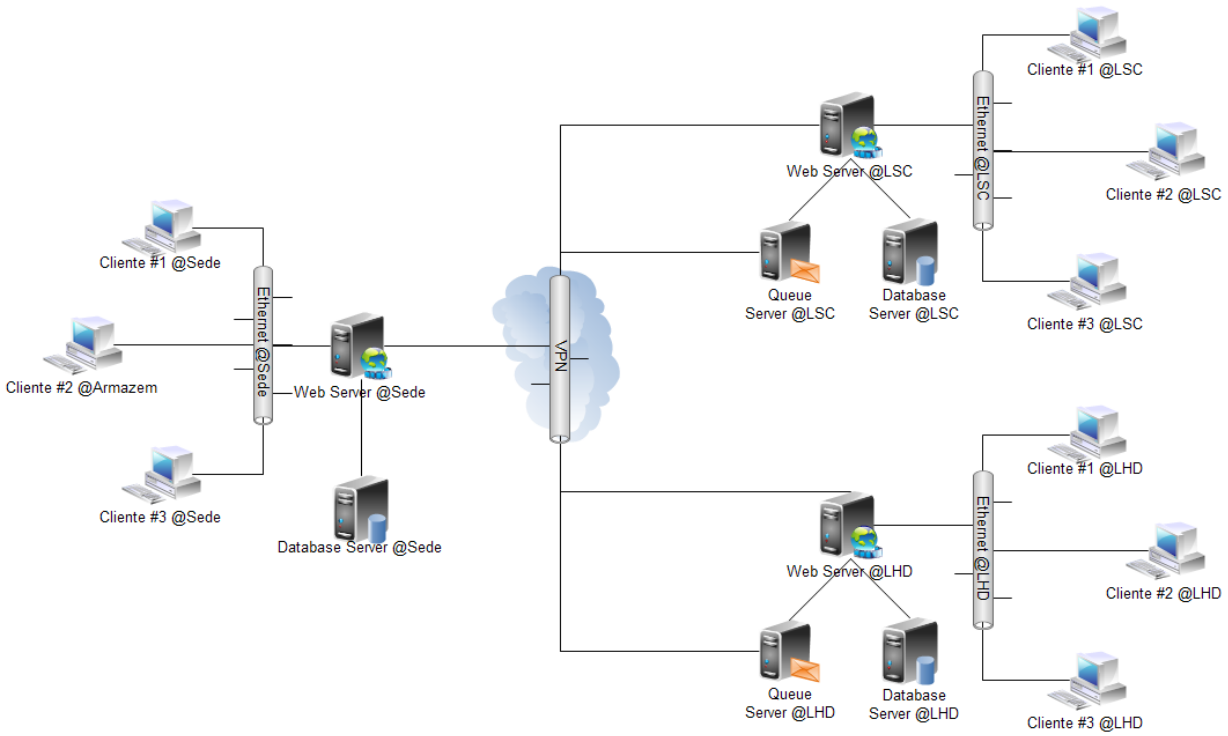
queueLSC	Nome da fila de espera usada pelas Lojas LSC para colocarem as suas mensagens de vendas
queueLHD	Nome da fila de espera usada pelas Lojas LHD para colocarem as suas mensagens de vendas
sp_inserirProduto	Bloco T-SQL usado na sede na inserção de um produto e sua propagação para o grupo de Lojas respetivo
sp_actualizarProduto	Bloco T-SQL usado na sede na atualização de stock de um produto recebido numa Encomenda
sp_removerProduto	Bloco T-SQL usado na sede para a remoção de um produto e sua propagação ao grupo de Lojas
sp_realizarVenda	Bloco T-SQL para processar um Venda recebida na sede e desencadear o processo de reposição de stock, quando aplicável
sp_realizarEncomenda	Bloco T-SQL usado na colocação de uma encomenda
sp_receberEncomenda	Bloco T-SQL usado na receção de produtos de uma encomenda

6 ARQUITETURA

A arquitetura da Solução ASIVeste que desenvolvemos, pretende dar resposta aos requisitos técnicos e operacionais apresentados e alguns emergentes de uma boa prática de engenharia, como é o caso da disponibilidade e do desempenho. Como componentes básicos da arquitetura que escolhemos, estão as bases de dados relacionais, as filas de mensagens e as aplicações em 3 camadas.

