Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Cross docking features in a VMI software

Tiago Ferreira Monteiro

Leiria, setembro de 2018

Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Cross docking features in a VMI software

Tiago Ferreira Monteiro

Relatório de Mestrado realizado sob a orientação do Professor Vítor Manuel de Oliveira Pegado de Noronha e Távora, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e do Professor Pedro Miguel Cardoso Gago, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, setembro de 2018

Esta página foi intencionalmente deixada em branco

Dedicatória

Dedico este trabalho à minha família

que sempre me apoiou no dia a dia

em todas as minhas batalhas

Esta página foi intencionalmente deixada em branco

Agradecimentos

Aproveito esta secção para dirigir uma palavra de agradecimento a todos aqueles que me ajudaram ao longo deste percurso e que tiveram um papel fundamental na conclusão deste mestrado.

Em primeiro lugar, quero agradecer aos meus orientadores pela disponibilidade e pelo esclarecimento de dúvidas na realização deste documento.

Quero expressar o meu agradecimento à entidade de acolhimento que reuniu as condições necessárias para o desempenho das minhas atividades e a todos os colaboradores com quem tive a oportunidade de trabalhar neste período e pela ajuda prestada sempre que foi necessário.

Quero deixar uma palavra de agradecimento a todos os colegas com quem tive o privilégio de conhecer e fraternizar neste período, especialmente aos membros da Interlog Solutions com quem mais tempo passei diariamente e que também me ajudaram e motivaram nesta jornada.

Por último, quero agradecer a toda a minha família que me deu desde sempre as condições necessárias para continuar a estudar. Estarei para sempre grato por todo o seu apoio e motivação.

Esta página foi intencionalmente deixada em branco

Resumo

O presente relatório visa documentar o estágio curricular realizado durante 9 meses na empresa Interlog Group, no âmbito da unidade curricular de Estágio do Mestrado de Engenharia Informática – Computação Móvel da Escola Superior de Tecnologia e Gestão pertencente ao Instituto Politécnico de Leiria.

O objetivo principal proposto para o estágio foi o de modelar e implementar novas funcionalidades para gerir o processo de *cross docking* num *software Vendor Managed Inventory* (VMI), utilizando a linguagem de programação Java.

Num *software* VMI o fornecedor possui a capacidade para processar e assumir a responsabilidade pelo reabastecimento de *stocks* dos seus clientes, considerando para tal a informação referente aos níveis de *stocks* e as vendas dos clientes.

Ao longo do estágio pretendia-se também consolidar e aprofundar os conhecimentos adquiridos ao longo do percurso académico, bem como melhorar as competências sociais e as relações interpessoais.

Palavras-chave: *Stock* gerido pelo fornecedor, Aplicação de gestão partilhada de *stocks*, Aplicação baseada em Spring, Funcionalidades de *Reporting*

Esta página foi intencionalmente deixada em branco

Abstract

This report aims to document the 9-month internship at Interlog Group, within the scope of the Master's Degree course in Informatics Engineering - Mobile Computing at the Higher School of Technology and Management belonging to the Polytechnic Institute of Leiria.

The main objective proposed for the internship was to model and implement new functionalities to manage the cross-docking process in Vendor Managed Inventory (VMI) software, using the Java programming language.

In a VMI software, the supplier has the capacity to process and take responsibility for the replenishment of stocks of its customers, considering the information regarding stock levels and sales of customers as well.

During the internship, it was also intended to consolidate and deepen the knowledge acquired throughout the academic course, as well as to improve social skills and to be able to form new social relationships.

Keywords: *Vendor Managed Inventory, Application of Shared Supply Management, Spring based Application, Reporting Features*

Esta página foi intencionalmente deixada em branco

Lista de Figuras

Lista de Tabelas

Esta página foi intencionalmente deixada em branco

Lista de Siglas

AJAX	*Asynchronous JavaScript and XML*
API	*Application Programming Interface*
BMI	*Buyer Managed Inventory*
CMI	*Co-Managed Inventory*
CRUD	*Create, Read, Update, Delete*
DC	*Distribution Centre*
DSD	*Direct Store Delivery*
DTO	*Data Transfer Object*
EDI	*Electronic Data Interchange*
EJB	*Enterprise JavaBeans*
FMCG	*Fast-moving consumer goods*
GUI	*Graphical User Interface*
GWT	*Google Web Toolkit*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
JAR	*Java Archive*
Java EE	*Java Platform, Enterprise Edition*
JAX-RS API	*Java API for RESTful Web Services*

JPA	*Java Persistence API*
KPI	*Key Performance Indicator*
MVC	*Model View Controller*
OCS	*Order Control System*
ORM	*Object-relational mapping*
POM	*Project Object Model*
POS	*Point of Sale*
PDF	*Portable Document Format*
REST	*Representational State Transfer*
SaaS	*Software as a Service*
SCM	*Supply Chain Management*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
UI	*User Interface*
VMI	*Vendor Managed Inventory*
VRI	*Vendor Replenished Inventory*
XML	*eXtensible Markup Language*

Indice

Capítulo 1

Introdução

Num mundo cada vez mais tecnológico e competitivo, muitas empresas ligadas ao setor das cadeias de abastecimento (*supply chains*) procuram adaptar-se às constantes mudanças do setor de forma a obter uma vantagem competitiva sobre os seus concorrentes. Um dos fatores essenciais para melhorar a eficiência neste negócio está relacionado com a partilha de informação, e com a colaboração entre parceiros estratégicos de forma a reduzir custos e acrescentar valor às partes envolvidas [1].

Várias publicações e estudos [1] [2] identificam os benefícios de se contruírem relações muito próximas e duradouras entre as empresas do setor das cadeias de abastecimento, sendo crucial a partilha da informação através de meios eletrónicos. Para tal, torna-se necessário que as aplicações informáticas estejam preparadas para dar resposta a essas necessidades dos seus clientes, nomeadamente na troca de dados e respetiva integração nos sistemas.

Na década de 80, do século passado, surgiu o conceito de *Vendor Managed Inventory* (VMI) que permitiu aos fornecedores efetuarem a gestão de *stocks* dos seus clientes. As empresas Procter & Gamble e Walmart beneficiaram bastante com a adoção dessa nova metodologia [3] [4].

No âmbito do estágio foi efetuada a modelação e a implementação de várias funcionalidades num *software* de VMI, denominada *Order Control System* (OCS), que suporta um sistema de gestão de reabastecimento gerido pelo fornecedor. Neste documento encontram-se descritas as várias tarefas desenvolvidas ao longo do estágio.

1.1 Motivação e Objetivos

Existiram vários motivos pelos quais se optou pela candidatura ao estágio proposto pela entidade de acolhimento, a empresa Interlog Group. Em primeiro lugar, a temática e a própria descrição do estágio pareceram ser aliciantes assim como alguns dos conceitos aí mencionados.

Para além desses motivos, existia também um grande desejo de aplicar e consolidar os conhecimentos em algumas ferramentas e tecnologias abordadas ao longo do percurso académico, bem como o de trabalhar com novas ferramentas. Outro fator relevante foi o facto de se tratar de um estágio renumerado.

De um modo geral, o objetivo principal do estágio era o de implementar novas funcionalidades num *software* VMI, denominado OCS e implementado com a linguagem de programação Java.

Antes de se iniciar o estágio, com a duração de 9 meses, foi efetuado o respetivo cronograma com o planeamento inicial das atividades, que se encontra apresentado no Anexo 1 e da Tabela 4 em anexo. Ao longo do estágio, e em função das prioridades definidas pela entidade acolhedora, foram efetuadas algumas alterações a esse planeamento.

Os principais objetivos que se pretendiam alcançar com o estágio foram os seguintes:

- Correção e desenvolvimento de testes unitários;
- Desenvolvimento de várias funcionalidades de *Reporting* e de análise;
- Implementação de funcionalidade para Cálculo Min Max;
- Criação de ficheiros PDF;
- Colocar em produção as funcionalidades desenvolvidas.

1.2 Entidade de Acolhimento

Criada no ano 1999 em Orleans, França, a Interlog surgiu como fornecedora de serviços de auditoria e de gestão e processamento de pagamentos de empresas norte americanas. Desde então, a Interlog cresceu, espalhando-se por outros continentes, agrupando mais valências, que culminou com a geração de 3 grupos de atividade, passando a designar-se Interlog Group. [5][6] Esses 3 grupos de atividade são:

- Interlog Services: Grupo especializado no controlo e análise de faturas e dados de transporte, permitindo a análise de despesas e gestão do orçamento de transporte; [5]

- Interlog Logistics: Valência especializada na gestão de operações da *supply chain*. Esta gestão tem como objetivo ajudar a melhorar a eficiência das organizações permitindo obter uma redução de custos, através do melhoramento das taxas de serviço e da organização dos transportes; [7]

- Interlog Solutions: Grupo com competências no desenvolvimento de *software*. As soluções desenvolvidas são adaptadas às necessidades dos seus clientes, de acordo com os seus modelos de negócio e são disponibilizadas via modo *Software as a Service* (SaaS). [8]

A Interlog Group possui escritórios em 7 países (França, Portugal, Estados Unidos, México, Brasil, Índia e China), tendo o estágio decorrido em Caxarias (Portugal). Neste escritório estão reunidos os 3 grupos de atividade.

1.3 Estrutura do Relatório

O presente relatório encontra-se dividido em 6 capítulos.

No Capítulo 2 é efetuado o enquadramento conceptual e tecnológico do estágio, sendo apresentada a temática do *Supply Chain Management (*SCM), e é caracterizado o conceito de VMI. São também apresentadas as principais características da anterior versão do *software* OCS que tem sido utilizado por alguns clientes. Neste capítulo são também descritas as tecnologias utilizadas no desenvolvimento deste *software*, e abordados os testes de *software*.

No Capítulo 3 é descrita a metodologia de desenvolvimento utilizada, e as respetivas ferramentas de suporte.

No Capítulo 4 é apresentada a arquitetura do programa OCS, e descritas as suas principais características e módulos.

No Capítulo 5 são descritas as novas funcionalidades implementadas no programa OCS e os testes realizados.

No Capítulo 6 é apresentada uma reflexão acerca do trabalho realizado, dos objetivos cumpridos, das dificuldades encontradas, dos conhecimentos adquiridos e do trabalho a ser desenvolvido no futuro.

Capítulo 2

Enquadramento conceptual e tecnológico

Este capítulo começa por fazer uma pequena introdução à temática do *Supply Chain Management*, e por descrever o processo de *Vendor Managed Inventory* (VMI) e a técnica de *cross docking*. São também apresentadas as principais funcionalidades da versão inicial do *software* de gestão de stocks *Order Control System* (OCS), que suporta um sistema VMI para a gestão de reabastecimento gerido pelo fornecedor.

Posteriormente, são apresentadas as principais características das tecnologias e das ferramentas utilizadas durante o estágio, nomeadamente no desenvolvimento das funcionalidades integradas na nova versão do *software* OCS.

Finalmente é abordada a temática dos Testes de *Software*, que foi umas das áreas principais em que se trabalhou durante o estágio.

2.1 *Supply Chain Management*

O termo *Supply Chain Management* (SCM) (gestão de cadeia de abastecimento, em português) surgiu no início dos anos 80, do século passado, e foi criado pelos consultores Oliver e Webber. Segundo os autores, este termo foi usado para descrever uma nova ideologia de *marketing*, que pretende efetuar a integração dos vários processos internos de negócio, e substituir as tradicionais práticas relacionadas com a gestão dos canais de distribuição. [9] [10]

Decorridos cerca de 40 anos desde que surgiu o termo SCM, não existe uma definição consensual apesar da existência de muitas opiniões e contributos. Para o termo *supply chain* existe uma definição relativamente unânime que diz o seguinte*:* "*set of three or more entities*

(organizations or individuals) directly involved in the upstream and downstream flows of products, services, finances, and/or information from a to a customer" [11].

Estes termos tornaram-se bastante populares, tal como se pode constatar pela elevada quantia de artigos publicados cujo título inclui os termos (Figura 1).

Years	Supply chain	Supply chain management
1981-1984	8	4
1985-1988	22	5
1989-1992	100	31
1993-1996	718	229
1997-2000	4 670	1 459
2001-2004	10 700	3 190
2005-2008	14 400	3 770
2009-2012	16 100	4 090

Figura 1 - Publicações científicas com os termos "supply chain" e "supply chain management" no título, adaptado de [9]

Também por volta dos anos 80, muitos retalhistas (clientes) solicitaram aos fornecedores que efetuassem a gestão dos seus *stocks* em função das vendas, dando origem a *supply chains* do tipo VMI, e ao nascimento do próprio conceito [3].

2.2 *Vendor Managed Inventory*

O *Vendor Managed Inventory* (VMI) (*stock* gerido pelo fornecedor, em português) é um processo que permite aumentar a eficiência na gestão de *stocks* cujo objetivo é o de garantir a disponibilidade nos *point of sale* (POS) (pontos de venda, em português) com os menores custos logísticos, e com os menores níveis de *stock* possíveis ao longo da *supply chain*. Estes são os fatores chave que devem ser tidos em consideração na criação de um sistema VMI. [4]

Ao contrário do que acontece num modelo tradicional de compra, onde o cliente planeia e realiza as suas encomendas, num VMI a situação é diferente. Neste sistema a responsabilidade da gestão e reabastecimento de *stocks* passa a ser diretamente do fornecedor. Deste modo, o fornecedor necessita de saber quantas unidades existem em *stock*, quantas unidades estão em trânsito (caso existam unidades a serem reabastecidas para um armazém), e quantas unidades estão a caminho das lojas. Estas informações estão detalhadas no relatório com a informação dos *stocks* (*inventory report)* que o cliente deverá enviar diariamente aos fornecedores, juntamente com o relatório de vendas (*sales report).* O envio dessa informação é efetuado

eletronicamente. Com a informação recebida do cliente, o fornecedor consegue tomar decisões acerca do momento de entrega das encomendas, os transportes mais adequados e as quantidades ideais para cada produto, para os quais existe um nível mínimo e máximo previamente definido entre as duas entidades, podendo ser alterado futuramente. A Figura 2 ilustra este processo [4].

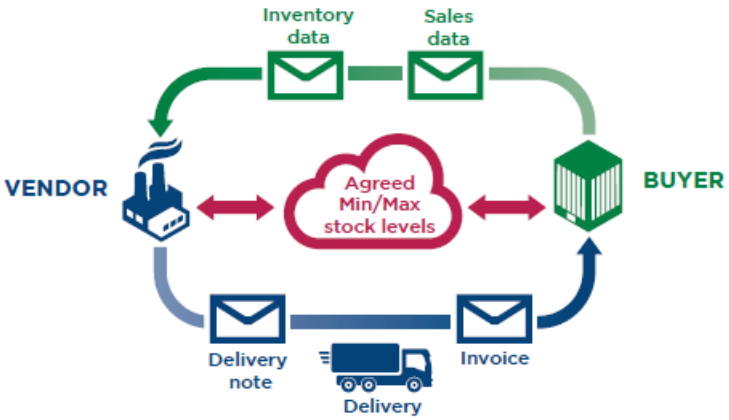


Figura 2 - Processo de reabastecimento em ambiente VMI, adaptado de [4]

Para alcançar o objetivo de aumentar a eficiência da gestão de *stocks* terá de existir um grande esforço e colaboração entre o cliente e o fornecedor, e que é superior ao existente num sistema tradicional de vendas. Por isso, os intervenientes terão de estabelecer previamente um conjunto de objetivos ou metas a atingir, com normas bem definidas de forma a não colocar em causa o sucesso do projeto. A implementação de um projeto VMI envolve a participação ativa de todos os intervenientes de forma a encontrar soluções que permitam atingir os objetivos previamente estabelecidos, por forma a existirem vantagens tanto para o fornecedor como para o cliente [4].

No que toca ao abastecimento contínuo, existem duas abordagens que divergem nos locais onde o fornecedor faz o descarregamento dos bens transportados:

- Entrega no Centro de Distribuição (DC);
- Entrega Direta no Armazém (DSD).

O DC é um local onde tipicamente chegam vários camiões com os produtos enviados pelo fornecedor, e onde se realizam operações de *cross-docking* (Figura 3). Uma operação de *cross-docking* é um processo logístico que envolve a transferência dos produtos do fornecedor para o cliente. Após ser feita a descarga dos produtos, estes ficam temporariamente no terminal de distribuição e, após uma triagem e seleção dos produtos, estes são carregados nos camiões dos clientes que se encontram na zona de saída do armazém. A regularidade do processo de

abastecimento num DC depende dos valores máximos e mínimos estabelecidos para o *stock*, pretendendo-se sempre garantir a disponibilidade e otimização de custos de transporte com o carregamento máximo permitido nos camiões [4].

