



Mestrado Integrado em Engenharia Informática e
Computação

Sistemas de Informação



Sales Order Picking
Especificação do projeto

Grupo I:

201204979 - Maria João Marques - ei12104@fe.up.pt
201208066 - Mário André Ferreira - ei12105@fe.up.pt
201206054 - Pedro Miguel Faria - ei12097@fe.up.pt
201005350 - Pedro Vieira Martins - ei10096@fe.up.pt
201200742 - Sofia Oliveira Reis - ei12041@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

11 de Outubro de 2015

1 Introdução

Esta especificação serve de apoio ao projeto a desenvolver para a disciplina de Sistemas de Informação lecionada no 4º ano do Mestrado Integrado em Engenharia Informática e Computação na Faculdade de Engenharia e Universidade do Porto.

O projeto é uma aplicação web responsável por otimizar e automatizar o processo de *picking*¹ e *putaway*². A plataforma chama-se CALIDRIS. Este é o nome da espécie de um pássaro que voou uma distância superior à distância entre a Terra e a Lua e que consegue viajar 8 mil km ou mais sem parar. Como o nosso objetivo, é ter uma aplicação que permita a ligação das entidades a todos os armazéns dispersados pelo mundo, nada como ter o CALIDRIS a representar a eficiência dessa ligação ainda que comprida e duradoura.

A ideia principal será permitir a um grupo de entidades, autorizada pela administração da CALIDRIS, que consiga fazer encomendas de produtos de moda e que os armazéns ao mesmo tempo consigam dar resposta a estas encomendas da forma mais eficaz, utilizando o software proporcionado por nós.

2 Visão do Projeto

Hoje em dia, as empresas têm que satisfazer as exigências de todos os seus clientes num mercado cada vez mais competitivo e intolerante a falhas, pelo que o nosso principal objetivo é agilizar a distribuição de artigos de vestuário e diminuir a ocorrência de erros ao longo da mesma.

Para que o sistema seja eficiente, é necessário ter uma ordem normal de acontecimentos desde que uma entidade solicita os artigos de vestuário até os receber. Essa ordem normal irá começar com a encomenda detalhada de artigos dos nossos armazéns por uma entidade, que irá estar autorizada pela administração da CALIDRIS a fazê-la (*picking*). Essa encomenda irá gerar uma ordem de venda para a entidade e uma ordem de compra que irá ser enviada ao armazém ou armazéns respetivos. No armazém, um armazenista irá pegar nas ordens de compra e verificar se as encomendas são satisfazíveis. Caso sejam, a encomenda irá ser aprovada e os artigos adjacentes irão ser preparados para a sua exportação (*putaway*).

Para facilitar todos estes processos, irão ser emitidas guias com as rotas para que os armazenistas consigam, responder às encomendas da forma mais rápida e eficiente possível; e irão ser emitidas listagens com os artigos correspondentes a cada encomenda, as suas quantidades e as *slots* onde se encontram. Além destas, as ordens de compra e venda também são consideradas listagens onde estarão detalhados os artigos de vestuário de cada encomenda e a procurar no armazém. Os armazéns estarão divididos por *slots*, que são os locais de armazenamento, com uma etiqueta única, cuja estrutura ainda se encontra por determinar, mas que terá em conta o corredor, o seu volume e a sua altura.

A nível de tecnologias, iremos utilizar o **Primavera** onde estará alocada a nossa base de dados, um **ERP**³ que fará a ligação aos *web servers* que irão ser

¹O processo de *picking* envolve requisitar produtos de um armazém e prepará-los para serem enviados à entidade que os solicitou.

²O processo de *putaway* envolve pegar nos produtos escolhidos para *picking*, retirá-los dos seus locais de armazenamento e enviá-los a quem os solicitou

³**Enterprise Resource Planning**: aplicação informática que integra todos os dados e processos de uma organização numa mesma plataforma.

implementados em **Ruby On Rails**. Em termos de interface, serão utilizadas duas *frameworks*, maioritariamente, que são o **Bootstrap** e **AngularJS**.

Para a aplicação ter valor a nível comercial, vão ser adicionadas funcionalidades que a tornam mais inteligente e dinâmica. Pode consulta-las no ponto seguinte.