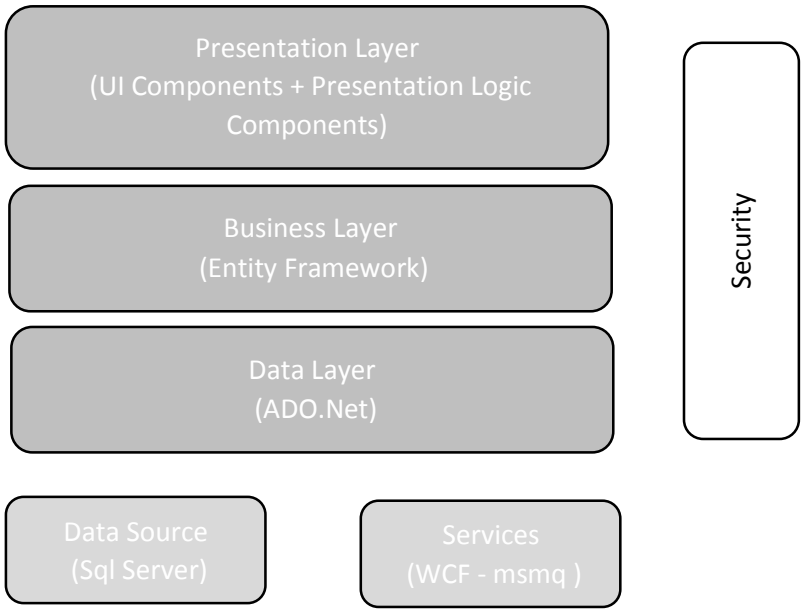
A figura abaixo apresenta a arquitetura geral da solução.

A infra-estrutura das sede é composta por um servidor aplicacional (SA) e um servidor de dados (SD) onde reside uma base dados relacional, a interface com os utilizadores usa *think clients*. A infra-estrutura para cada uma das lojas é semelhante. Entre cada grupo de lojas e a sede é usada uma fila de mensagens que reside no servidor do seu grupo de lojas.



A imagem seguinte apresenta a arquiectura aplicacional usada, e que é comum às aplicações ASIVesteSede e ASIVesteLoja. As lojas são apenas diferencidas ao nível das configurações dos produtos que vendem e da fila de mensagens que usam, para enviar as suas vendas para a sede.

Nota, para este protótipo a segurança foi deixada um nível minimo, e seria uma das áreas de melhoria para uma implementação em produção, num ambiente real.



8 MODELO DE DADOS

8.1 ESQUEMA LÓGICO GLOBAL

O esquema lógico global de dados do sistema distribuído ASIVeste será o seguinte:

Produto(ProdutoID[PK],Tipo,Codigo,Designacao,StockQtd,StockMinimo,Preco,
Fornecedor_FornecedorID[FK])

Fornecedor(Fornecedor ID[PK],Numero,Nome,Morada)

Venda(VendaID,NomeCliente,MoradaCliente,Estado)

VendaProdutos(VendaProdutosID[PK],Codigo,Qtd,Estado,Venda_VendaID[FK],Produto_ProdutoID[FK])

Encomenda(EncomendaID[PK],Qtd,Estado,Produto_ProdutoID,VendaProduto_VendaProdutoID[FK])

8.2 ESQUEMA DE FRAGMENTAÇÃO

Dado o requisito de autonomia local de cada uma das entidades do sistema distribuído (Sede, e Lojas),o esquema global do Produto será fragmentado verticalmente e horizontalmente:

8.2.1 Partição vertical (Sede e Lojas)

A relação entre ProdutoSede e ProdutoLojas é feita com base no Codigo do produto (tal como solicitado no enunciado).