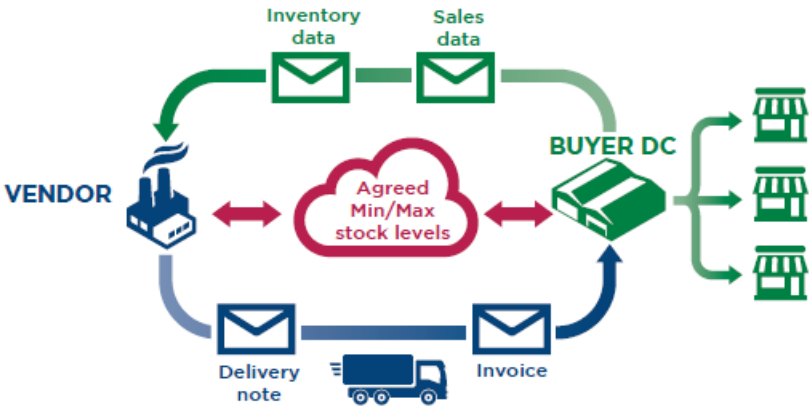


Figura 3 - Processo de reabastecimento realizado num centro de distribuição do comprador, adaptado de [4]

No modelo DSD (Figura 4) os produtos chegam diretamente à loja do revendedor, tratando-se de uma prática menos comum do que o DC. À semelhança do que acontece num armazém, a informação recolhida diariamente nos POS é enviada aos fornecedores, para que estes fiquem encarregues do processo de reabastecimento, tendo por base os valores pré acordados. Este modelo origina custos acrescidos em transportes para a entidade fornecedora, uma vez que os produtos são normalmente enviados em menores quantidades, e com maior frequência. Apesar deste facto, os produtos chegam mais rapidamente às lojas, pelo que apresentam uma elevada frescura e estão prontos para serem comercializados, o que é bom para o negócio (vendas) e poderá compensar os custos adicionais de transporte. [4]

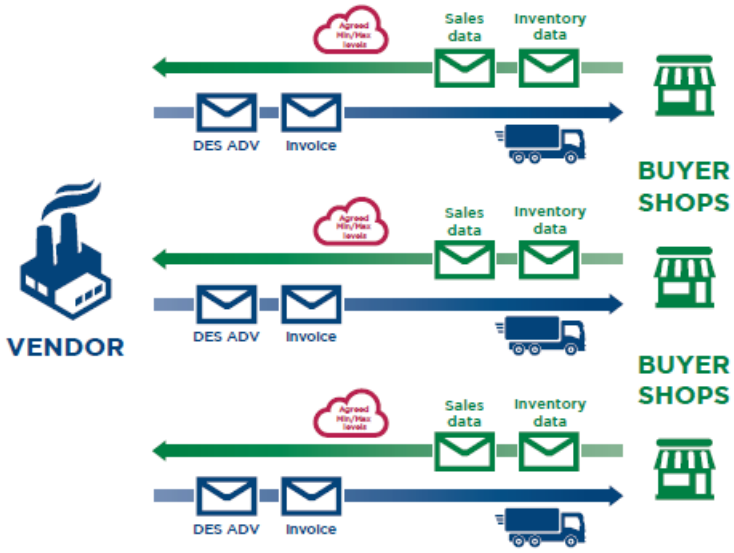


Figura 4 - Processo de reabastecimento realizado diretamente nas lojas do comprador, adaptado de [4]

Quanto aos modelos de gestão de *stock*, poderão ser considerados os seguintes quatro modelos:

- BMI (*Buyer Managed Inventory*): modelo maioritariamente utilizado no processo de reabastecimento, em que o cliente possui o controlo do *stock* e planeia as suas compras.
- CMI (*Co-Managed Inventory*): o fornecedor envia uma proposta de encomenda com quantidades definidas para cada um dos produtos. A decisão final é do cliente pois terá de aprovar ou não a encomenda.
- VRI (*Vendor Replenished Inventory*): as ordens são processadas pelos fornecedores, sem que o cliente tenha de aprovar a encomenda.
- VMI (*Vendor Managed Inventory*): As ordens são processadas pelos fornecedores sem que o cliente tenha a necessidade de aprovar a encomenda, tal como sucede no VRI. A diferença reside no facto de no modelo (puramente) VMI, as encomendas requerem um maior planeamento, nomeadamente para determinadas áreas ou períodos sazonais.

Apesar da existência destes modelos, é comum designar os modelos CMI e VRI com o termo VMI, pois o fornecedor é o responsável pelo processo de abastecimento.

Os modelos BMI, CMI e VMI apresentam algumas diferenças na forma como desempenham esse processo, tal como se pode observar na Figura 5.

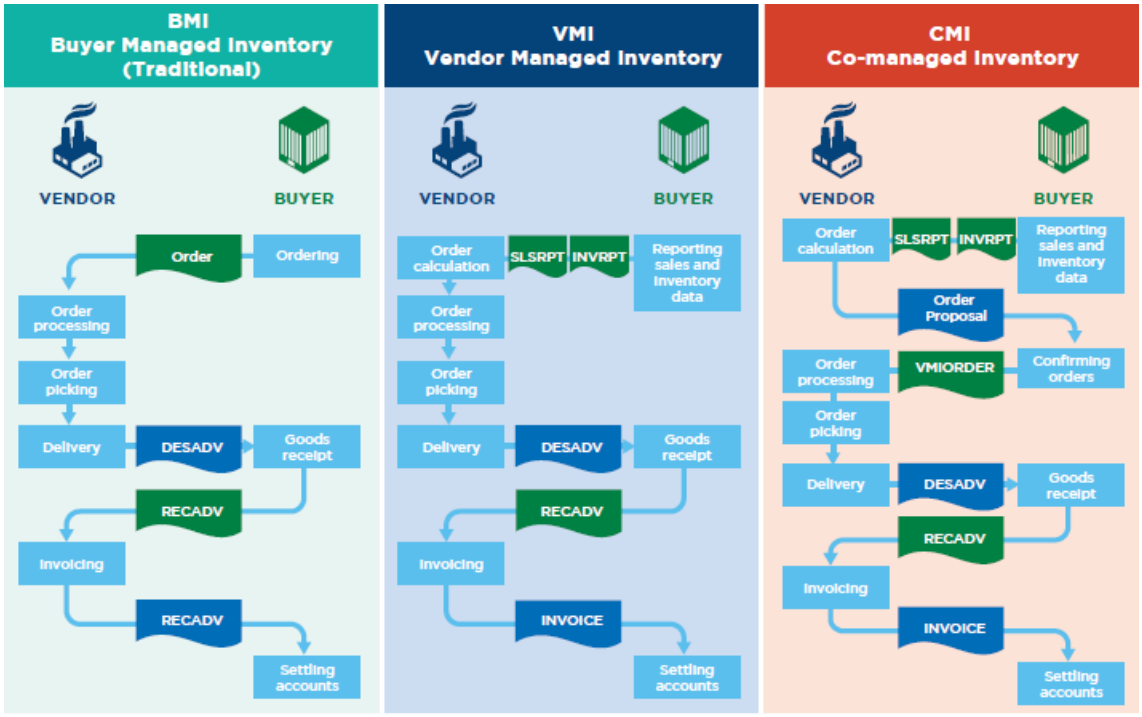


Figura 5 - Processo de funcionamentos dos modelos de gestão de stock BMI, VMI e CMI, adaptado de [4]

Dos quatros modelos apresentados (Figura 6), o VMI é a que permite uma maior redução de custos, seguido dos modelos VRI e CMI e finalmente pelo BMI [4].

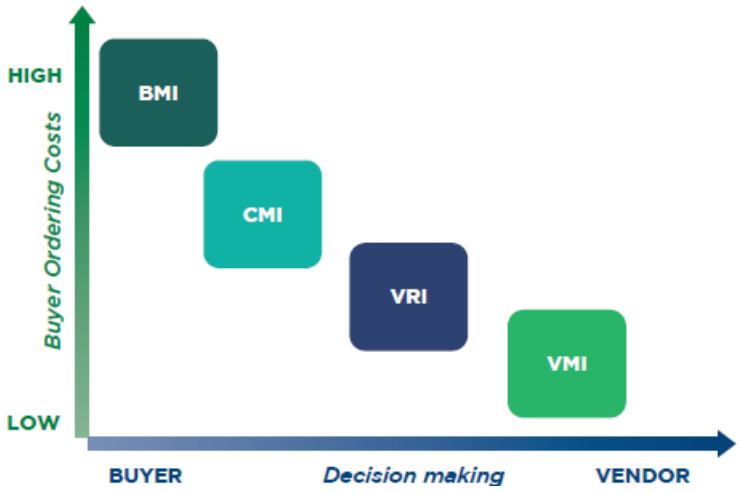


Figura 6 – Relação entre custos e capacidade de decisão para os modelos de gestão de stock, adaptado de [4]

2.3 Cross Docking

A técnica de *cross docking* consiste na descarga e no movimento da carga recebida (proveniente dos camiões dos fornecedores que chegam à entrada do centro de distribuição) para os vários camiões que se encontram à saída da estação, no menor tempo possível (Figura 7). Normalmente, à entrada a carga dos vários camiões passa por uma fase de inspeção, e os vários itens são recombinados de forma a originar cargas mais heterogéneas para cada um dos camiões que se encontram na saída do centro de distribuição [12].

Este processo é muito comum em processos logísticos, devido à sua simplicidade e aos benefícios deste sistema de distribuição. Entre eles permite:

- Reduzir custos operacionais;
- Reduzir a movimentação da carga;
- Reduzir os níveis de *stock*;
- Aumentar a taxa de serviço.

Em conjunto, todos estes benefícios proporcionam a satisfação dos clientes.

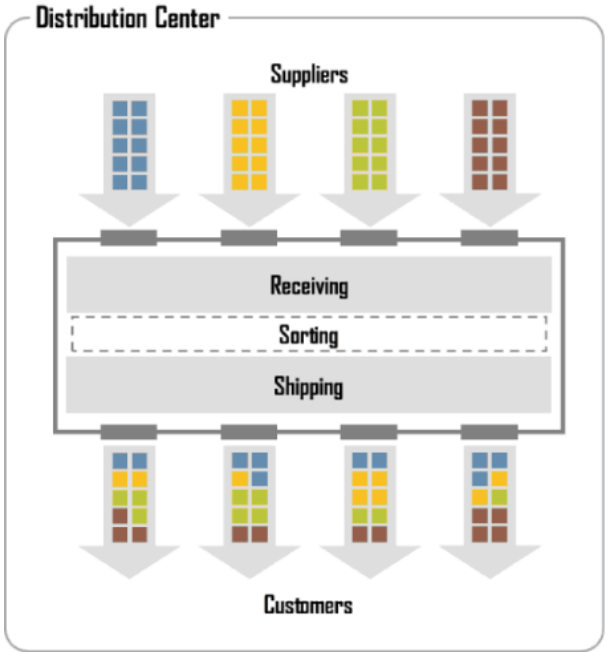


Figura 7 - Processo de cross docking num centro de distribuição, adaptado de [13]

2.4 O *software* Order Control System (OCS)

Com o objetivo de melhorar o desempenho das suas cadeias de fornecimento, a entidade acolhedora do estágio pertencente ao grupo Interlog, adquiriu o *software* OCS desenvolvido pela empresa CTS em 2016 [14].

O *software* de gestão de *stoc*ks *Order Control System* (OCS), suporta um sistema de gestão de reabastecimento gerido pelo fornecedor. Trata-se de um sistema do tipo *Vendor Managed Inventory* (VMI), e tem como objetivo principal o aumento da eficiência dos processos na cadeia de fornecimento de bens (*supply chain*).

A versão inicial do *software* OCS disponibilizava os seguintes módulos aos utilizadores:

- Operações: módulo que agrega as funcionalidades de reabastecimento dos armazéns, de que o utilizador é responsável, em função dos movimentos e das propostas de encomenda realizadas. Trata-se do módulo de maior importância para o utilizador;
- Utilidades: módulo que permite a definição de parâmetros gerais da aplicação. Apenas é disponibilizado aos administradores da aplicação;
- Interfaces EDI: módulo que regista os parâmetros dos ficheiros eletrónicos, e gere outras funções;
- Automatismo: módulo de inicialização do processo automático de integração de movimentos e de propostas de reabastecimento;
- *Reporting*: módulo de análise de *stocks* em falta, *stock* atuais e quantidades vendidas;
- Previsão: módulo que efetua a previsão das quantidades de produtos necessárias.

Esta versão inicial do OCS serviu de base para a nova versão e foi desenvolvida pelo grupo Interlog Solutions.

2.5 Tecnologias e Ferramentas

Nesta secção serão apresentadas as principais características das tecnologias e das ferramentas utilizadas durante o estágio, nomeadamente no desenvolvimento do *software* OCS.

Quando se iniciou o estágio, o *software* OCS já se encontrava em desenvolvimento, pelo que as tecnologias e ferramentas já haviam sido selecionadas. Assim, no início do estágio, foi necessário consolidar os conhecimentos, e efetuar a adaptação a essas tecnologias e respetivas ferramentas.

Inicialmente será efetuada a descrição e a evolução aos sistemas empresariais desenvolvidos na plataforma *Java Platform Enterprise Edition* (Java EE). Esta evolução abriu caminho para a criação da *framework* Spring e da *framework* Spring Boot (baseada no Spring), que se encontra a ser utilizada no desenvolvimento do OCS. Serão também apresentadas as seguintes tecnologias e ferramentas utilizadas durante o estágio: Vaadin, Maven, Jersey e Hibernate.

2.5.1 Arquitetura de um Sistema Empresarial Java EE

O *Java Platform Enterprise Edition* (Java EE) fornece uma plataforma standard para a criação de aplicações empresariais robustas e escaláveis sendo por isso utilizada em diferentes setores industriais, tais como bancos, seguros e companhias de viagens. Existe um vasto conjunto de ferramentas e tecnologias ao dispor aquando da iniciação de um novo projeto de *software* e por isso é preciso ter em consideração as que mais são adequam a cada tipo de projeto [15].

A primeira arquitetura concebida apenas continha uma camada física onde residia toda a parte lógica e dados da aplicação sendo por isso uma arquitetura *Single-Tier* (camada única). Mais tarde, com a expansão e modernização dos sistemas distribuídos, em grande parte devido à popularização do computador pessoal e do crescimento da Internet, houve o aparecimento das arquiteturas 2-*tier*, 3-*tier* e N-*tier*. Por vezes a arquitetura N-*tier* é considerada na realidade 3-*tier* devido ao facto de existir uma separação das localizações dos três componentes físicos (máquinas cliente, servidor Java EE e base de dados ou *legacy machine*) [16].

As aplicações não estão somente organizadas a nível físico, mas também são divididas em diferentes níveis lógicos. Normalmente existem as três seguintes *layers* ou camadas:

- Camada de apresentação: Responsável pela construção da interface com o utilizador (*user interface*);
- Camada de negócio: Responsável pela execução de regras de negócio;
- Camada de acesso aos dados: Responsável por recolher e manipular os dados guardados.

Na Figura 8 está representada a esquematização de uma arquitetura dividida em 3-tiers. A camada (*tier*, em inglês) do cliente é designada por *thin client*, pois apenas necessita de ter um programa (normalmente um *browser*) para apresentar a informação proveniente do servidor aplicacional. No servidor aplicacional residem as três camadas lógicas (*layers*), e por fim um sistema de base de dados relacional onde os dados são guardados [15].

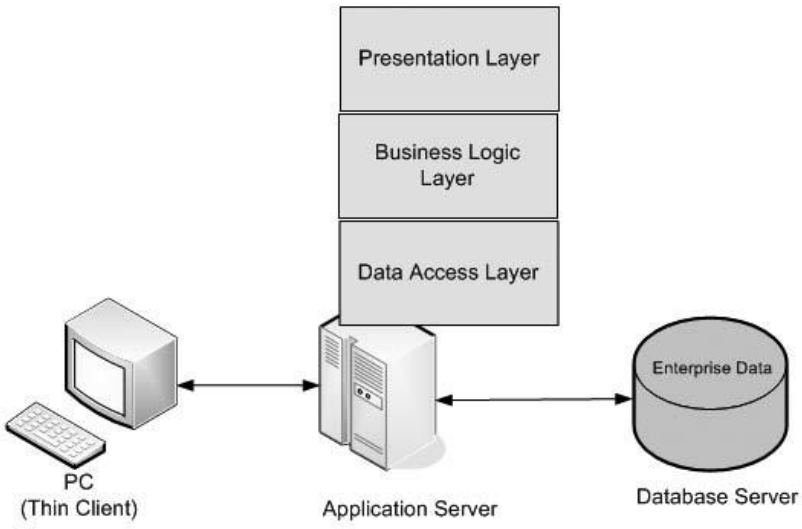


Figura 8 - Arquitetura empresarial dividida em 3 camadas, adaptado de [15]

A plataforma Java EE está dividida numa série de "*containers*" que fornecem os serviços fundamentais a qualquer projeto Java. Basicamente os "*containers*" são interfaces que permitem a ligação dos seus componentes à respetiva funcionalidade de baixo nível, e que suportam a criação de componentes Java. Os "*containers"* fornecem serviços de segurança, serviços para gestão de transações e ciclo de vida, e serviços de persistência e de comunicação para assegurar o funcionamento dos módulos do Java EE [16].

Na Figura 9 é apresentado um esquema onde residem dois *containers* do servidor: o *container Web e o container* EJB que comunicam entre si. O *container Web* detém componentes relacionados com a camada de apresentação, neste caso *Java Server Pages e Servlets*, e

comunica com o *Web Browser* usando o protocolo *Hypertext Transfer Protocol* (HTTP). O *container* EJB controla os respetivos *Enterprise Java Beans* (EJB) que armazenam as regras do negócio.

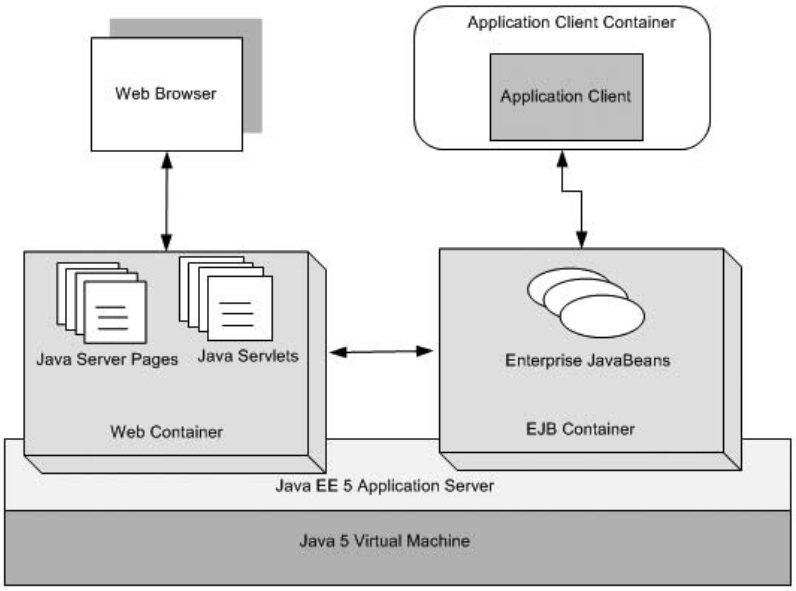


Figura 9 - Arquitetura da plataforma Java EE, adaptado de [15]

2.5.2 Spring

A *framework* Spring é a *framework* mais popular para a criação de aplicações empresariais Java e foi concebida de maneira a simplificar o desenvolvimento em Java EE. Um dos aspetos fundamentais nesta *framework* deve-se ao facto de as equipas de desenvolvimento não estarem dependentes de uma infraestrutura de suporte para a aplicação. Isto permite às equipas de desenvolvimento estarem mais concentradas ao nível da lógica de negócio das aplicações, não ficando dependentes de um ambiente de *deployment* específico [17] [18].

