ASIVeste, Lda.

# 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.

# Requisitos

# 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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

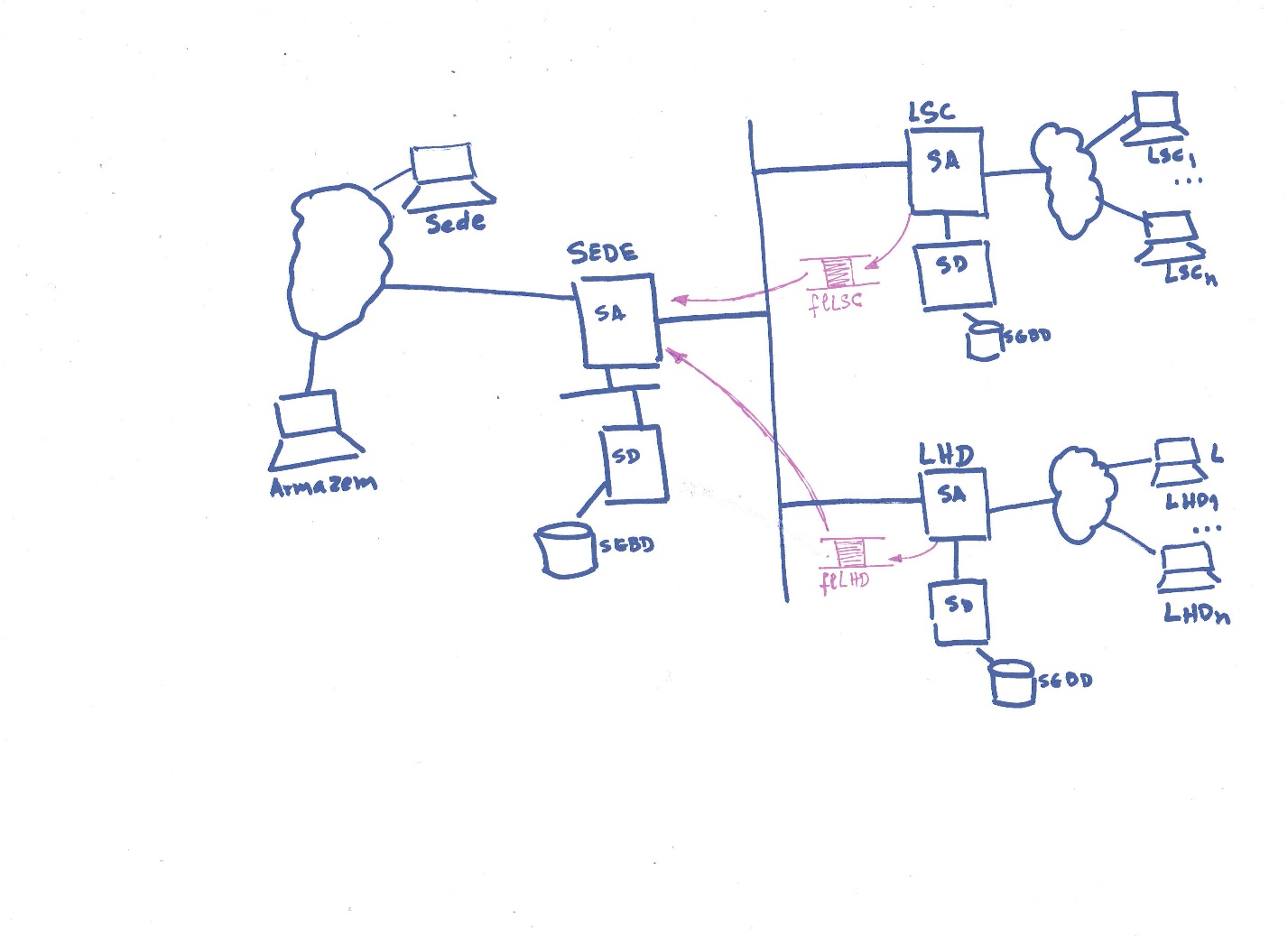
Lista de termos usados na implementação

|  |  |
| --- | --- |
| 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 |
|  |  |
|  |  |

# 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.

Data Source

(Sql Server)

Services

(WCF - msmq )

Data Layer

(ADO.Net)

Business Layer

(Entity Framework)

Presentation Layer

(UI Components + Presentation Logic Components)

Security

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.

# modelo de Dados

* 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])

* 1. **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:

* Partição vertical (Sede e Lojas)

ProdutoSede(ProdutoID[PK],Tipo,Codigo,Designacao,StockQtd,StockMinimo,Preco,Fornecedor\_FornecedorID[FK])

π{ProdutoID,Tipo,Codigo,Designacao,StockQtd,StockMinimo,Preco,Fornecedor\_FornecedorID}(Produto)

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

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

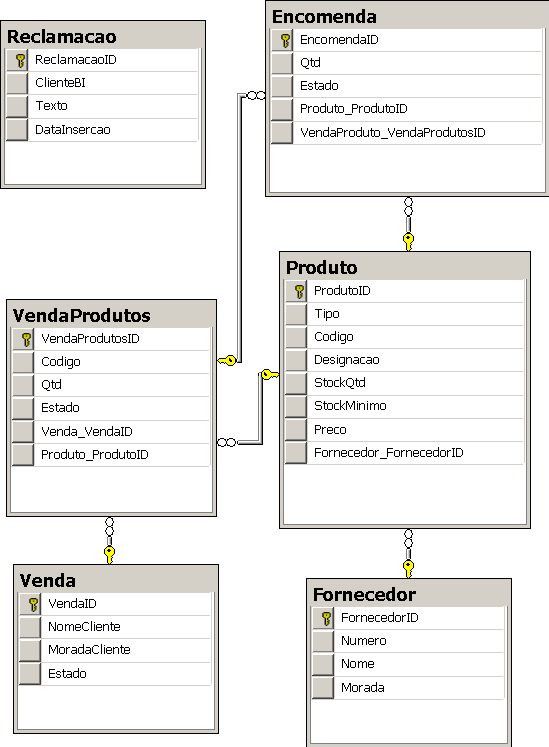
* Partição Horizontal (Sede)

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

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

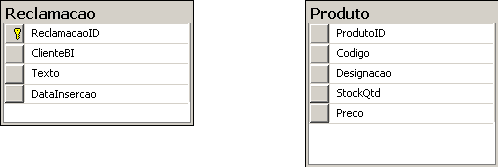
* 1. **Modelo ER Sede**

Partindo dos esquemas apresentados anteriormente obtém-se na Sede o seguinte esquema de dados:



* 1. **Modelo ER Lojas**

Partindo dos esquemas apresentados anteriormente obtém-se em cada conjunto de lojas (LSC e LHD) o seguinte esquema de dados:



# Implementação

## 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 (atividade a realizar pelos armazéns);
* 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.

## 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.

Trata-se de uma aplicação de consola, que disponibiliza a implementação, VendaServico, do serviço especificado pela interface IVendaServico, é este serviço que consome as mensagens de Vendas, e através de EF acede à base de dados da sede para realizar as operações necessárias.

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.

## 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. Após a validação coloca a mensagem da Venda na fila, e atualiza o stock do produto localmente, isto num contexto transacional.

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 “optimista”:

* Modo “Péssimista”:
* Modo “Optimista”:

Tendo em conta o tipo de negócio a nossa solução foi pela utilização do optimista, 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>

## 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.

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.

# “Deployment”

# Opções: Alta Disponibilidade e Escalabilidade

# Notas Finais

# 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