ProdutoSede(ProdutoID[PK],Tipo,Codigo,Designacao,StockQtd,StockMinimo,Preco,
Fornecedor_FornecedorID[FK])

π{ProdutoID,Tipo,Codigo,Designacao,StockQtd,StockMinimo,Preco,Fornecedor_FornecedorID}(Codigo)

ProdutoLojas(ProdutoID[PK],Codigo,Designacao,StockQtd,Preco)

π{ProdutoID,Codigo,Designacao,StockQtd,Preco}(Codigo)

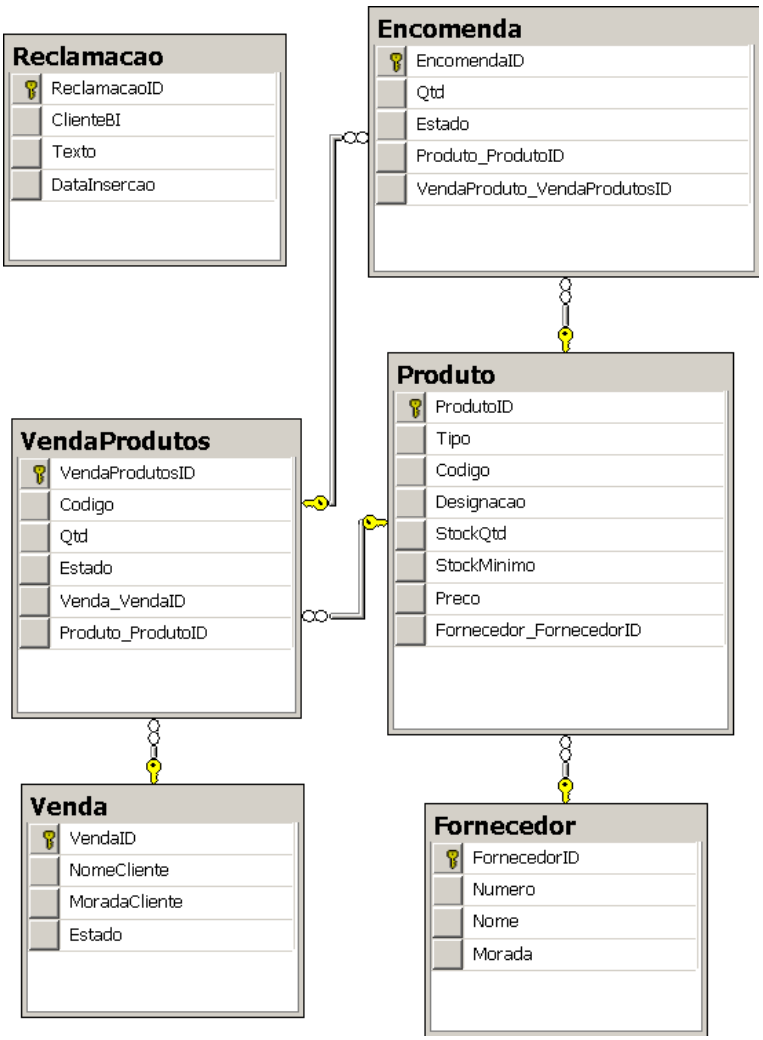
8.2.2 Partição Horizontal (Sede)

ProdutoLSC= σ{tipo = Senhora V tipo = Crianca}(ProdutoLojas)

ProdutoLHD= σ{tipo = Homem V tipo = Desporto }(ProdutoLojas)

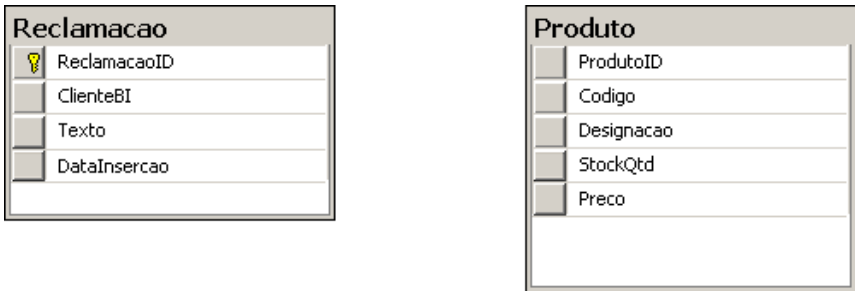
8.3 MODELO ER SEDE

Partindo dos esquemas apresentados anteriormente, já incluindo os requisitos da opção 1, obtém-se na Sede o seguinte esquema de dados:



8.4 MODELO ER LOJAS

Partindo dos esquemas apresentados anteriormente, já incluindo os requisitos da opção 1, obtém-se em cada conjunto de lojas (LSC e LHD) o seguinte esquema de dados:



9 IMPLEMENTAÇÃO

9.1 GERAL

A implementação foi feita usando como base a tecnologia .NET 4.5, EF 5, WCF, SQL Server 2012 e o Visual Studio 2012. Para permitir um desenvolvimento em equipa usamos um repositório central no GitHub.

A solução implementada é composta por:

- Aplicação da Sede;
- Aplicação de Loja;
- Filas de Mensagens para as Vendas;
- Aplicação "consola" para extração de mensagens das filas de vendas;
- Três instâncias de base de dados SQL Server;
- Código T-SQL;
- Scripts SQL.

Os pontos seguintes descrevem em maior detalhe cada um destes componentes do sistema e algumas particularidades da implementação.

9.2 ASIVESTESEDE

A ASIVesteSede é a aplicação que implementa as funcionalidades requeridas na sede e no armazém. Disponibiliza as facilidades de:

- Consulta, criação, alteração e remoção de Fornecedores;
- Consulta, criação e alteração de Produtos;
- Consulta de Encomendas. Embora possam ser colocadas também pelo utilizador, o processo normal de criação de Encomendas é resultado da Vendas recebidas das lojas;
- Entrada de Encomendas, isto dar como recebido o produto encomendado, para atualizar os stock locais no grupo de lojas respetivo;
- Consulta do histórico de vendas;
- Expedição de Vendas, que é uma atividade a realizar pelos armazéns e que ao nível da aplicação se resume à mudança de estado da Vendas;
- Listagem do Stock de Produtos.

A aplicação foi implementada em 3 camadas usando o padrão MVC com EF 5 e .NET 4.5. Nesta implementação usámos a opção de *Code First* começando por definir as nossas Entidades de Negócio, tendo a framework definido as tabelas relacionais subjacentes. A framework gera um conjunto de código para a criação das tabelas na base de dados e também para a sua atualização sempre que o modelo é usado. Contudo a aplicação deste código (geração e alteração da estrutura) são corridos fora do programa / aplicação na consola PM> e sob total controlo do programador. Para a aplicação das alterações é ainda configurada a possibilidade de aplicação (ou não) caso existam dados na base de dados.

A criação e alterações da base de dados estão contidas no folder *Migrations* onde se inclui um ficheiro de configuração inicial, para incluir dados semente ou para popular o modelo com dados de teste (muito útil para o desenvolvimento/debug do código que vai sendo produzido). Para controlar a aplicação das *migrations* é criada automaticamente uma tabela na base de dados onde são registados as migrações realizadas.

Ao nível da consola é possível criar pontos de migração para futuro aplicação, por exemplo em produção (a base de dados em uso está definida em Web.config).

É possível passar a controlar diretamente a base de dados (fora do controlo de Code First), mas não é recomendável (ou mesmo viável) manter os dois modelos e uma vez mudado o controlo para "manual" deverá ser mantido a partir daí.

Desta forma pareceu-nos que o code first poderá ser uma boa forma de arranque de um projeto, quando o modelo de entidades já foi pensado e existe uma linha geral de evolução para esse modelo, com passagem para um controlo *database first* quando for necessário compatibilizar com produção (existência de dados) ou necessidades de outras aplicações, ou ainda uma metodologia / processo de trabalho centrado primeiro na definição do modelo de dados.

A implementação por camadas é perfeitamente identificada na estrutura do projeto estando repartida pelos *folders: Models e DAL, Views e Controllers,* correspondendo à camadas de dados, apresentação e lógica aplicacional.