O Spring é uma *framework* modular o que significa que é possível tirar partido apenas dos módulos essenciais, permitindo à equipa de desenvolvimento selecionar os módulos a utilizar em cada cenário. Os cerca de 20 módulos disponibilizados estão agrupados em diferentes grupos, de acordo com as suas funções. Na Figura 10 encontram-se representados os seguintes quatro maiores grupos: *Data Access/Integration*, *Web*, *Core Container* e *Test*. Por exemplo, o grupo *Data Access/Integration* é composto por vários módulos relacionados com o acesso e integração dos dados, e cada um desses módulos apresenta características distintas dos outros pertencentes a esse grupo [18].

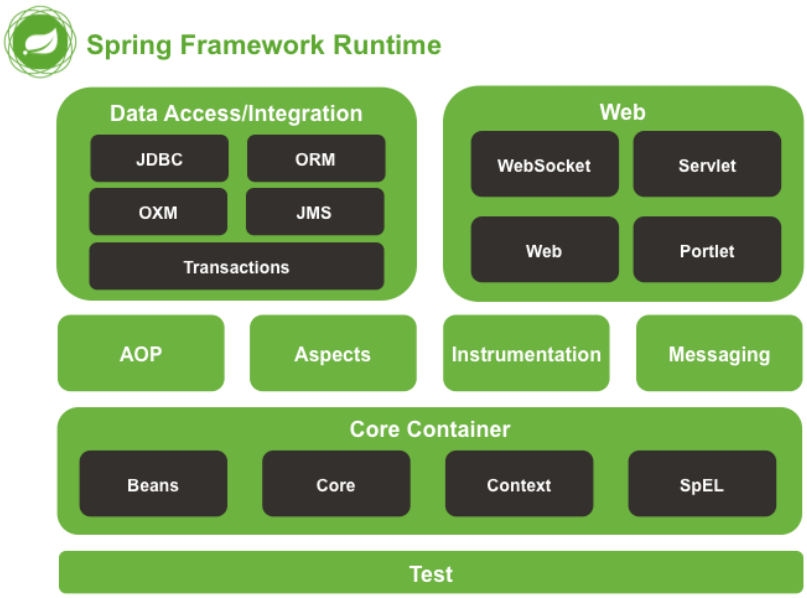


Figura 10 - Módulos da framework Spring, adaptado de [18]

A *framework* Spring suporta a integração com outras *frameworks* externas, e disponibiliza as seguintes funcionalidades oriundas dos seus módulos [17]:

- Core technologies: dependency injection, events, resources, internationalization, validation, data binding, type conversion, SpEL, AOP;
- Testing: mock objects, TestContext framework, Spring MVC Test, WebTestClient;
- Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML;
- Spring MVC and Spring WebFlux web frameworks;
- Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache;
- Languages: Kotlin, Groovy, dynamic languages.

2.5.3 Spring Boot

O Spring Boot suporta a criação de aplicações Spring para serem executadas em ambientes de produção. Esta ferramenta pretende acelerar o processo de criação da maioria das aplicações empresariais.

A criação de uma aplicação baseada em Spring Boot é mais facilitada uma vez que são utilizadas configurações da *framework* Spring, são incorporadas bibliotecas externas nas suas

dependências e permite ainda a seleção do tipo de *servlet* a utilizar (Tomcat, Jetty ou Undertow). Assim, apenas será necessário realizar uma configuração ou *setup* mínimo para a execução da aplicação. Resumidamente, o Spring Boot apresenta seguintes características principais [19] [20]:

- Disponibiliza de forma rápida e acessível uma experiência no desenvolvimento Spring;
- Fornece um conjunto de dependências iniciais para simplificar o *build* das aplicações;
- Disponibiliza funcionalidades não funcionais em projetos de larga dimensão, tais como métricas, verificações de integridade e configuração externa;
- É desprovido de mecanismo de geração de código, e não tem requisitos para configuração XML.

2.5.4 Vaadin

O Vaadin é uma *framework* Java para desenvolvimento de aplicações *web*, com objetivo de criar e gerir facilmente interfaces de utilizador (UIs) para aplicações empresariais. Esta tecnologia permite reduzir tempo e recursos na aprendizagem de tecnologias *front-end*, permitindo a concentração no desenvolvimento da lógica de negócio das aplicações. A *framework* Vaadin assenta nos seguintes pontos [21]:

- Ferramenta para criação de UIs, orientada para a construção de aplicações *web* em Java;
- Simplicidade e capacidade de manutenção;
- Criação de UIs de maneira declarativa ou dinâmicas;
- Permite criar ou reutilizar componentes quando não existirem componentes disponíveis na *framework.*

Para a construção de aplicações *web* a *framework* Vaadin (versão 8) disponibiliza os modelos de desenvolvimento *client-side* e *server-side.*

No modelo de desenvolvimento *client-side* é possível desenvolver *widgets* e aplicações em Java usando *Google Web Toolkit* (GWT), que são depois compiladas para JavaScript através do

Vaadin Client Compiler. As aplicações ficam aptas para serem executadas no *browser*, sem a necessidade de comunicação com um servidor [22] [23].

No modelo de desenvolvimento *server-side*, de maior complexidade, a lógica da UI e a escolha dos componentes gráficos é feita no *Java Vaadin Servlet* (um pequeno servidor contido num servidor Java aplicacional).

Os modelos de desenvolvimento *client-side* e *server-side* podem comunicar entre si. Nas aplicações Vaadin *server-side* existe uma interação entre o cliente e o servidor, que é efetuada através do *Vaadin Client-Side Engine*. Este é baseado em AJAX, o que permite efetuar o *render* da UI no *browser* na parte *client-side* [23].

De notar que a informação a ser integrada na UI deverá ser maioritariamente proveniente de um sistema *back-end* (onde os dados estão tipicamente armazenados). Sempre que o cliente efetuar um novo pedido ao *servlet*, uma UI será construída ou atualizada. Após receber a resposta, o cliente converte o código UI Java em código *JavaScript* através do *Client-Side Engine* que está em execução no *browser* (Figura 11) [22] [23].

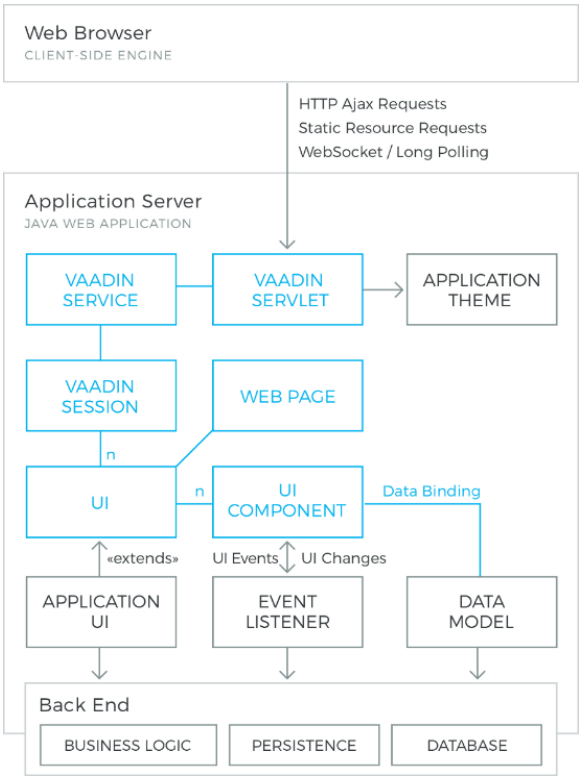


Figura 11 - Arquitetura Aplicacional da framework Vaadin, adaptado de [24]

2.5.5 Maven

O Maven é uma ferramenta que permite gerir a compilação, relatórios e documentação de um projeto. Para tal será necessário efetuar a configuração de ficheiros do tipo *Project Object Model* (POM), que permitem gerir um projeto a partir de um repositório central de informação [25].

Com o objetivo de criar standards para a compilação dos projetos, a *Apache Maven* [26] definiu um conjunto de regras que permitem identificar as dependências de um projeto, e torná-las públicas através da distribuição ficheiros do tipo *Java ARchive* (JAR) por vários projetos. Estas regras facilitam o trabalho de compreensão do estado atual de um projeto. De modo simplificado, o Maven foi criado com o intuito de: [26]

- Facilitar o processo de compilação;
- Fornecer um sistema de compilação uniforme através do uso de POMs;
- Disponibilizar informações relevantes acerca do projeto (*logs*, dependências, *reports* de testes);
- Fornecer orientações para o desenvolvimento de aplicações segundo boas práticas (Separação do código de testes, criação de nomenclaturas de teste para localizar e executar testes, criação de configurações específicas para testes);
- Permitir migrações transparentes na integração de novos recursos.

2.5.6 Jersey

O Jersey é uma *framework open source* para o desenvolvimento de *RESTful Web Services* baseados na implementação da *Java API for RESTful Web Service*s (JAX-RS API). A JAX-RS API é uma *Application Programming Interface* (API) que permite o desenvolvimento de Web Services em Java de acordo com o padrão arquitetural *Representational State Transfer* (REST). O Jersey tira partido desta API, simplificando o desenvolvimento dos Web Services ao permitir a sua exposição sem o conhecimento dos detalhes das comunicações entre o cliente e o servidor [27].

Uma API disponibiliza a outras aplicações um conjunto de serviços (web), suportando a comunicação e a partilha de dados entre aplicações. Estes serviços web podem ser acedidos através de APIs *Simple Object Access Protocol* (SOAP) ou *Representational State Transfer* (REST), que são construídas em torno do protocolo HTTP [28] [29].

2.5.7 Hibernate

O Hibernate é uma *framework Object-Relational Mapping* (ORM) que disponibiliza um mecanismo de persistência de dados, com maior foco em base de dados relacionais. Assim, permite que os dados das aplicações sejam mantidos, independentemente da duração do ciclo de vida das aplicações. As aplicações que utilizarem esta *framework* podem posteriormente aceder aos dados armazenados, pois eles não são voláteis [30].

Uma *framework* ORM é caracterizada pela capacidade de mapear dados de um objeto (a partir de um modelo) para um modelo de dados relacional, e vice-versa.

As aplicações que utilizam o Hibernate tiram proveito das suas funcionalidades para inicializar automaticamente as tabelas de uma base de dados. Quando a aplicação é inicializada, ou durante a execução de testes, o Hibernate gera grande parte do código *Structured Query Language* (SQL) de uma base de dados.

2.6 Testes de *software*

Um dos temas mais importantes no que diz respeito ao desenvolvimento de *software* está relacionado com os testes de *software* que permitem determinar ou obter um indício da qualidade de *software*. É muito importante decidir o que se quer alcançar com a produção de testes por isso cada teste deve ter um objetivo muito claro. Um erro comum é afirmar que o propósito de testar é o de encontrar o maior número de defeitos no *software*. Na realidade, um teste deve ser preparado para garantir que o programa funciona de acordo com a especificação efetuada [31].

Poderão ser realizados os seguintes tipos de testes: [31]

- Testes Unitários

  Os testes unitários, também designados testes de módulo, visam a procura de defeitos em componentes que possam ser testados de modo isolado ou separado do resto do sistema, dando um indício do seu funcionamento. Dependendo do nível de risco envolvido num módulo, este tipo de testes pode não ser desenvolvido pelo programador que efetuou a implementação, e que pode deste modo detetar mais facilmente defeitos no *software*.

- Testes de Integração

  Nos testes de integração são verificadas as interações entre diferentes componentes e partes de um sistema. Devido à complexidade deste tipo de testes, que podem ter vários níveis de integração, aconselha-se envolver uma equipa ou um membro especializado para o efeito.

  Existem diferentes modos de execução dos testes de integração. Poderão ser integrados todos os componentes e partes de um sistema, e testar o resultado da integração como um todo. As desvantagens deste modo estão relacionadas com a quantidade de tempo necessária para concluir a atividade, e com a dificuldade em encontrar as razões de falha do sistema.

  Também pode ser efetuada uma integração componente a componente, efetuando um teste por cada integração realizada. Este modo facilita a deteção de defeitos, mas é bastante dispendiosa. Em cada fase de integração os testes estão vocacionados para a comunicação entre os componentes, em oposição à sua funcionalidade.

  Dependendo da abordagem escolhida, os testes podem começar no topo de hierarquia até chegar à base da hierarquia (*top-down*), ou vice-versa (*bottom-up approach*). Recomenda-se que a integração se inicie pela componente que mais problemas pode causar ao sistema, eliminando potenciais riscos de integração futuros.

- Testes de Sistema

  Os testes de sistema pretendem analisar o comportamento do sistema de acordo com os requisitos funcionais e não funcionais previamente definidos.

Estes testes são normalmente realizados por equipas de testes especializadas e externas aos projetos, num ambiente o mais próximo possível do ambiente de produção, para se tentar assegurar que o produto a ser entregue corresponde de facto à sua especificação.

- Testes de Aceitação

  Os testes de aceitação são normalmente realizados pelo cliente com o propósito de assegurar a sua confiança na utilização do sistema, e pretendem também verificar se o sistema corresponde de facto ao seu propósito. Esta tarefa de validação das funcionalidades (*user acceptance testing*) poderá ser efetuada por gestores de projeto ou por outros colaboradores.

  Existe um tipo de testes de aceitação operacional que avalia a capacidade do sistema em tarefas muito específicas, tais como, carregamento de dados, análise de segurança, tarefas de manutenção, capacidade de backup e de restauro do sistema, e recuperação em caso de desastre.

2.6.1 JUnit

O JUnit é uma *framework open source* para codificar e executar testes de maneira repetitiva em ambiente de desenvolvimento Java. Esta *framework* é derivada da arquitetura xUnit para desenvolvimento de testes unitários, e é suportada nativamente em ambientes de desenvolvimento como o Eclipse. Com o JUnit é possível: [32]

- Escrever casos de testes;
- Partilhar dados comuns entre testes;
- Executar um conjunto de testes a partir da configuração de um *Test Runner*.

Para aumentar a qualidade do *software* produzido, a realização de testes é recomendada no desenvolvimento de aplicações. Para se determinar o resultado de um teste, há que comparar os resultados esperados com os resultados obtidos, tipicamente resultantes de chamadas a determinadas funções. Se os resultados a comparar não coincidirem com os esperados num determinado ponto, a execução do teste falha imediatamente nesse ponto. Caso contrário a análise prossegue até o final do método de teste.

Capítulo 3

Metodologia e Gestão de Projeto

Este capítulo descreve a metodologia ágil de desenvolvimento utilizada no decorrer do trabalho desenvolvido. São também apresentadas algumas ferramentas de suporte neste capítulo com as quais se interagiu.

3.1 Metodologia e o trabalho desenvolvido

De acordo com o previsto no planeamento inicial, como descrito no Anexo 1 e na Tabela 4, em anexo, o estágio iniciou-se por uma fase de ambientação ao projeto OCS em que foi estudada a documentação referente à anterior versão.

Posteriormente ocorreu a integração na equipa de desenvolvimento do *software* OCS, tendo o *Product Manager* atribuído como tarefa inicial a correção e desenvolvimento de testes (descritos na Secção 5.4). Esta tarefa inicial foi remetida num *email* em que se explicava sucintamente os objetivos da tarefa, tendo também sido efetuadas algumas sugestões de implementação. Durante esse desenvolvimento não foi solicitada uma estimativa do tempo da tarefa, pelo que se procurou iniciou de imediato o trabalho, e foram-se esclarecendo as dúvidas existentes com o *Product Manager*, através de *email* ou pelo *chat* Google *Hangouts*. Em algumas ocasiões, existiu a disponibilidade para efetuar uma revisão do trabalho e discutir alguns pormenores técnicos.

A metodologia de desenvolvimento sobre a qual decorreram as próximas tarefas, esteve mais próxima sobre os princípios ágeis da metodologia Scrum.