3 Funcionalidades

Dentro do escopo de desenvolvimento da **CALIDRIS**, foram identificadas funcionalidades de acordo com os utilizadores da plataforma, que a nível de engenharia de software, são chamados de atores. Portanto, irão haver 3 tipos:

- **Entidade**, o ator que faz o *picking*, isto é, as encomendas;
- **Armazenista**, o ator que é responsável por toda a atividade em armazém, nomeadamente, o *putaway*;
- **Administrador**, o ator que faz o controle da atividade dos outros atores.

Para satisfazer as necessidades e melhorar a experiência de navegação na plataforma deste conjuntos de atores, vão ser criadas as funcionalidades mencionadas, em seguida.

3.1 Entidade

- **Login**, cada entidade terá um par de credenciais associada. Esta associação é feita pela administração da CALIDRIS, não permitindo assim o registo na plataforma.
- **Criar Encomenda**, será permitido às entidades criarem as suas encomendas, escolhendo os artigos disponíveis e as suas quantidades.
- **Visualizar informação das suas encomendas**, a entidade poderá visualizar a informação das encomendas feitas desde sempre, ordenadas cronologicamente. Informação como o estado da encomenda, os artigos, as suas referências, a quantidade de encomenda, o preço de artigo por unidade e total, entre outras características.
- **Visualizar catálogo com artigos**, irá ser possível visualizar a listagem de todos os artigos disponíveis na plataforma para encomendar.
- **Editar definições da entidade**, a entidade terá a possibilidade de editar as suas preferências na nossa plataforma.
- **Eliminar encomenda**, a entidade poderá eliminar uma encomenda feita num prazo não superior a 24H.
- **Editar encomenda**, a entidade poderá editar uma encomenda feita num prazo não superior a 24H. Será permitida a edição dos artigos encomendados e suas respetivas quantidades.

3.2 Armazenista

- **Visualizar informação das encomendas**, o armazenista irá ter acesso às ordens de compra das várias encomendas feitas para aquele armazém.
- **Gerir Stock**, terá a possibilidade de atualizar o stock em casos, por exemplo, de *putaway* e estrago de artigos.
- **Gerir Inventário**, poderá acrescentar artigos novos e as suas quantidades em *slots* destinadas a produtos novos, e eliminar artigos que deixem de fazer parte do catálogo.
- **Procurar slots em que se encontram os artigos de uma encomenda**, procurar no sistema *slots* correspondentes aos artigos das encomendas daquele armazém.
- **Aprovar/Reprovar encomenda**, o ator depois de verificar todas as *slots* correspondentes a uma encomenda poderá aprová-la, caso existam todos os artigos em stock, ou reprová-la caso falte algum artigo em stock.
- **Calcular rota**, poderá pedir ao sistema o cálculo da rota tanto para verificar se a encomenda pode ser aprovada, como a rota mais eficiente para fazer *putaway* em caso de aprovação da encomenda.

3.3 Administrador

- **Adicionar entidades**, será responsável por criar as credenciais para cada entidade e assim permitir a sua atividade na nossa plataforma.
- **Visualizar e comparar estatísticas**, poderá ver e comparar estatísticas das várias entidades a nível de encomendas, rapidez de resposta a essas encomendas, atividade de armazenistas, alterações de inventário e stock, entre outros.
- **Controlar entidades e sua atividade**, verificar se a conduta feita pelos armazenistas e entidades é praticável, para tomar as devidas precauções caso não sejam.
- **Aprovar pedido de alteração de stock**, o administrador poderá aprovar alterações feitas ao stock.
- **Aprovar pedido de alteração de inventário**, o administrador poderá aprovar alterações feitas ao inventário.

Para que estas funcionalidades sejam adicionadas à **CALIDRIS**, será necessário primeiro a construção da base da plataforma e definir a sua arquitetura que será abordado, no próximo ponto.

4 Arquitetura

É necessário fazer a pré-arquitetura do sistema para que a aplicação seja construída de forma eficaz. Portanto, os próximos pontos vão abordar as tecnologias que vamos utilizar e o seu porquê, a estrutura da base de dados alocada no Primavera, e, por fim, as *cores views* e a sua descrição.

4.1 Tecnologias utilizadas

Para o desenvolvimento do software, irão ser usadas várias tecnologias, tanto a nível de servidor como cliente.

A nível de servidor irá ser usado *Ruby On Rails*, uma vez que nos permite de forma **simples** e **segura** fazer toda a comunicação entre o cliente e o Primavera, onde se encontrará alocada a base de dados. O ERP fará, então a ligação entre a base de dados e os webserver, que serão implementados em *Ruby On Rails*.