No acesso aos dados na camada DAL, alguns acessos foram substituídos por chamadas a procedimentos T-SQL, por forma a dar resposta aos requisitos do enunciado e estão neste caso:

- O tratamento do Produto e a sua replicação, implementados por sp_inserirProduto, sp_actualizarProduto e sp_removerProduto;
- O tratamento de um venda e o desencadear das respetivas atividades na sede, implementados por sp_realizarVenda;
- O tratamento das encomendas com sp_realizarEncomenda e sp_receberEncomenda.

Controlo de acessos: embora a framework usada MVC gere código para o controlo de acesso e prepare a aplicação gerada para essa funcionalidade, optámos por não concluir a sua parametrização por não ser essencial ou relevante para o protótipo em causa.

9.3 VENDASRECEIVER

Este componente corre do lado da Sede e é o responsável pela extração das mensagens de Vendas colocadas nas filas dos dois grupos de lojas e a criação dos registos de Vendas na sede.

Sendo uma *console application,* corre de forma autónoma da aplicação da sede, a ASIVesteSede, e extrai de forma transacional os registos das referidas filas. Em termos da escalabilidade, este aplicativo poderia ter várias instâncias se fosse necessário para esvaziar mais depressa as filas.

9.4 ASIVESTELOJA

A aplicação a correr nas lojas tem por funcionalidade principal a "Venda" de produtos. Uma venda deverá validar o Stock disponível e deverá ser abortada se a informação de stock disponível for insuficiente para satisfazer a venda pretendida.

As vendas ocorrem em simultâneo em diversas lojas, e um produto poderá ser vendido ao mesmo tempo em diversas lojas, pelo que será necessário implementar um mecanismo de controlo de concorrência, escolhendo um dos 2 disponíveis o "pessimista" e o "otimista":

- Modo "Pessimista":
- Modo "Otimista":

Tendo em conta o tipo de negócio a nossa solução foi pela utilização do otimista, que permite um melhor desempenho total do sistema, com um custo eventual de alguns cancelamentos de Venda no seu fecho. No modelo de negócio isso permite também informar de imediato o cliente sem criar falsas expectativas.

Referência de implementação de modelos de concorrência em EF MVC:
http://www.asp.net/mvc/tutorials/getting-started-with-ef-5-using-mvc-4/handling-concurrency-with-the-entity-framework-in-an-asp-net-mvc-application

9.5 FILAS DE VENDAS

As filas de vendas são o mecanismo escolhido para o desacoplamento temporal entre os grupos de lojas e a sede. Também permitiriam um desacoplamento tecnológico, que neste caso não é necessário.

Na implementação das filas foi automatizada a sua criação no código de colocação da Venda, caso não exista. Em termos do código foi necessário considerar que as filas só têm um sentido, e que não permitem reposições.

As filas criadas foram duas *queueLSC* e *queueLHD* uma para cada grupo de lojas "Loja Senhora e Criança" e "Loja Homem e Desportista" respetivamente. E são filas Privadas e transacionais e foram usados os valores default para os parâmetros *receiveRetryCount, maxRetryCycles, retryCycleDelay e receiveErrorHandling*. No ambiente de simulação usado o computador não fazia parte de um domínio e não tinha Active Directory instalada, pelo que *transport security* está desligado.
A leitura das filas é feita diretamente da sua origem (server do grupo de loja respetivo), uma implementação alternativa poderia ser a implementação de filas também na sede e usar os mecanismos de transporte entre filas para colocar nessas filas as vendas recebidas nas filas das lojas, criando um cenário de um message bus.

10 ELEVADA DISPONIBILIDADE NO SERVIÇO DE VENDAS

Considerando a necessidade de ter alta disponibilidade no serviço de vendas, seria necessário ter em cada grupo de lojas mecanismos, que permitissem assumir redundância para o Servidor Aplicacional e para o SGBD, para obviar a falhas destes elementos.