3.1.1 Metodologia Scrum

O Scrum é conhecido por ser uma metodologia que tem como objetivo permitir uma melhor colaboração entre membros de uma equipa e ao mesmo tempo clarificar e dividir tarefas entre o grupo. O conceito ou evento talvez importante associado à metodologia é o conceito de Sprint. Uma Sprint ou iteração (normalmente com a duração de 2 a 4 semanas), tem o objetivo daquilo que deve ser feito e um planeamento que visa guiar o desenvolvimento do produto ou incremento (Figura 12). Outro evento muito popular no Scrum está relacionado com o Daily Scrum, que basicamente é uma reunião com todos os *stakeholders* com duração média de 15 minutos onde se faz um ponto da situação atual e o planeamento das atividades a desenvolver. [33]

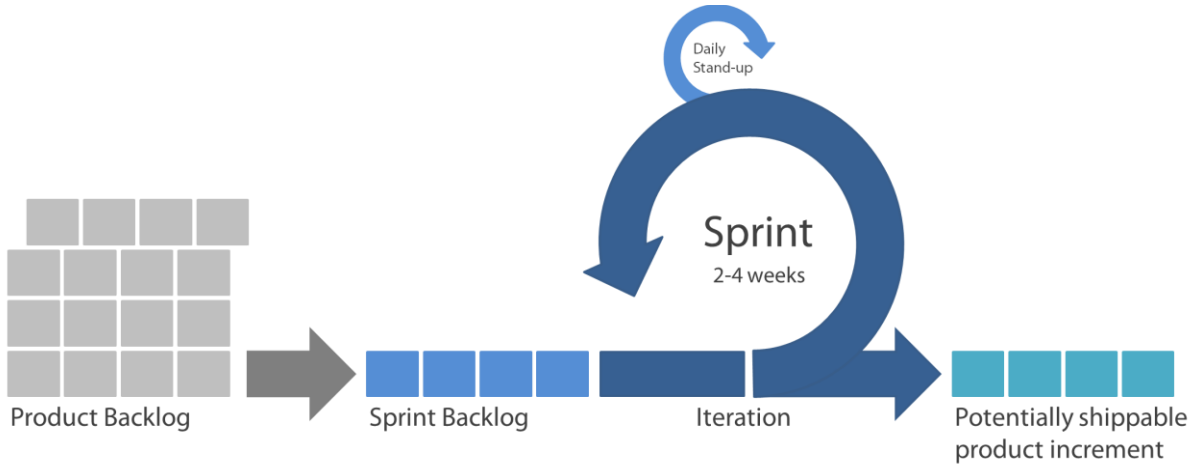


Figura 12 - Metodologia de desenvolvimento ágil Scrum, adaptado de [34]

Falando dos 3 tipos de papéis no *Scrum*, o *Product Owner* que é responsável por gerir o *Product Backlog* e por definir os objetivos da Sprint que devem ser atingidos. A partir deste ponto, é definida a Sprint Backlog que corresponde às tarefas que a equipa de desenvolvimento tem em mente acabar durante a Sprint. Já o *Scrum Master*, tem a responsabilidade de guiar e verificar que o processo Scrum é cumprido de acordo com as suas *guidelines*. Por fim, a equipa de desenvolvimento, constitui o grupo de pessoas que trabalham no sentido de entregar um produto ou incremento no final de cada Sprint, que está pronto para formar uma nova *release*. [33][35]

3.1.2 Metodologia utilizada e analogias com a metodologia Scrum

A metodologia de desenvolvimento começou a ser adquirida progressivamente, após a conclusão desta primeira tarefa, com a introdução da primeira funcionalidade de *Reporting* no OCS. A partir deste momento a equipa de desenvolvimento passou a ser composta por 3 elementos. Neste contexto, os papéis de *Product Owner* e *Scrum Master* ficaram a cargo de um desses elementos, o gestor de projeto. apesar de, por vezes, contribuído no desenvolvimento do *software*, as maiores prioridades do gestor de projeto estavam relacionadas com a gestão do *Product Backlog*, pela distribuição destas tarefas pelos membros da equipa, e por liderar as reuniões da equipa. Os restantes membros desempenharam as funções de programador dando ênfase à produção de código de acordo com as especificações recebidas. A equipa passou também a interagir regularmente através da realização de videochamadas, utilizando a ferramenta join.me.

Foi também nesta fase que se passou a realizar todas as manhãs o *Daily Scrum* com a equipa de desenvolvimento. A duração pré-estabelecida era de 15 minutos. Nessas reuniões discutia-se o trabalho realizado no dia anterior e o trabalho planeado para o dia corrente. Estas reuniões foram importantes para a equipa ter conhecimento pois permitiam obter uma noção geral do que cada elemento ia desempenhar. Por vezes, quando existiam dificuldades no desenvolvimento ou dúvidas mais complexas, a duração da reunião era estendida. Em certas ocasiões foram agendadas e realizadas reuniões adicionais, para não alongar demasiado o *Daily Scrum*.

Sempre que possível, para se obter *feedback* por parte do gestor de projeto durante o desenvolvimento de algumas tarefas, era efetuada uma demonstração através de videochamada e partilha de ecrã. Quando eram detetadas falhas durante a demonstração, era dada prioridade máxima à resolução desses problemas. Quando não era possível realizar a demonstração para aprovação final da tarefa, ficava-se a aguardar pela realização da reunião, ou, em alternativa, o trabalho enviado para um repositório para a aprovação do *Product Manager*.

Quando não era possível realizar as reuniões, e quando alguns elementos da equipa não poderiam estar presentes, nomeadamente o gestor de projeto, a comunicação era efetuada por email e/ou através do Google Hangouts. Esta situação ocorreu com maior frequência nos últimos meses do estágio, em que o número de reuniões foi menor.

3.2 Ambiente de desenvolvimento

Os ambientes de desenvolvimento integrados, também conhecidos como *Integrated Development Environment* (IDE), são programas que servem para agilizar o processo de desenvolvimento de *software*. Normalmente cada IDE contém um editor de código, um compilador e um *debugger* (que auxilia o programador na deteção de erros) acessíveis a partir de uma *graphical user interface* (GUI) [36].

A equipa de desenvolvimento utilizou o IDE Eclipse Java EE - versão Neon.3 Release 4.6.3 (Figura 13), que é uma das ferramentas *open source* mais populares no mundo do desenvolvimento Java. Apresenta um ambiente simplista, e pode ser utlizada em projetos de várias dimensões. Baseada em componentes plug-in, esta ferramenta inclui na sua especificação ferramentas para realizar a gestão de um projeto e ferramentas para a execução de testes (neste caso foi utilizado o JUnit) [37].

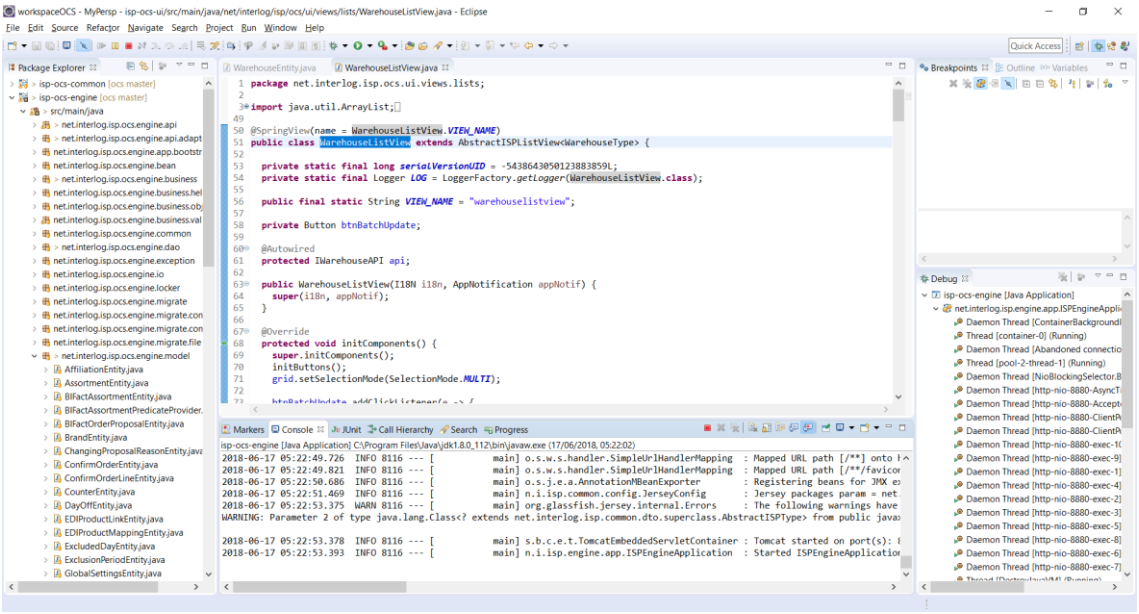


Figura 13 - IDE Eclipse

3.3 Sistema de Controlo de Versões

Um sistema de controlo de versões por definição permite controlar as alterações realizadas num conjunto de ficheiros num dado espaço de tempo [38].

O Git é atualmente uma das ferramentas mais populares para o efeito e permite ter um sistema de controlo de versões distribuído onde todos os colaboradores tem acesso a uma cópia integral

do conteúdo de um determinado repositório de um servidor. Também é distinguido pela sua fácil integração em projetos de pequena ou grande escala, eficiência e facilidade de utilização. Esta ferramenta é utilizada na empresa para a gestão dos seus projetos [39].

Relacionado com os sistemas de controlo de versões, existem determinadas aplicações para facilitar, através de uma interface gráfica, o processo de interação com o Git. Neste contexto, foram utilizadas as ferramentas *open source TortoiseGit* [40] e *SourceTree* [41] que fornecem praticamente as mesmas opções, mas de formas distintas.

O *TortoiseGit* foi utilizado maioritariamente para a resolução de conflitos entre ficheiros. Quando era necessário obter as atualizações de código mais recentes, era efetuado o '*stashing*[1]' das alterações desenvolvidas no ambiente de desenvolvimento local, para que não houvesse conflitos. Posteriormente, após as atualizações serem recebidas, era necessário integrar o código desenvolvido que estava guardado num *stash* e nessa altura era feita a resolução de conflitos usando o *TortoiseGit*.

O *SourceTree* fornece uma interface mais apropriada e eficaz para se observar de forma rápida as alterações que foram efetuadas a cada ficheiro. A maioria das vezes foi utilizado este programa para o envio das alterações, pois facilmente era verificado o conteúdo alterado em cada ficheiro, adicionando-o à lista de modificações.

A entidade acolhedora utiliza a plataforma *Easy Redmine* [42] para o alojamento dos seus repositórios Git e para efetuar a gestão dos seus projetos. De entre as suas funcionalidades destacam-se o *Resource Management*, o *Easy Project Gantt*, o *Global Gantt*, e o *Help Desk*, que são particularmente úteis para *product managers*, *scrum masters*, *team leaders* e *business owners* [43].

O *Easy Redmine* foi utilizado sobretudo pelo gestor de projeto, para o planeamento e atribuição de tarefas a cada um dos elementos da equipa. A partir das últimas semanas do estágio, a equipa de desenvolvimento começou a utilizar esta ferramenta, onde passaram a ser definidas as tarefas e os seus parâmetros.

[1] *stashing* – Processo de gravar todas as modificações de ficheiros não finalizadas numa *stack*, deixando a diretoria de trabalho limpa. É possível mais tarde voltar a aplicar as alterações gravadas. [54]

Esta página foi intencionalmente deixada em branco

Capítulo 4

Arquitetura Aplicacional

Este capítulo faz a descrição da arquitetura aplicacional do projeto OCS que segue o padrão arquitetural das 3 camadas. Essa implementação foi efetuada através da *framework* Spring Boot que está preparada para fazer uma divisão simplificada entre cada uma dessas camadas, melhorando a manutenção e consistência do código.

4.1 Características da Arquitetura Aplicacional

À semelhança do que acontece em outros projetos Java empresariais, o projeto OCS está dividido em 3 camadas claramente distintas: apresentação, lógica de negócio ou serviço e acesso a dados (representação na Figura 14).

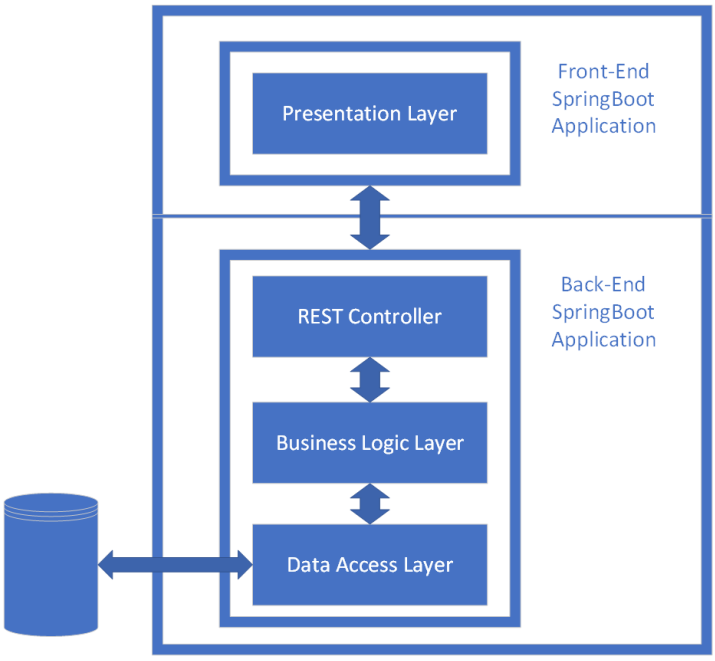


Figura 14 - Arquitetura Aplicacional do projeto OCS

A camada de apresentação encontra-se na aplicação denominada "Front-End". O código responsável das camadas de negócio e de acesso a dados reside na aplicação designada "Back-End". Na realidade existem duas aplicações Spring Boot (descrito em 2.5.3) que comunicam entre si, utilizando:

- Spring Boot – versão 1.5.4, baseada na *framework* Spring – versão 4.3.9 (mínima);
- Tomcat 8 (*servlet* disponível "*out of the box"* para a versão 1.5.4 do Spring Boot).

Para ser possível utilizar o sistema no *browser*, ambas as aplicações necessitam de um *servlet* configurado e executado em portas distintas, utilizando-se a porta 8880 para o servidor "Back-End" e a porta 8881 para o servidor "Front-End".

4.1.1 Comunicação entre as aplicações de "Back-End" e de "Front-End"

A comunicação entre as aplicações é efetuada através de um *proxy*, que funciona como um intermediário que solicita os recursos de um ou mais servidores.

Quando o utilizador se autêntica na aplicação "Front-End" é registado um *proxy* (que produz um *bean* para ser gerido pelo Spring *container*) para cada tipo de API existente.

Deste modo, a chamada a um *endpoint* específico apenas terá de ser efetuada do lado "Front-End", através da invocação do método da respetiva interface de comunicação. O *proxy* intervém no processo, criando um novo cliente JAX-RS apontando para o recurso (URI) especificado, através do qual constrói uma nova abstração desse recurso, pronta para ser invocada. Na prática, através da especificação do URI da aplicação "Back-End", é possível executar um determinado método do seu controlador REST. Quando o método do controlador devolver uma resposta, o invocador do método da interface obtém-na, e pode processá-la (com a ajuda de um método auxiliar), para obter os dados devolvidos do controlador REST.

4.1.2 Características gerais da aplicação "Front-End"

Na aplicação de "Front-End" encontra-se a camada de apresentação, que é compostas por várias UIs e foi construída em torno da *framework* Vaadin (versão 8). Nas UIs são efetuadas chamadas às APIs REST, através da utilização de proxies de comunicação, que ocorrem, por exemplo, toda a vez que um utilizador clica num botão para guardar dados ou quando o utilizador

simplesmente navega de página em página. Desta forma é possível que a aplicação "Front-End" obtenha os dados necessários para a construção da UI, e que também realize operações de atualização da base de dados.

O processo de construção das várias UIs necessárias no OCS foi facilitada com a utilização de algumas classes de um outro projeto, denominado is-platform, que utilizam e integram com a *framework* Vaadin.

O OCS também faz uso de classes definidas no is-platform para a construção do *header*, *footer* e menu lateral, numa classe do OCS própria para o efeito. Sempre que existir um evento de mudança de vista apenas será necessário carregar o conteúdo (dinâmico) que se altera de página para página, não sendo necessário carregar os todos elementos comuns.

4.1.3 Características gerais da aplicação "Back-End"

A aplicação de "Back-End", possui as seguintes três camadas principais (Figura 14):

- REST *Controller;*
- *Business Logic Layer* (ou *Service Layer*);
- *Data Access Layer*.

Nos REST *Controllers* é efetuada a implementação das APIs (REST), existindo tantos métodos quantos os que estão definidos nas interfaces. Nestes controladores são efetuadas chamadas aos diferentes métodos do seu serviço, e são o ponto de entrada para o fluxo de instruções cuja execução continuará em outras classes *back-end*. O tratamento de erros é efetuado nos métodos dos controladores, que posteriormente enviam a informação para a aplicação de *front-end* para ser apresentada ao utilizador.

A *Service Layer*, também conhecida por *Business Logic Layer*, é responsável por validar os dados a utilizar nos métodos que estão ao nível da *Data Access Layer*, e por chamar métodos de classes responsáveis para executar algoritmos e/ou tarefas de maior complexidade. Nesta camada também existem métodos responsáveis pela validação de parâmetros aquando da ocorrência de operações de CRUD (*Create, Read, Update, Delete*) nos dados.

A *Data Access Layer* serve como elo de ligação entre a aplicação e a base de dados. Nesta camada é onde reside todo o código para executar *queries* à base de dados, o que permite obter, criar, atualizar ou eliminar dados das tabelas da base de dados. As *queries* foram desenvolvidas com recurso à *framework* Hibernate (descrita na secção 2.5.7), e apresentam uma sintaxe semelhante à linguagem SQL (para a qual o código é automaticamente convertido).

4.2 Módulos do projeto OCS e os seus relacionamentos

Tal como foi anteriormente referido, o projeto OCS depende de outro projeto da entidade acolhedora, denominado is-platform. Ambos os projetos são compostos por vários módulos, e encontram-se interligados através das configurações, definidas nos ficheiros *pom.xml* que são necessárias para o Maven compilar os projetos. Na Figura 15 encontram-se representados os módulos dos dois projetos, bem como os respetivos relacionamentos (setas azuis) e dependências (setas vermelhas).

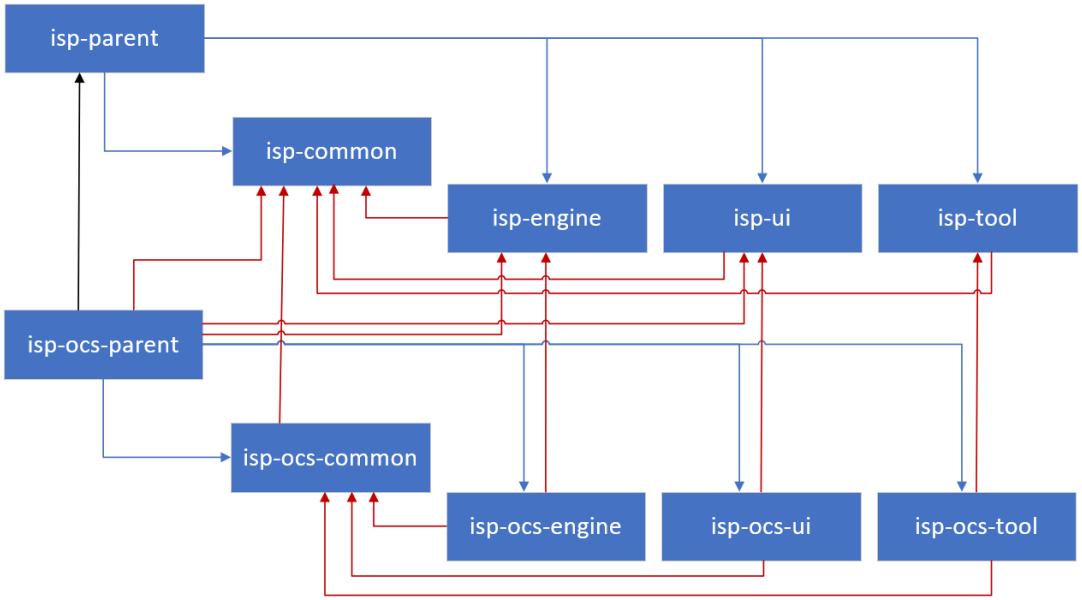


Figura 15 - Módulos do projeto OCS (mais abaixo) e do projeto is-platform (mais acima)

Entre projetos foi necessário definir que o isp-ocs-parent tem como módulo hierarquicamente superior o módulo isp-parent, definindo-se as propriedades e as relações de herança nos ficheiros POM.