Quanto ao lado do cliente, usaremos *Twitter Bootstrap*, como *framework* de CSS, uma vez que já inclui elementos (tabelas, menus, gráficos, etc.) capazes de permitir uma boa experiência na utilização da plataforma, assim como, facilita o desenvolvimento responsivo da plataforma. Usaremos também *AngularJS* com o principal objetivo de termos páginas o mais dinâmicas possíveis, isto é, a alteração do estado das encomendas, da quantidade de ordens de compra/venda esteja sempre o mais atualizada possível.

4.2 Base de Dados

No Primavera irá estar alocada a base de dados que irá guardar toda a informação da aplicação: registo dos clientes que poderão fazer encomendas na nossa plataforma; registo dos armazenistas; registo dos administradores; registo registo das encomendas feitas; registo das ordens de venda e compra; registo dos artigos disponíveis; registo dos artigos em stock; registo das rotas; e, ainda registo das estatísticas.


Iremos tentar armazenar o máximo possível de informação, para obter estatísticas viáveis que nos permitam tomar decisões bem pensadas e estruturadas com base em bons resultados.

4.3 Core Views

As *core views* serão apenas duas, a de *picking* e a de *putaway*. Decidimos que apenas estas *views* é que são responsáveis pelo grande valor a nível comercial da nossa aplicação, e que as outras são *views* com funcionalidades adjacentes que melhoram o produto mas que não são prioritárias.


4.3.1 Picking

Como é possível observar, a primeira é a de *picking* onde podemos visualizar um menu com as *tabs* para a entidade utilizar, do lado esquerdo. E do lado direito, além de podermos ver a listagem de encomendas feitas e o seu estado podemos criar novas. Sendo que em cada encomenda, será possível selecionar o número de itens, os tamanhos, entre outras características; e adicionar os artigos à encomenda final.



Rui Meng
Zara

- Nova Encomenda
- Estado das Encomendas 4/10
- Armazenadores
- Dados
- Definições


Log out

Encomendar

Nova Encomenda / Nova

Nome do Produto	Fornecedor	Preço	Tamanhos	Estado
BOTIM RASO C/FIVELA	Irmãos Martins Lda	50.00€	36-40	Disponível

Descrição: Coleção de inverno. Botim raso cinzento. Materiais combinados. Fecho de fivela. Altura do tãço: 2,5 cm.

1 Botim

50 Botim

1

13

26

38

50

Selecione a quantidade que deseja encomendar

Prazo de Entrega: 10/20/2015 a 10/29/2015

Mensagem: Insira mensagem adicional:

Inserir tags: TRF x Calçado x Couro

TOTAL 30.00€

Adicionar

Produto	Referência	Custo
CASACO PÊLO	3811/206	6995.00€
VESTIDO BORDADO	0881/233	6495.00€
CALÇÕES NÓ	7901/320	3995.00€

[+info](#)
[editar](#)
[apagar](#)

[+info](#)
[editar](#)
[apagar](#)

[+info](#)
[editar](#)
[apagar](#)


TOTAL
17485.00€

[Finalizar](#)
[Cancelar](#)

Copyright Calidris © 2015-2016


4.3.2 Putaway

Em relação ao *putaway* será possível o armazenista ter acesso às *tabs* de gerências de stock e inventário, listagem de encomendas, pesquisas de slots, criar rotas e alterar as suas definições, no menu do lado esquerdo. Quanto ao lado direito, será possível ver os detalhes da encomenda, a entidade, o prazo de entrega, e será permitido calcular a rota de verificação para poder proceder à aprovação da encomenda e preparar os artigos para a exportação.



João Fonseca
Irmãos Martins LDA

- Estado das Encomendas
- Gerir Stock
- Gerir Inventário
- Pesquisar Slots
- Rotas
- Definições


Log out

Estado das Encomendas

Encomenda / Aprovar

Encomenda #12

CLIENTE: Zara

PRAZO DE ENTREGA: 20 OUT - 29 OUT 2015

Selecionar Rota: Verificação de encomenda

Produto	Referência	Quantidade	TOTAL
CASACO PÊLO	3811/206	500	6995.00€
VESTIDO BORDADO	0881/233	500	6495.00€
CALÇÕES NÓ	7901/320	350	3995.00€

TOTAL
17485.00€

[Aprovar](#)
[Não Aprovar](#)

Copyright Calidris © 2015-2016

5 Interoperabilidade com o Primavera

Em seguida irão ser apresentados os *webservices* que achamos necessários para fazer a integração com o **Primavera**, um software ágil e intuitivo que, neste caso, irá permitir a gestão rápida e eficiente das encomendas, o controlo simples de stocks dos vários armazéns, assim como controlo dos clientes, e outros agentes agregados ao nosso ERP.