Ao nível do servidor aplicacional e tendo por base o tipo de desenvolvimento e implementação que realizámos, poderia ser considerada a inclusão de mais um servidor aplicacional (em paralelo com o primeiro) tendo como front-end um balanceador de carga. Este balanceador reencaminharia automaticamente o trafego para o servidor aplicacional operacional no caso de falha de um deles, e quando todos operacionais, permitiria balancear a carga e aproveitar a capacidade instalada na resposta pedidos dos clientes.

Ao nível do SGBD e mais precisamente da base de dados de suporte às vendas, a redundância teria de ser usada por um (ou vários) servidor de base de dados ao qual seriam aplicadas as alterações efetuadas na Base de Dados primária. Estas alterações permitir-lhe-iam assumir o controlo no caso de falha do primário.

Na utilização de apenas um segundo servidor de base de dados a recomendação seria o uso de Database Mirroring, enquanto que, no caso de uso de vários, seria necessário usar também Log Shipping.

Embora o Database Mirroring possa ser substituído por Log Shipping, o potencial de perca de informação é maior, uma vez a informação contida num log aberto não é passada para o secundario, e é uma unidade maior que os blocos usados no log shipping.

Um SGBD adicional poderia ser usado como *Witness* para permitir a comutação automática no caso de faillover do nó principal.

A solução de aumento da disponibilidade para as Vendas implicaria a introdução dos seguintes elementos adicionais:

- Servidor (s) aplicacional e de um load balancer;
- Um SGBD com capacidade igual ao Principal a funcionar como Database Mirroring;
- Um SGBD com capacidade menor, a funcionar como “Witness”.

A perca de informação em caso de falha seria:

- Transações a decorrer (e não concluídas) no caso de falha do servidor aplicacional;
- Transações a decorrer e mais algumas ainda não replicadas no caso do Database Mirroring (assumindo que se pretendia, como parece fazer sentido neste caso que o Database Mirroring estivesse a correr de forma assíncrona).

O investimento necessário seria basicamente um pouco mais que a duplicação do inicial no suporte cada grupo de lojas. O “pouco mais” resulta do acrescentar do load balancer e do “witnesse”.

Nota: a alternativa de failover cluster seria igualmente uma alternativa para disponibilizar uma elevada disponibilidade ao nível da base de dados. Se a instancia do SGBD tivesse outras bases de dados, dependendo da sua dimensão esta alternativa poderia perder contra o Database Mirroring, que apenas replica base de dados (e não instância) que seria por isso menos pesado.

Neste caso a diferença de custos estaria na infra-instrutura de suporte à partilha de discos que deverá ser potencialmente mais cara, que a utilização de um "witness" no caso que apresentámos. (eventualmente o witness poderá estar a funcionar num outro sistema existente ou em último caso no sgbd secundário).

11 MAIOR ROBUSTEZ NA SEDE

Ao nível da sede é pretendido aumentar a robustez às falhas do seu sistema, considerando, que seria possível tolerar algumas horas de falha de funcionamento. Neste cenário, a alternativa "Log Shipping" seria bastante interessante, pois o tempo de aplicação do último log propagado para a base de dados Secundária, não seria impedimento.

O custo de implementação neste caso seria apenas a duplicação do sistema de suporte ao SGBD.

A potencial perca de informação seria determinada pela dimensão dos logs, uma vez que as transações registadas num log aberto serão perdidas. Se um log não tiver sido transferido para o log o sistema de destino será igualmente perdido e será esse o atraso (não recuperável) entre o sistema principal e o secundário.

12 REFERÊNCIAS

Best Practices for Queued Communications - http://msdn.microsoft.com/en-us/library/ms731093(v=vs.110).aspx

Queues Overview - http://msdn.microsoft.com/en-us/library/ms733789(v=vs.110).aspx

Microsoft Application Architecture Guide – 2nd Edition

Programming Entity Framework DbContext – Julia Lerman & Rowan Miller – O'Reilly

Programming ASP.NET MVC 4 – Jess Chadwick, Todd Snyder & Hrusikesh Panda – O'Reilly

Database Mirroring and Log Shipping - http://technet.microsoft.com/en-us/library/ms187016.aspx