Cada módulo separa as classes idênticas, ou com o mesmo comportamento, em *packages* de forma a facilitar a consulta das classes por parte da equipa de desenvolvimento. Nas secções

seguintes encontram-se sumariamente descritas as classes existentes em cada módulo do projeto OCS.

4.2.1 Módulo isp-ocs-common

O módulo isp-ocs-common contém as classes responsáveis pela definição das várias APIs REST, cujas interfaces são utilizadas na aplicação de *front-end* e as respetivas implementações na aplicação de *back-end*. Nessas classes encontram-se implementados os *Data Transfer Objects* (DTOs)[2] que são enviados e recebidos tanto no *back-end* como no *front-end*.

Este módulo possui ainda classes com a definição de constantes, códigos de erro e funções utilizadas de forma transversal no projeto. As diferentes classes deste módulo encontram-se disponíveis nos restantes módulos do projeto.

4.2.2 Módulo isp-ocs-engine

O módulo isp-ocs-engine é o responsável pelo funcionamento *core* da aplicação, contendo as classes onde se encontra implementada toda lógica de negócio e a interação com a camada de acesso aos dados, ou seja, é o módulo mais importante. As classes encontram-se distribuídas por vários *packages*.

Neste módulo encontram-se as classes responsáveis pela implementação técnica de cada API REST, as classes *Adapters* responsáveis pela conversão de DTOs num objeto do modelo de dados, ou vice-versa. Encontram-se também e as classes contendo a lógica de negócio e as classes de acesso aos dados.

4.2.3 Módulo isp-ocs-ui

O módulo isp-ocs-ui contém as classes responsáveis pela construção da camada de apresentação, utilizando para esse efeito algumas classes do projeto is-platform.

As classes deste módulo estão distribuídas por vários *packages*, de acordo com o tipo de conteúdo a apresentar. Assim, as classes que realizam a listagem da informação estão definidas

[2] DTO - Objeto que transporta dados entre processos de maneira a reduzir o número de chamadas aos métodos - [55]

num *package* definido para o efeito. As classes responsáveis pela lógica comum da criação e atualização de um formulário também estão definidas num *package* específico. Esta abordagem também foi seguida para as outras classes com o mesmo tipo de função, como, por exemplo, para os componentes Vaadin personalizados, *pop-ups* e *tabs*.

Cada uma destas classes necessita de utilizar um ficheiro HTML para determinar o posicionamento dos vários componentes que variam de UI para UI na parte dinâmica da aplicação. Essa definição é feita aquando da criação da classe.

Este módulo inclui também ficheiros CSS, imagens e ficheiros onde é efetuada a tradução de cada chave para o valor correspondente (francês ou inglês).

4.2.4 Módulo isp-ocs-tool

No módulo isp-ocs-tool foram criadas classes para a execução de comandos que efetuam chamadas a determinados métodos das APIs REST. Desta forma é possível, por exemplo, atualizar determinadas tabelas da base de dados.

Esses comandos podem ser executados a partir do ambiente de desenvolvimento, ou a partir da linha de comandos e de forma periódica (por exemplo diariamente ou semanalmente).

Capítulo 5

Implementação e Testes

Este capítulo descreve a implementação técnica das funcionalidades desenvolvidas no projeto OCS, bem como os testes realizados no âmbito dessas funcionalidades. Inicialmente é apresentado o projeto is-platform e são descritas as tarefas desenvolvidas no projeto.

5.1 Projeto is-platform

Na planificação inicial previa-se que apenas fossem efetuados testes e desenvolvidas novas funcionalidades no projeto OCS. No entanto, para se preparar para esse projeto, foi posteriormente definido que no início do estágio haveria uma fase de estudo e de teste do projeto is-platform.

O projeto is-platform têm um impacto significativo em vários outros projetos, incluindo o OCS, pois é nele que se encontram implementados, de forma genérica, os mecanismos básicos e a lógica associada à persistência de dados. Possui também uma lógica genérica para a criação de vários tipos de vistas, por exemplo formulários e tabelas, bem como alguns componentes (visuais) que podem ser reutilizados noutros projetos. Desta forma evita-se a reescrita de código em projetos que necessitem de funcionalidades já implementadas no projeto is-platform.

Foram estudados e testados os módulos de demonstração de funcionalidades do projeto is-platform que poderão ser integrados em módulos principais de outros projetos. Esses módulos (isp-demo-ui, isp-demo-engine e isp-demo-common) permitem integrar novos componentes gráficos e efetuar operações básicas ao nível da persistência de dados numa base de dados relacional.

Nessa fase foi possível perceber as funcionalidades e a forma como esses módulos se encontravam relacionados, o que permitiu também entender o funcionamento geral dos restantes módulos do is-platform e dos módulos do OCS. Para tal foram efetuadas diversas experiências com a aplicação em execução (acessível no *browser*) e observou-se o seu comportamento. Também foram analisados os *logs* gerados (provenientes da atividade do utilizador na aplicação), inseridos *breakpoints* no código, e analisadas as modificações de valores das tabelas da base de dados.

Para aprofundar e consolidar o conhecimento adquirido, foi desenvolvida uma nova funcionalidade básica. A tarefa consistiu em efetuar operações CRUD de uma nova entidade "Country", e fazer as respetivas UIs de forma semelhante às UIs existentes, utilizando a *framework* Vaadin e seguindo o padrão das 3 camadas.

Para poder efetuar o estudo e o teste da plataforma is-platform foi previamente preparado o respetivo ambiente de desenvolvimento, através da instalação e configuração das seguintes ferramentas iniciais:

- Eclipse - versão Neon (após prévia instalação do Java Development Kit – versão 8u112);
- TortoiseGit - versão 2.5.0.0;
- HeidiSQL - versão 9.3.0 (após prévia instalação do MariaDB – versão 10.1.8).

5.2 Integração do projeto is-platfom no projeto OCS

À semelhança do procedimento adotado com o projeto is-platform, o projeto OCS foi clonado a partir de um repositório. Foi também importado o ambiente de desenvolvimento (Eclipse) para um *workspace* diferente por forma a obter-se uma melhor organização e a distinção dos projetos.

Ao analisar o projeto OCS constatou-se que existiam muitas semelhanças com o projeto is-platform ao nível da organização e implementação do código em cada uma das três camadas da arquitetura.

Apesar dos projetos is-platform e OCS se encontrarem em repositórios diferentes, e serem geridos por equipas de desenvolvimento distintas, a possibilidade de existirem problemas no

projeto OCS inicialmente poderia parecer reduzida. No entanto, sempre que se pretendessem alterar alguns comportamentos no projeto is-platform, não era fácil refleti-los no OCS visto que são projetos distintos. A solução passou pela criação de um *branch* do projeto is-platform, onde é possível efetuar alterações para utilização exclusiva por parte do OCS.

Com a utilização deste *branch* do is-platform por parte do OCS, as novas alterações efetuadas na *branch master* a cada *release* do is-platform não teriam impacto no OCS. No entanto, sempre que se pretendia utilizar uma *release* mais recente do is-platform teria de se realizar um *merge* entre as alterações na *branch* master, e por consequência realizar possíveis alterações no projeto OCS.

5.2.1 Utilização de classes do projeto is-platform no projeto OCS

Sempre que possível foi reutilizado o código do projeto is-platform no OCS, nomeadamente na implementação de uma nova entidade, e respetivas vistas no sistema, para listar, criar/editar e remover dados. Nesta secção são apresentadas as principais classes envolvidas no desenvolvimento de um simples CRUD, a sua hierarquia e o módulo do projeto OCS em que se encontram.

Inicialmente é necessário criar um novo "Type" (que representa um DTO) e uma nova "Entity" (que representa uma entidade) nos módulos isp-ocs-common e isp-ocs-engine, respetivamente, e que incorporam os comportamentos definidos nas classes AbstractISPType e MDEntity definidas nos módulos isp-common e isp-engine do is-platform (Figura 16).

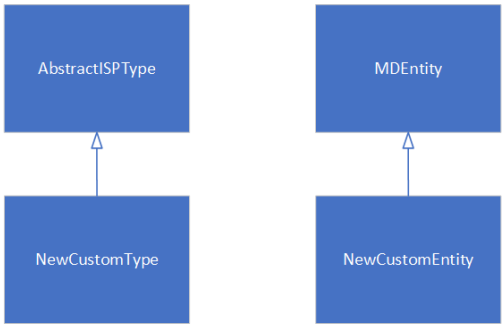


Figura 16 - Esquematização da hierarquia de classes ao nível dos DTOs e entidades para construção de um CRUD

Nos módulos isp-ocs-common e isp-ocs-engine é necessário seguir a estrutura de classes representada na Figura 17, de modo a a cumprir os requisitos de uma arquitetura de 3 camadas. As designações "<T>" e "<E>" são respetivamente referentes a "Type" e "Entity"

genéricos. A API para comunicação entre os módulos isp-ocs-engine e isp-ocs-ui é definida no módulo isp-ocs-common.

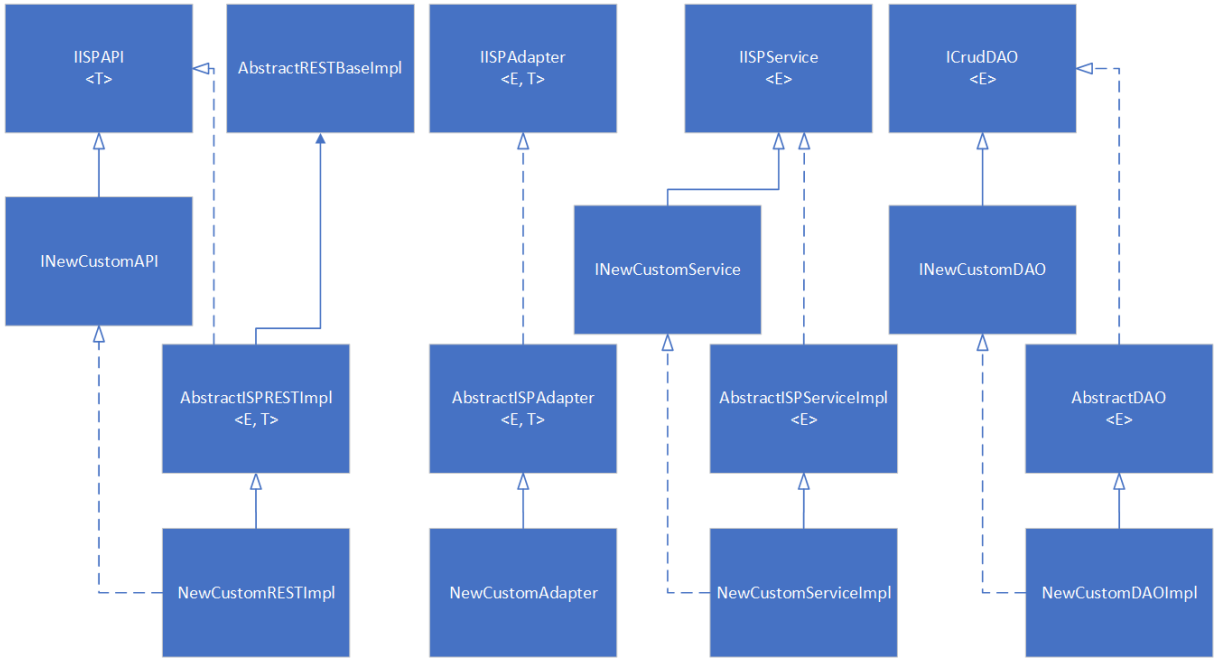


Figura 17 - Esquematização da hierarquia de classes dos módulos isp-ocs-common e isp-ocs-engine para construção de um CRUD

No módulo isp-ocs-ui é necessário seguir a estrutura de classes da Figura 18 de modo a construir as várias vistas pertencentes à camada de apresentação, com base no módulo isp-ui do projeto is-platform que integra com a *framework* Vaadin.

Figura 18 - Esquematização da hierarquia de classes do módulo isp-ocs-ui para construção de um CRUD

Terá de se criar um ficheiro HTML para indicar onde serão inseridos os vários componentes Vaadin, e que se encontram declarados em cada uma das classes acima mencionadas. Isto é feito através do atributo "location" a nível do "corpo" de uma vista genérica, ou de um formulário a reutilizar.

Quando uma nova vista for incluída no sistema, será necessário atualizar o código da classe OCSDatabaseLoader. Esta classe possui um método responsável pela criação das permissões na tabela "permissions" da base de dados, e outro método para criar os menus acessíveis da aplicação, e a sua posição. Após a atualização destes métodos, será necessário limpar todos os registos das tabelas "profiles" e "profiles_permissions", para que na inicialização seguinte do sistema sejam executados os métodos alterados, e para que as modificações sejam refletidas na base de dados.

Note-se que no momento de criação das permissões são especificados os "profiles" que terão acesso à permissão que estará prestes a ser criada, pelo que terão de ser atualizadas as tabelas "permissions" e "profiles_permissions". Cada utilizador da tabela "users" sabe o seu "profile" através da tabela "users_profiles", e só poderá aceder às vistas que o profile tiver acesso, e só poderá realizar as operações CRUD para as entidades incluídas no profile do utilizador.

5.3 Tarefas no projeto OCS

Das várias tarefas realizadas na plataforma OCS, destacam-se as relacionadas com o desenvolvimento de testes unitários e as tarefas relacionadas com a criação de análises/*reports*. Estas atividades foram as tarefas de maior duração durante estágio.

Foram também realizadas outras atividades tais como a geração de ficheiros PDF, calcular/otimizar o *stock* mínimo e máximo (em dias) para cada produto, adicionar produtos a armazéns entre outras.

Tal como sucede em todos os projetos que envolvem uma componente de implementação, foram efetuadas deteções e correções de erros (*bugs*), e/ou de pequenas falhas.

A Figura 19 permite dar uma melhor visualização do trabalho que foi realizado ao longo do tempo no projeto OCS. A Tabela 1 identifica cada uma das tarefas desenvolvidas.

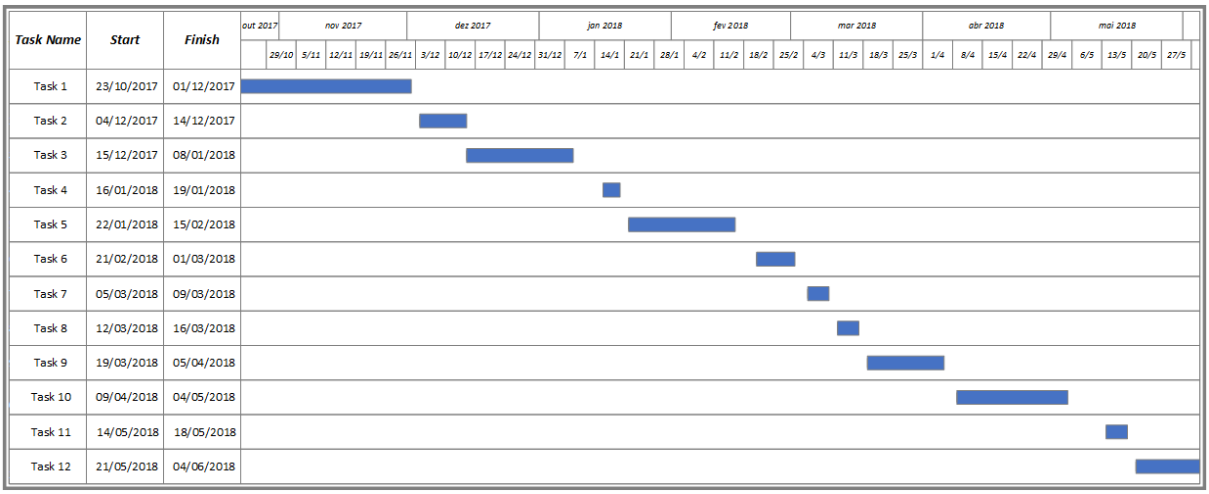


Figura 19 - Tarefas realizadas no OCS e a respetiva duração

Tabela 1 - Lista e nome das tarefas realizadas no OCS

Tarefa	**Nome da Tarefa**
Task 1	Correção e desenvolvimento de testes JUnit
Task 2	Definição de períodos
Task 3	Análise da qualidade de serviço
Task 4	Mapeamento de links
Task 5	Análise de *out-of-stock*
Task 6	Análise de cobertura média
Task 7	Adicionar produtos a armazéns
Task 8	Análise da unidade de logística ótima
Task 9	Cálculo Min Max
Task 10	Geração de ficheiros PDF
Task 11	Gestão da visibilidade dos componentes Vaadin
Task 12	Apresentação e otimização dos dashboards da página inicial

Cada uma das tarefas identificadas na Tabela 1 encontra-se descrita nas secções seguintes.

5.4 Correção e desenvolvimento de testes JUnit

No projeto OCS foram implementados e realizados três principais tipos de testes unitários na camada de acesso a dados e na camada da lógica de negócio. Foram também desenvolvidos mecanismos para suportar a execução sequencial de testes do mesmo tipo.

5.4.1 Testes realizados ao nível da camada de acesso a dados

Os testes realizados ao nível da camada de acesso a dados permitem verificar os mecanismos básicos de persistência (CRUD). Este tipo de testes é efetuado nas classes com sufixo "DAOImplTest", e visam testar o código implementado nas classes de sufixo "DAOImpl" que estão envolvidas nas diversas operações ao nível da camada de acesso a dados.