5.1 Obter Encomenda

WebService ID	GET_ORDER
Description	Obter encomenda
Method	GET
Routes	/orders/<id_order>
Parameters	id_order : identificador da encomenda
Input Example	/orders/id_order=12345/
Expected Output	JSON com os detalhes da encomenda

5.2 Obter Encomendas

WebService ID	GET_ALL_ORDERS
Description	Obter todas as encomendas de uma entidade
Method	GET
Routes	/orders/<id_entity>
Parameters	id_entity : identificador da entidade
Input Example	/orders/id_entity="zara"/
Expected Output	JSON com os detalhes das várias encomendas

5.3 Criar Encomenda

WebService ID	MAKE_ORDER
Description	Entidade faz encomenda.
Related Core Views	PICKING
Method	POST
Routes	/orders/new_order
Parameters	id_entity : identificador da entidade articles : array com os vários artigos e as suas informações (quantidade, referência, ...)
Input Example	/order/id_entity="zara"&articles=shopping_articles/

5.4 Aprovar Encomenda

WebService ID	APROVE_ORDER
Description	Aprovar encomenda no armazém
Related Core Views	PUTAWAY
Method	POST
Routes	/order/<id_order>
Parameters	id_armazem: identificador do armazém aprove: variável 0 (não aprova) ou 1 (aprova)
Input Example	/order/id_armazem=4 & aprove=1/

5.5 Obter Inventário

WebService ID	GET_INVENTORY
Description	Obter todos os artigos disponíveis para <i>picking</i>
Related Core Views	PICKING
Method	GET
Routes	/inventory/
Expected Output	JSON com os detalhes dos vários artigos

5.6 Obter Artigo Por Categoria

WebService ID	GET_INVENTORY
Description	Obter todos os artigos disponíveis para <i>picking</i> por categoria
Method	GET
Routes	/inventory/<category>
Parameters	category: categoria de artigos
Input Example	/inventory/category="sapatos"
Expected Output	JSON com os detalhes dos vários artigos de uma certa categoria

5.7 Obter Rota

WebService ID	GET_ROUTE
Description	Obter Rota para verificar possibilidade de encomenda e para ser realizado o <i>putaway</i>
Method	GET
Routes	/routes/
Parameters	id_armazem: identificador do armazém slots: array com a posição de entrada e das slots de passagem
Input Example	/routes/id_armazem=4 & slots=stopings/
Expected Output	JSON com os detalhes da rota

5.8 Atualizar Stock

WebService ID	UPDATE_STOCK
Description	Atualizar <i>stock</i> após <i>putaway</i> , mais stock a chegar ao armazém ou devido a outro problema, como artigos estragados.
Related Core Views	PUTAWAY
Method	POST
Routes	/stock/
Parameters	reference: referência do artigo quantidade: quantidade a retirar ou adicionar à <i>slot</i> id_armazem: identificador do armazém
Input Example	/stock/reference=174830275 & quantidade=3 & id_armazem=4/

5.9 Atualizar Inventário

WebService ID	UPDATE_INVENTORY
Description	Atualizar inventário no caso de haver uma artigo novo a ser adicionado.
Method	POST
Routes	/inventory/
Parameters	reference: nova referência do artigo id_armazem: identificador do armazém
Input Example	/inventory/reference=174830275 & id_armazem=4/

5.10 Aprovar Atualização de Stock

WebService ID	APROVE_NEW_STOCK
Description	Aprovar atualização de stock
Method	POST
Routes	/order/aprovação
Parameters	id_armazem: identificador do armazém aprove: variável 0 (não aprova) ou 1 (aprova)
Input Example	/order/id_armazem=4 & aprove=1/

5.11 Aprovar Atualização de Inventário

WebService ID	APROVE_NEW_INVENTORY
Description	Aprovar atualização de inventário
Method	POST
Routes	/inventory/aprovation
Parameters	id_armazem: identificador do armazém aprove: variável 0 (não aprova) ou 1 (aprova)
Input Example	/inventory/id_armazem=4 & aprove=1/