Os testes unitários desta camada, que têm dados previamente definidos num *dataset* inicial e estão escritos da seguinte maneira:

- Aquando da procura por todos os registos de uma entidade do *dataset* é verificado que a lista devolvida não é nula e que tem N elementos, em que N representa o número de registos de uma determinada entidade no *dataset* utilizado;
- Na procura por um determinado registo através do seu id é verificado que este existe no *dataset* e que os dados contidos nesse registo são consistentes com os dados esperados;
- Na criação de um registo, o programador define *à priori* todos os seus campos (à exceção do seu id) e de seguida persiste-o, sendo devolvido com um id automaticamente atribuído e é verificado que este id é válido. De seguida, é feita uma pesquisa por esse id, como descrito no ponto anterior, validando a coerência entre os dois registos;
- Na atualização de um registo, inicialmente é feita uma pesquisa, por um determinado id, para verificar se os dados obtidos são consistentes com os dados esperados. Depois são alterados alguns dos seus campos, e é efetuada uma atualização dos dados. Finalmente, é efetuada uma outra pesquisa, com mesmo id anteriormente utilizado, e é feita uma verificação dos campos alterados;
- O teste de remover é iniciado com uma pesquisa de uma entidade através de um id e é verificado que a entidade pesquisada existe. De seguida, é removida a entidade encontrada e é verificado que é devolvido o id com valor 0. Para finalizar, é feita novamente a pesquisa como foi feito no início do teste e é verificado que o registo da entidade pesquisado é nulo.

A Figura 20 mostra a execução de um desses testes unitários pertencentes ao projeto OCS.

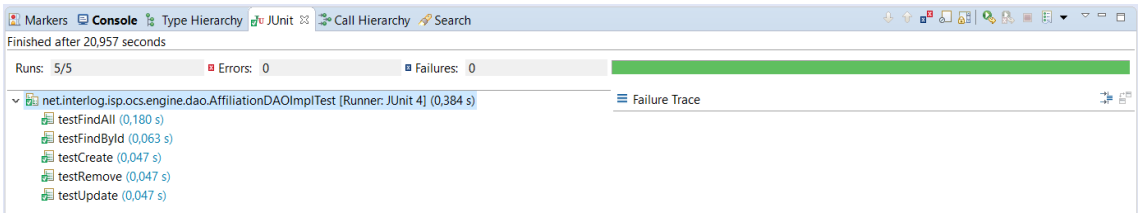


Figura 20 - Exemplo de testes unitários presentes na classe AffiliationDAOImplTest

Nos testes efetuados às classes da camada de acesso a dados foram encontrados diversos erros, sobretudo devido aos seguintes motivos:

- Conteúdo desatualizado dos *datasets* utilizados no teste, que contêm um conjunto de registos semelhante ao conteúdo da base de dados definidos nos ficheiros XML). Este tipo de erro foi o mais comum, e ocorreu quando o modelo de dados ao nível das entidades do sistema não estava em concordância com o modelo de dados do teste. Para solucionar este problema é necessário atualizar os dados dos ficheiros de teste, assegurando que se encontram refletidas todas as modificações efetuadas às entidades do sistema.

- Código de teste incorreto (desatualizado). As alterações ao nível de uma entidade podem causar erros de regressão, pelo que os testes devem ser executados na entidade alterada. Para solucionar este problema o teste terá de ser corrigido até ser aprovado.

A Figura 21 mostra resumidamente o processo aplicado a cada um dos testes de camada de acesso a dados (sufixo DAOImplTest) localizados nas classes do Anexo 2 e da Tabela 5, em anexo.

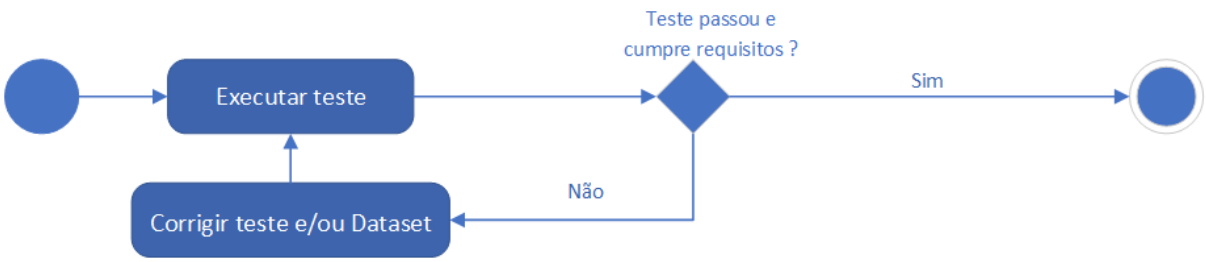


Figura 21 - Fluxograma contendo o processo de aprovação de testes da camada de acesso a dados

Foi criada uma classe para executar código comum a vários testes da camada de acesso a dados, e que foi executado pelas restantes classes de teste. Os campos das entidades que foram alvos de teste foram a data de criação de um registo, a data da última modificação, e os campos

"criado por" e "modificado por". Todos estes campos são automaticamente incluídos numa entidade sempre que ela esta "estenda" a classe MDEntity.

5.4.2 Testes de validação de input

Os testes de validação de input foram efetuados após a conclusão dos testes à camada de acesso, e pretendiam verificar se, antes de uma operação de persistência, os dados de uma determinada entidade obedeciam a determinadas regras.

Normalmente quando um erro é encontrado, por exemplo se o campo de um formulário exceder o tamanho máximo permitido, o sistema adicionará um código de erro a uma lista. Posteriormente será apresentada uma mensagem associada a esse código.

Na criação e na atualização de registos de uma entidade, normalmente as regras a verificar são exatamente as mesmas, pelo que apenas terá de ser criado um dos testes. Também existe a possibilidade de se pretender eliminar um registo, de se averiguar se um determinado registo é nulo, e ainda de verificar se uma lista de elementos está vazia.

O objetivo deste tipo de testes é simular/introduzir valores num ou mais registos, e verificar se todos os valores introduzidos respeitam ou não as regras de validação definidas. Sempre que forem inseridos valores não permitidos pelas regras, uma mensagem de erro é adicionada à lista, e, ao nível do teste, é verificado se a lista não está vazia e se tem determinados códigos de erros esperados. Se o *input* do registo estiver dentro das regras definidas, o teste apenas verifica se a lista de erros está vazia, ou seja, se se trata de um registo válido.

A Figura 22 mostra a lista de testes unitários executados definidos na classe WarehouseValidatorTest.

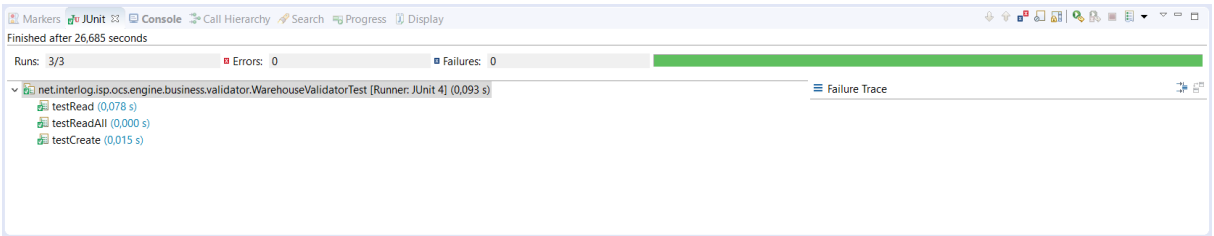


Figura 22 - Exemplo de testes unitários presentes na classe WarehouseValidatorTest

5.4.3 Testes realizados ao nível da camada da lógica de negócio

Na camada da lógica de negócio os testes unitários permitiram testar alguns algoritmos e alguns excertos de código realizados ao nível dessa camada. Estes testes normalmente necessitam de um conjunto de dados iniciais, pelo que foram criados *datasets* (ficheiros XML) da mesma maneira que nos testes de camada de acesso a dados. Os *datasets* continham dados controlados pelo programador, e que foram essenciais para se proceder à execução dos testes unitários com JUnit. A Figura 23 mostra um exemplo dos testes definidos na classe TruckLoadBusinessTest.

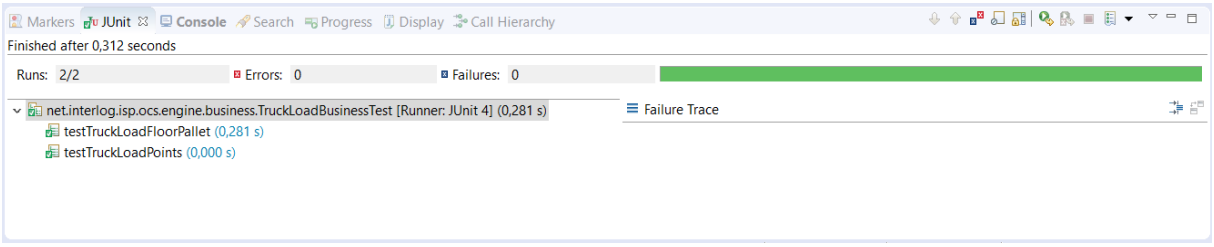


Figura 23 - Exemplo de testes unitários presentes na classe TruckLoadBusinessTest

5.4.4 Mecanismos para execução sequencial de testes do mesmo tipo

Foram criadas classes adicionais para executar todos os testes de um determinado tipo, o que permite poupar tempo em relação à execução individualizada de cada teste. Assim, para executar todos os testes unitários de um determinado tipo, apenas será necessário executar a classe apropriada que agrega todas as classes dos testes desse tipo. Por exemplo, caso se pretenda executar todos os "ValidatorTests", apenas será necessário executar a nova classe ("AllValidatorTests") criada para o efeito, tal como se pode observar na Figura 24. Sempre que se pretenda adicionar novas classes de testes, apenas terão de se atualizar as referências previamente definidas. Deste modo, assegura-se que são executados todos os testes de um determinado tipo, mesmo quando se inserirem novas classes de teste.

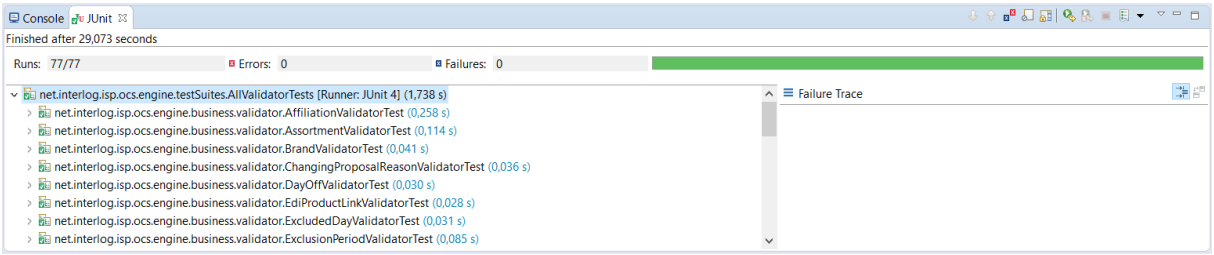


Figura 24 - Execução dos testes do tipo de validação de input

Também foi explorada a execução de testes através de linha de comandos. A execução de um comando Maven, apresentado na Figura 25, permitiu executar todos os testes de camada de acesso a dados.

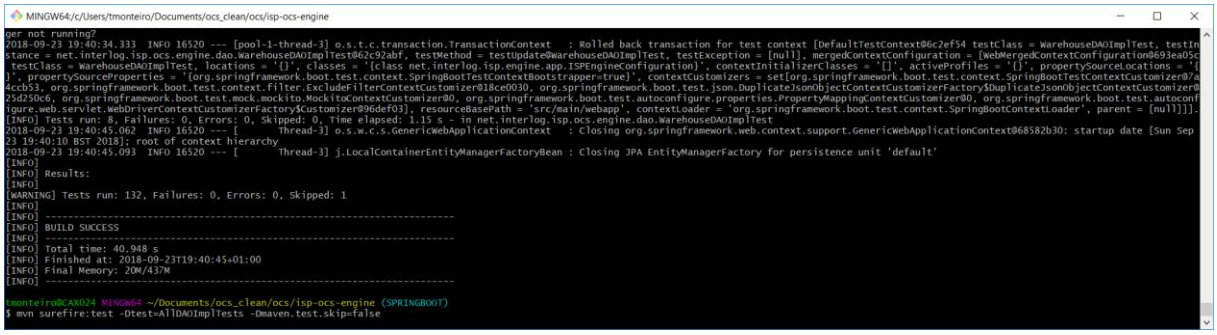


Figura 25 - Execução de testes unitários via linha de comandos

5.5 Definição de períodos para análise de dados

Quando se pretende efetuar uma análise, ou obter uma estatística, que incida sobre os dados do sistema, normalmente é definido um intervalo de tempo sobre o qual a análise incide.

Neste projeto foi necessário desenvolver a funcionalidade de criação de intervalos de tempo, ou períodos, para utilizar aquando da realização de análises de dados armazenados na base de dados. Os requisitos desta funcionalidade encontram-se descritos no Anexo 3.

Esta funcionalidade envolveu a criação de uma nova entidade, e a criação da respetiva UI no OCS, pelo que se utilizou o processo descrito na Secção 5.2.1. Foram disponibilizadas ao utilizador as seguintes opções (em termos da UI) para "dividir um ano completo":

- Por semana (Divisão de um ano em 52 ou 53 semanas);
- A cada 4 semanas (Divisão de um ano em períodos de 4 semanas);
- Por mês (Divisão de um ano nos respetivos meses civis).

É de salientar que no sistema não há períodos sobrepostos nem datas "esquecidas" ou em falta num intervalo de tempo. Se assim fosse, a produção das análises estaria em causa, produzindo resultados não necessariamente corretos. Assim, o início de um período tem de ser feito logo após haver o término do período anterior, para que não haja falhas. A título de exemplo, se para o ano de 2018 a divisão de períodos for feita mensalmente, então o primeiro período vai desde

1 janeiro até 31 de janeiro. O período seguinte de 1 fevereiro até 28 de fevereiro e assim por diante.

A escolha anual do período por parte do utilizador não tem de ser necessariamente a mesma durante o uso da aplicação. O sistema permite sempre a seleção entre as 3 opções disponíveis, armazenando as modificações na base de dados (tabela "period").

Também é permitido que o utilizador altere as datas dos períodos, com exceção da primeira e da última data, para garantir que as análises incidam num só ano. O sistema faz uma validação prévia para verificar a continuidade das datas, e, caso tal não se verifique, será apresentada uma mensagem de erro e não efetuará a atualização desses dados.

Para esta funcionalidade foram criados 3 tipos de teste unitários. Inicialmente foram criados os testes da classe *PeriodDAOImplTest* e *PeriodValidatorTest*. Após sua conclusão foi criada a classe *PeriodBusinessTest*, e foram também adicionados os seguintes testes unitários (Figura 26):

- Verificar que são criados 52 períodos semanais no ano 2017, sendo que a primeira data de começo do primeiro período é o dia 2 de janeiro e a última data corresponde ao dia 31 de dezembro. Verificar que são criados 13 períodos com duração de 4 semanas onde as datas iniciais e finais coincidem. Este procedimento de teste foi continuado até o ano 2022, onde as datas iniciais e finais coincidiram entre das verificações.

- Verificar que 29 períodos mensais são calculados/gerados entre as datas 01/09/2016 e 31/01/2019 (4+12+12+1)

- Verificar que são criados 15 períodos diários entre as datas 01/01/2018 e 15/01/2018

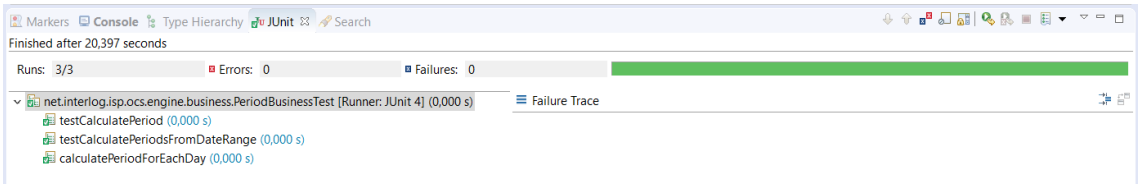


Figura 26 - Execução dos testes na classe PeriodBusinessTest

5.6 Análise da qualidade de serviço

Nos projetos do tipo VMI é necessário definir objetivos e selecionar algumas métricas a considerar. Estas métricas são conhecidas como *Key Performance Indicator*s (KPIs) e medem aspetos relacionados com o desempenho do negócio, e que afetam tanto fornecedores como clientes.

Com isto em mente, esta funcionalidade foi criada com objetivo de medir o critério/indicador *availability* ou disponibilidade (uma métrica de alta relevância neste projeto). O cálculo deste indicador (em percentagem) é dado pela razão entre as quantidades em falta e as quantidades efetivamente entregues, como mostra a seguinte expressão:

$$\text{tx} = \left(\left(1 - \frac{\text{totalOfMissingCases}}{\text{totalOfCases}}\right)\right) * 100$$

Os requisitos desta funcionalidade encontram-se descritos no Anexo 3.

No que toca a regras de negócio, esta análise é realizada através do histórico de encomendas no sistema. Esse histórico é guardado na tabela "bi_fact_order_proposal" da base de dados e a tabela é alterada sempre que uma nova encomenda é validada pelo utilizador (Figura 27).

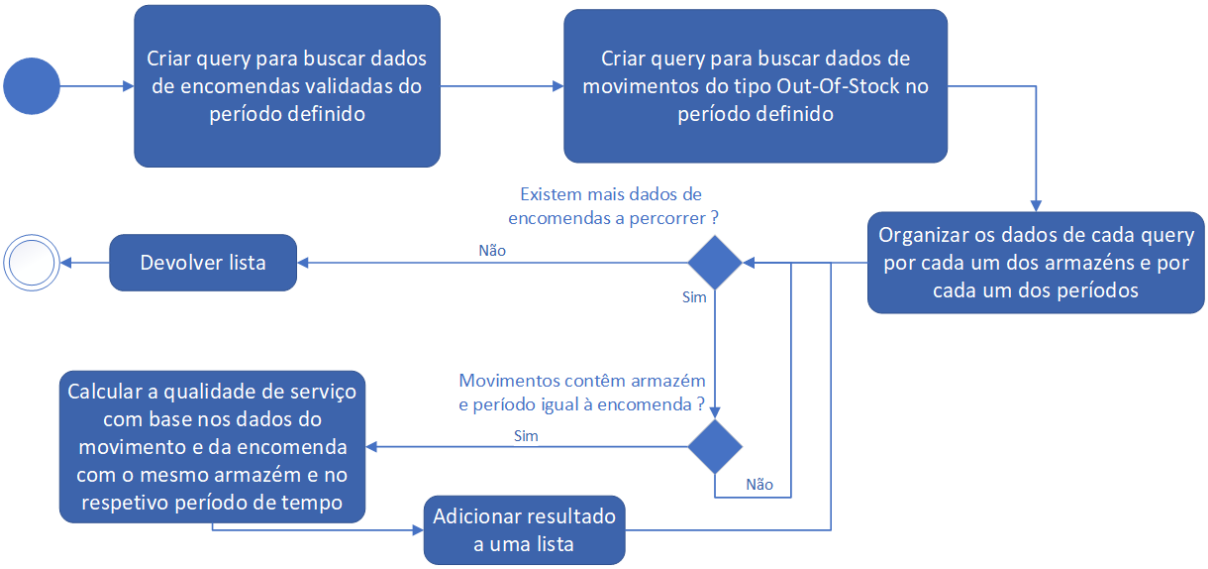


Figura 27 - Processo para calcular a qualidade de serviço

No desenvolvimento desta funcionalidade, a UI construída foi dividida em 3 áreas distintas. Uma área para definição dos critérios, outra para a área de resultados e finalmente uma área para albergar um gráfico de acordo com os resultados obtidos.

A área para definição dos critérios permite restringir as afiliações e os armazéns a considerar, e definir o período sobre o qual se pretende realizar a análise de qualidade de serviço. Este período por predefinição é mensal, podendo o utilizador definir um outro período.

A geração da análise é realizada de acordo com os critérios selecionados, e é apresentada na área de resultados em formato de tabela. O gráfico é elaborado com a mesma informação proveniente da área de resultados, e apresenta a evolução da métrica disponibilidade durante o intervalo de tempo escolhido.

No âmbito desta funcionalidade foi também atualizado um dos gráficos apresentado noutra vista do sistema, e que continha um conjunto de valores previamente estabelecidos. O propósito da atualização do gráfico foi o de apresentar a evolução da taxa de serviço, mas com a particularidade de agrupar os dados por afiliação/grupo (que contém um conjunto de armazéns), e considerando o que se passou em cada semana (as últimas 4). O gráfico a apresentar também está dependente da existência de um filtro que poderá ser aplicado por defeito ao carregar a vista. Este filtro tem como objetivo selecionar a lista de armazéns que é apresentada ao utilizador, e o sistema apenas deverá incluir os itens filtrados na construção do gráfico. Na ausência do filtro, o sistema considerará todos os armazéns da base de dados

Posteriormente, em relação a esta funcionalidade, foi pedido para adicionar um rácio entre as quantidades em falta cuja responsabilidade foi atribuída somente ao fornecedor e as quantidades entregues, dada pela expressão:

$$\mathrm{tx} = \left(\left(1 - \frac{\mathrm{totalOfProviderMissingCases}}{\mathrm{totalOfCases}}\right)\right) * 100$$

Para além dos testes de usabilidade, foi feito um teste de lógica de negócio no contexto desta funcionalidade. O teste pretendia estimar os valores do serviço tendo em conta os dados de movimentos, armazéns e períodos que foram definidos ao nível do *dataset*.

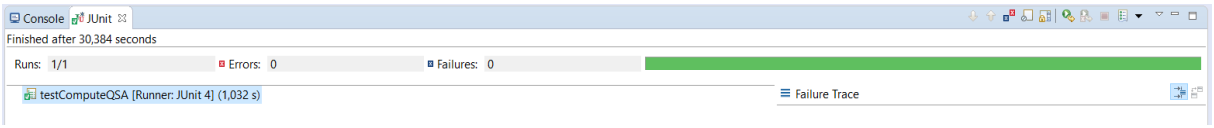


Figura 28 - Teste de lógica de negócio realizado na tarefa de Análise da qualidade de serviço

5.7 Mapeamento de *links*

Em processos logísticos, é recomendado utilizar alguns *standards* num sistema VMI dos quais se destacam os *standards* GS1. Estes standards foram criados com o intuito de criar regras únicas capazes de identificar, capturar e partilhar informação acerca de produtos, localizações e ativos, tornando a informação acessível, correta e percetível. [4]

Em sistemas VMI é usual o envio de informação digital entre parceiros de negócio. O *Electronic Data Interchange* (EDI) é uma forma de trocar documentos em formato eletrónico entre computadores, e é um dos requisitos para qualquer projeto VMI.

No OCS, existe um código EAN secundário (limitado a 20 caracteres) semelhante ao código EAN principal de um produto. A função deste código EAN secundário é o de auxiliar na pesquisa do código EAN primário que por sua vez permite saber a que armazém está associado o código EAN secundário. Este código EAN secundário é utilizado quando se converte movimentos EDI e quando há a confirmação de uma encomenda.

Os requisitos desta funcionalidade encontram-se descritos no Anexo 4. Na sua implementação foram criadas as páginas para o utilizador poder usufruir das funcionalidades básicas, tendo sido criada uma vista para editar “links” e outra vista com a listagem de todos os “links” prontos a ser editados ou eliminados (com uma opção para criar um novo “link”). Ainda neste contexto, a vista de edição do “link” permite que o utilizador escolha um dos armazéns disponíveis no sistema e insira/atualize o código secundário, gravando as alterações na tabela “location_link”, que teve de ser adicionada à base de dados.

Foram realizados testes unitários aos mecanismos de persistência CRUD e foram realizados testes unitários ao nível da validação das regras de negócio (Figura 29 e Figura 30).

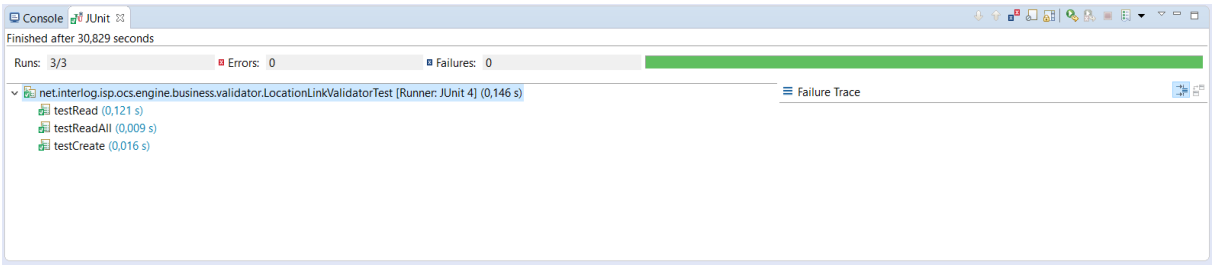


Figura 29 - Testes de validação de input para a tarefa Mapeamento de links

Figura 30 - Testes à camada de acesso a dados para a tarefa Mapeamento de links

5.8 Análise de *out-of-stock*

Numa *supply chain* um dos maiores problemas para os clientes retalhistas é o de não possuir os produtos disponíveis para serem comercializados. Estas situações também são conhecidas como situações de *out-of-stock*. Podem existir várias razões ou causas para esse problema, pelo que foi desenvolvida uma funcionalidade para efetuar uma análise com o intuito de tentar descobrir essas razões. Os requisitos desta funcionalidade encontram-se descritos no Anexo 3.

No desenvolvimento do ecrã desta análise foram criadas as seguintes áreas: área de seleção de período, área de critérios, área de opções de análise e área de seleção de produtos. É nesse ecrã que o utilizador define os parâmetros da análise.

Na área de seleção de período existem 3 opções de seleção disponíveis: por data, por período, e por mês. Após selecionar uma destas opções tem de ser definido o intervalo da pesquisa, pelo que há necessidade de introduzir um valor inicial e um valor final. Estes "limites" são ambos inclusivos, pelo que, por exemplo ao selecionar-se a opção "por mês" e ao definir as opções "2018 - janeiro" e "2018 - abril", o sistema tem em consideração os meses de janeiro, fevereiro, março e abril do ano de 2018.

Na área de critérios pode ser selecionado o armazém sobre o qual a análise incide, considerando o sistema por defeito todos os armazéns existentes. O mesmo se aplica para às famílias de

produtos existentes, e para o seu tipo (*standard* ou promocional). Por defeito a opção inicial selecionada considera toda a informação existente.

A opção de análise "por causa" agrupa o resultado apenas pela causa (de uma situação *out-of-stock*) ao passo que, a opção "por causa/produtos" agrupa o resultado pela causa e pelos produtos selecionados. Poderá também aplicar-se uma terceira opção de agrupamento dos dados, neste caso pelo período de seleção escolhido. As unidades de consumo disponíveis para seleção são "UVC" e "Caixa".

Por defeito, todos os produtos do sistema considerados, podendo também ser selecionados os produtos específicos sobre os quais a análise vai ocorrer.

Ao executar a análise com os parâmetros definidos, o sistema vai fazer os cálculos necessários para apurar as razões pelas quais os produtos estão em falta. O resultado é apresentado numa nova vista, onde aparece um resumo com as opções escolhidas, uma área com a lista de resultados encontrados e uma área com um gráfico circular com base nos dados da lista anterior. O processo encontra-se descrito na Figura 31.

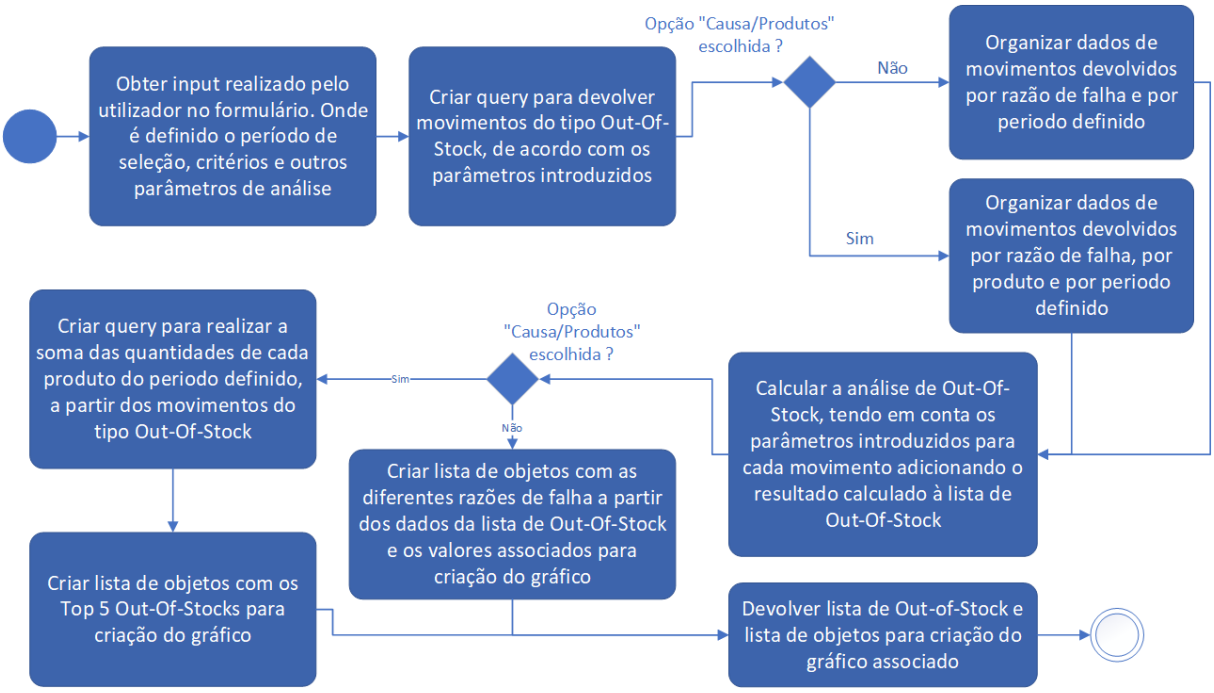


Figura 31 - Fluxograma para criação da análise out-of-stock

Nesta funcionalidade foram feitos dois testes de lógica de negócio, um testar o código com a opção "Por Causa" e outro com a opção "Por Causa/Produto". Para cada um dos testes, foram

verificadas as diferentes opções de agrupamento dos períodos, e o objetivo passou pela verificação das quantidades e percentagens resultantes do algoritmo aplicado. Também foi testado o número de linhas esperadas.

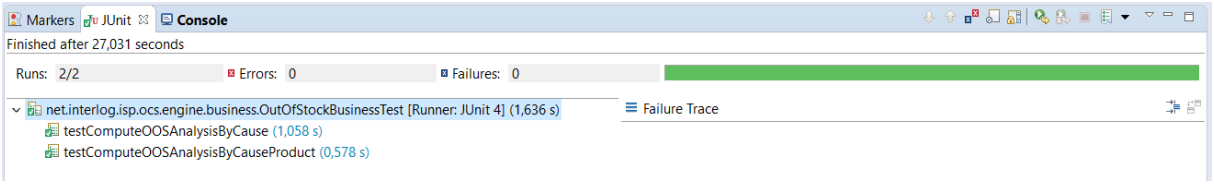


Figura 32 - Testes à camada de lógica de negócio da funcionalidade de Análise de out-of-stock

5.9 Análise de cobertura média

Esta análise deriva da quantidade de saídas (*outputs*) e dos *stocks* contidos na tabela "bi_fact_assortment" num determinado período permitindo realizar a análise da evolução da cobertura de *stock* sobre alguns parâmetros.

No desenvolvimento do ecrã desta análise foram criadas as seguintes áreas: área de seleção de período, área de critérios (por defeito considera todos os dados) e área de opções de análise (onde se destaca o número de dias a considerar). Ao executar a análise o sistema redireciona para a página com a apresentação dos resultados de acordo com os parâmetros inseridos.

No algoritmo criado para esta análise, o sistema vai por cada dia do período selecionado calcular a cobertura de acordo com o número de dias introduzido. A fórmula é dada pela divisão entre a soma dos *stocks* no número de dias introduzido, e a soma das quantidades saídas. De notar que nos cálculos apenas são considerados os dias em que o armazém aceita movimentos (Figura 33).

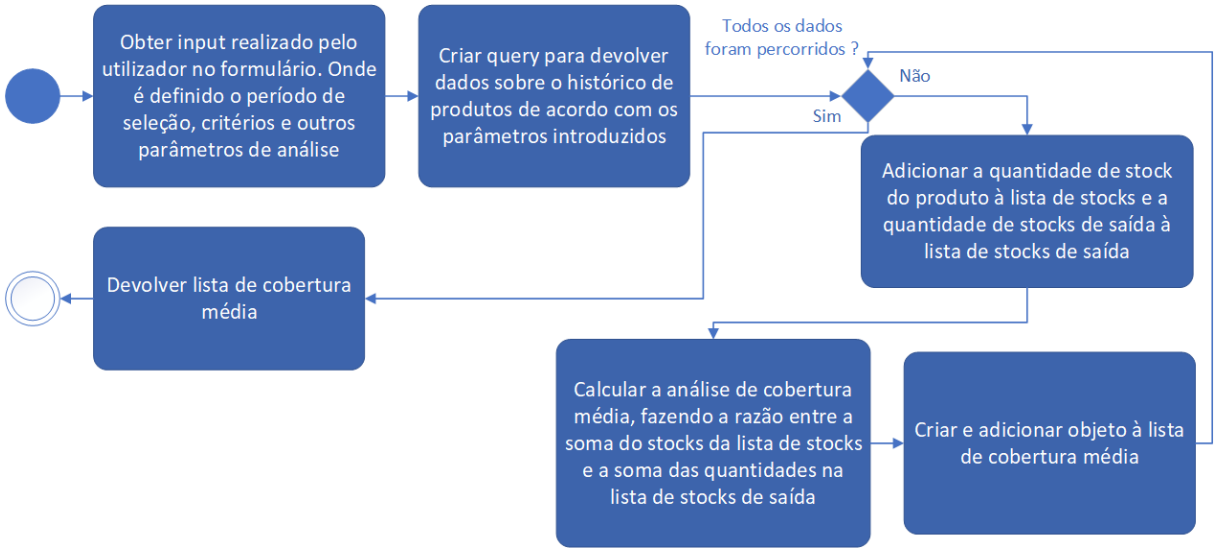


Figura 33 - Análise de cobertura média

5.10 Adicionar produtos a armazéns

A lista de produtos de um determinado armazém pode ser alterada, ou seja, não está estritamente fechada a hipótese de incluir ou excluir produtos. Com isto em mente, procedeu-se a implementação desta funcionalidade que também fazia parte da versão OCS mais antiga.

A partir da edição de um produto (de um armazém), é possível aceder à página para adicionar produtos e alterar as seguintes opções relativamente aos produtos:

- Unidade de reabastecimento: Paletes, *Layers* (Camadas) e Unidades (opção por defeito). Esta opção é obrigatória;
- Código do Cliente (campo opcional);
- Data de Início de Ativação (campo obrigatório) e de Fim de Ativação (campo opcional).

Ao gravar as alterações o sistema faz as seguintes validações no módulo isp-ocs-engine:

- A data de fim não é obrigatória, mas no caso de ser inserida tem de ser igual ou superior à data de início;
- O campo código de cliente no caso de ser preenchido tem de ter no máximo 20 caracteres.

No momento da escolha do(s) armazém(éns) que vão integrar o novo produto, é realizada para cada um dos armazéns uma verificação para determinar se o produto já existe. Em caso afirmativo, o sistema apresenta uma mensagem de erro a indicar que o produto já existe num dos armazéns, não sendo armazenada nenhuma informação na base de dados. Caso os armazéns selecionados não tenham o produto na sua lista de produtos, o sistema atualiza cada uma das listas e armazena essa informação na tabela "assortment" da base de dados.

5.11 Análise da unidade de logística ótima

No que diz respeito a processos logísticos dos armazéns, é importante monitorizar a informação referente aos produtos comercializados, às suas unidades, locais de carga e descarga, horas de partida e chegada, etc.

Os reabastecimentos dos produtos de um armazém podem não ter as quantidades corretas ou adequadas, caso não seja efetuada uma análise prévia para determinar a unidade logística correta. O objetivo para esta análise é o de reduzir custos de pick-up para o fornecedor e reduzir os custos de manutenção para o distribuidor. Para efetuar essa análise o sistema calcula qual das 3 unidades de reabastecimento (paletes, camadas ou caixas) melhor se adequa aos produtos, e as respetivas quantidades (Figura 34). Os requisitos desta funcionalidade encontram-se descritos no Anexo 3.

Como tem vindo a ser habitual, existe uma área para a definição de critérios para restrição. Por defeito o sistema considera todas as afiliações, armazéns e famílias dos produtos. Existe uma opção para especificar um período através da inserção da data de início e da data de fim (ambas os campos são obrigatórios) e finalmente o modo de comparação que é obrigatório (Min-Max ou Diferença entre Min-Max) e que tem influência na escolha da unidade de logística ótima. O sistema apresenta os resultados numa nova vista, com os parâmetros introduzidos e a tabela de resultados.

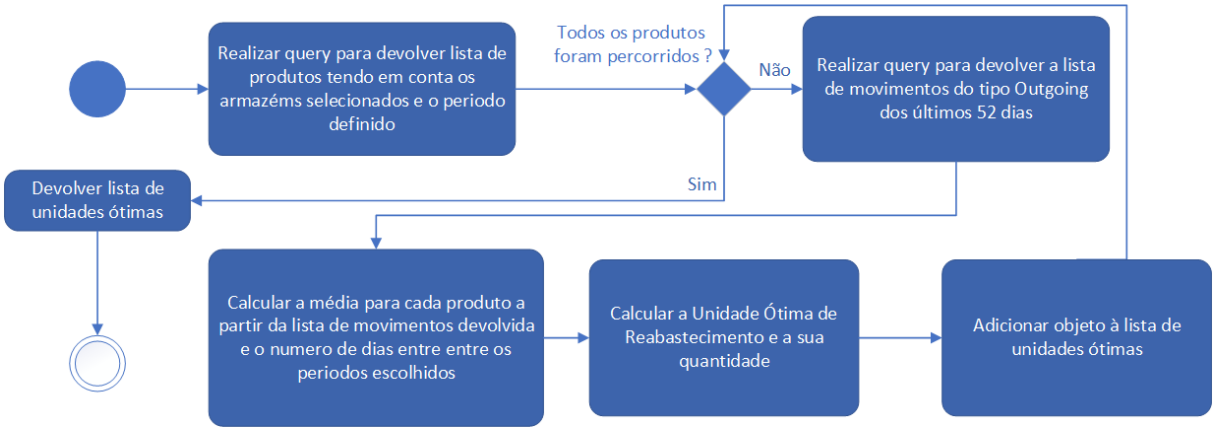


Figura 34 - Fluxograma para criação da análise da unidade ótima de logística

5.12 Cálculo Min Max

Para qualquer armazém é possível otimizar as configurações em dias de *stock* mínimo e máximo para os seus produtos. Neste contexto, para realizar o cálculo Min Max, o utilizador necessita de introduzir para cada gama de produtos do armazém os parâmetros ou campos obrigatórios apresentados na Tabela 2.

Tabela 2 – Campos obrigatórios para o cálculo Min Max

Campo	Tipo de Campo	Valor por omissão	Valor Mínimo	Valor Máximo
Histórico de consumo (em semanas)	Numérico	4	0	99
Tempo de entrega (em semanas)	Numérico	Tempo de espera para o armazém	0	99
Taxa de qualidade de serviço (em percentagem)	Numérico (A selecionar)	Nenhum	98.5; 99.0; 99.5; 99.75; 99.99	
Armazenamento / Descarga Mínima (em dias)	Numérico	Nenhum	0	99
Armazenamento / Descarga Máxima (em dias)	Numérico	Nenhum	0	99

Para além desses parâmetros, existe ainda o seguinte parâmetro obrigatório comum: tipo de cálculo. Este parâmetro define a frequência de atualização automática das configurações. Pode assumir um dos seguintes valores: a cada semana, a cada 2 semanas, a cada 3 semanas ou a cada 4 semanas.

Os requisitos desta funcionalidade encontram-se descritos no Anexo 4. No desenvolvimento da funcionalidade foi necessário construir a UI e armazenar os parâmetros introduzidos na base de dados, na tabela “warehouse_goal_optimization” que foi criada para esse efeito. Para além das

validações anteriormente descritas, houve também uma validação adicional considerada, e que obriga o campo Armazenamento / Descarga Máxima a ter um valor não inferior ao campo Armazenamento / Descarga Mínima.

Foi também desenvolvido um simulador (temporário) para calcular os valores mínimos e máximos de *stock* em dias para cada produto de um armazém, de acordo com o *input* do utilizador e sem guardar os dados na base de dados. Concluída a simulação, o utilizador é redirecionado para uma nova UI onde pode selecionar os produtos para os quais pretende realmente atualizar os valores do *stock* mínimo e máximo (calculados através do simulador utilizado), e o sistema grava as alterações na tabela "assortment" da base de dados.

Para além dos testes de usabilidade efetuados, foram realizados testes à camada de acesso a dados como se encontra na Figura 35.

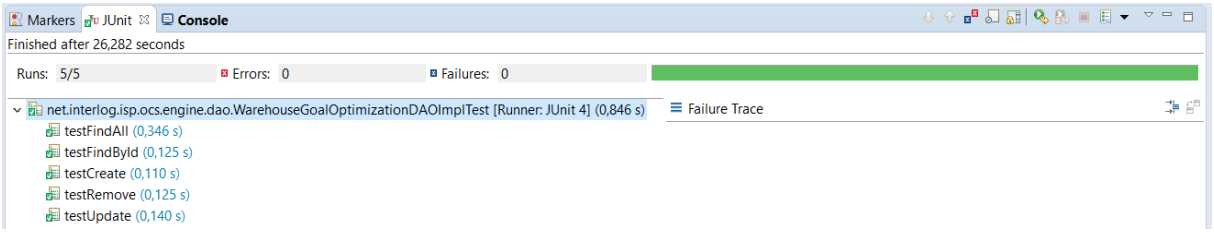


Figura 35 - Testes realizado à camada de acesso a dados (funcionalidade Cálculo Min Max)

5.13 Geração de ficheiros PDF

Uma das funcionalidades existentes na versão inicial do OCS é a de impressão de documentos. Na nova versão do OCS esta funcionalidade manteve-se, tendo adicionalmente sido adicionada a geração/criação de ficheiros PDF, que poderão ser visualizados ou impressos a partir de um *browser*.

Antes de se desenvolver a funcionalidade, foi efetuada uma pesquisa de potenciais ferramentas, ou bibliotecas, que pudessem ser utilizadas no projeto. As ferramentas analisadas encontram-se descritas na Tabela 3.

Tabela 3 – Ferramentas pesquisadas para geração de ficheiros PDF

Ferramenta	Descrição e características
iText [44]	Ferramenta popular que permite a criação de PDFs com base na manipulação de objetos (que implementem a interface Elements). **Vantagens**: Sintaxe simples, com vários exemplos documentados. Permite inserir texto, imagens e tabelas entre outros. **Desvantagens**: As versões iText 5.x.x e iText 7.x.x usam uma licença AGPL, o que significa que é necessário respeitar as regras impostas por esta licença para se utilizar estas versões sem custos. Caso contrário, é necessário adquirir uma licença comercial.
Apache PDFBox [45]	Ferramenta para criar, manipular PDFs baseada na manipulação de *streams*. **Vantagens**: Inserção de Texto, Inserção de Imagens **Desvantagens**: Difícil construção de certos elementos (exemplo: tabelas)
Jaspersoft Studio [46]	Ferramenta de design baseada no Eclipse para criação de layouts sofisticados. **Vantagens**: Integração com várias fontes de dados (JDBC, Hibernate, XML, CSV, …) e posterior criação de vários tipos de documentos (PDF, CSV, Word, XML). **Desvantagens**: Cumprimento de requisitos mínimos do sistema, curva de aprendizagem
OpenPDF [47]	Biblioteca baseada em Java que foi desenvolvida sobre uma versão do iText 4 para criação e edição de PDFs. **Vantagens**: Biblioteca com licença LGPL **Desvantagens**: Pouca documentação e exemplos de utilização

A partir da versão 5 do iText, a licença usada na biblioteca é AGPL, o que levaria a aquisição de uma licença comercial do iText ou tornar público o código fonte. Devido a estas limitações excluiu-se o uso desta biblioteca.

O JasperSoft Studio é uma solução que está preparada para utilizar vários tipos de documentos e produzir *reports*. Este simples facto, exige uma elevada curva de aprendizagem para trabalhar

com a ferramenta e realizar configurações, e tendo em conta que se queria gerar ficheiros PDF era uma solução esmagadora para o que se pretendia fazer, o que levou à sua exclusão.

O Apache PDFBox fornece as características básicas para a construção de um PDF, embora não possua o suporte para a criação de tabelas. Atendendo a que os *reports* incluem tabelas, não pareceu ser uma boa solução a utilizar.

O OpenPDF, que utiliza o iText4, pareceu ser a melhor alternativa, uma vez que as condições da licença LGPL estavam reunidas e pela forma simplificada de programar. A principal limitação residia na falta de documentação disponível, visto que a API do iText 4 deixou de estar disponível.

Foi efetuado um pequeno teste para verificar se seria possível utilizar o OpenPDF para criar/gerar um PDF. Ao realizar o teste, baseado em alguns exemplos do iText 5 e iText 7, verificou-se que muitos dos métodos disponíveis já tinham sofrido alterações. Ainda assim, foi possível resolver esse problema, e concluir com sucesso o teste de criação de um documento PDF.

Na sequência da apresentação à equipa de desenvolvimento da pesquisa e dos testes efetuados, por proposta de um membro, decidiu-se explorar a utilização das bibliotecas Flying Saucer [48] e o Thymeleaf [49] para gerar PDFs a partir de um *template*. O Thymeleaf é uma ferramenta Java, executada no lado do servidor, para criação de *templates* em HTML através de um conjunto de dados. Utiliza uma licença Apache 2.0. O Flying Saucer é uma ferramenta Java capaz de transformar XML ou XHTML em PDFs, e que utiliza CSS para a formatação do seu conteúdo. Na construção dos PDFs foram utilizados dois dos artefactos desta ferramenta: o flying-saucer-core e o flying-saucer-pdf-itext5. A licença encontra-se sobre a versão 2.1 do GNU LGPL. A Figura 36 esquematiza o processo de geração de PDFs utilizando simultaneamente as duas bibliotecas [50] [51].

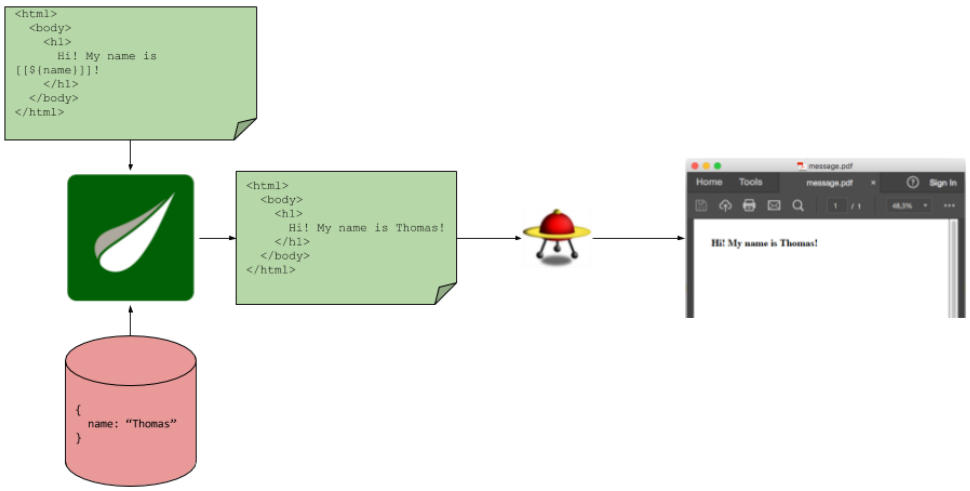


Figura 36 - Pipeline para renderizar PDFs com o Thymeleaf e Flying Saucer, adaptado de [50]

Estas ferramentas foram facilmente integradas no projeto ao introduzir a respetiva dependência no ficheiro pom.xml do módulo isp-ocs-engine.

Nesta funcionalidade pretendia-se que o PDF gerado não fosse imediatamente transferido para o dispositivo, mas que fosse apresentado num novo separador do *browser*. O utilizador poderia armazenar no computador e/ou imprimir. Para cumprir esse requisito foi necessário recorrer a uma biblioteca adicional, que é uma extensão de um componente Vaadin. A sua integração foi feita através de uma dependência Maven no ficheiro pom.xml do projeto isp-ocs-ui [52].

5.14 Gestão da visibilidade dos componentes Vaadin

Esta funcionalidade permite alterar a visibilidade de determinados componentes Vaadin de uma página, tais como *TextFields*, *ComboBoxs* e *DateFields*. Foram considerados os seguintes três estados principais de visibilidade para os componentes: visível (por defeito), escondido e desativado.

Após ter sido implementada a alteração do estado dos componentes, foi desenvolvido um mecanismo para definir o estado de um componente de acordo com as permissões do utilizador autenticado no sistema. Para tal, foi criada, e programada, uma anotação (e respetiva programação), implementada no módulo isp-ocs-ui, e os componentes selecionados teriam de se especificar na altura da sua declaração. A anotação apenas recebe como parâmetros de entrada o nome da permissão a verificar, e a visibilidade a aplicar no caso a permissão especificada não exista. Aos componentes da UI que não estejam anotados, não são sujeitos à aplicação do algoritmo da Figura 37.

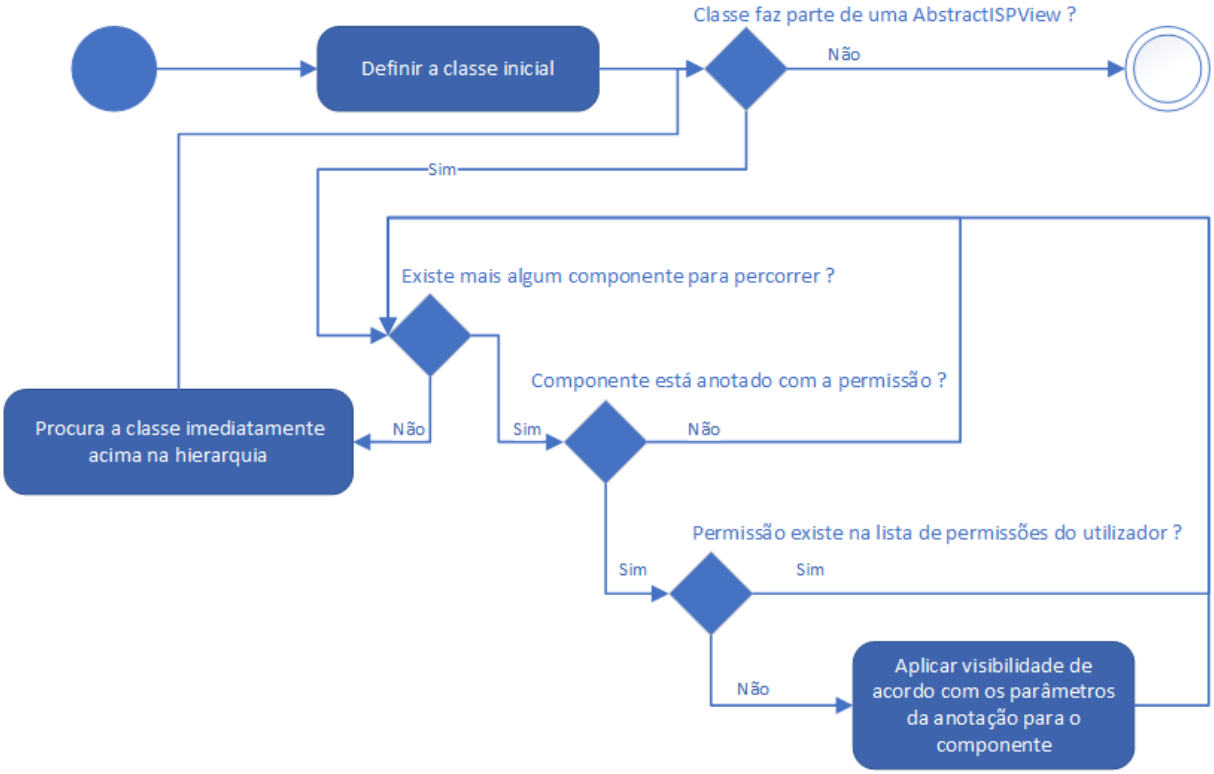


Figura 37 - Aplicação da visibilidade dos componentes numa página

Quando a implementação final do mecanismo foi concluída, após a realização de vários testes de usabilidade aos componentes, o mecanismo foi estendido a quase todas as páginas da aplicação onde se criavam ou editavam registos.

5.15 Apresentação e otimização dos *dashboards* da página inicial

Com esta funcionalidade pretendia-se efetuar a apresentação dinâmica de gráficos na página inicial, e que aparecem quando o utilizador se autêntica. Inicialmente a página inicial da aplicação apresentava 3 gráficos com valores simulados e uma lista de armazéns.

Esta funcionalidade em particular foi realizada de maneira faseada. Isto porque a construção de cada um dos gráficos foi pedida e realizada em alturas e contextos diferentes.

O primeiro gráfico a ser construído de maneira dinâmica foi o gráfico que diz respeito à evolução ao longo das últimas 4 semanas do nível da qualidade de serviço (em percentagem), por cada afiliação (Figura 38 - Processo para calcular o gráfico da Qualidade de Serviço). Para realizar este cálculo foi necessário realizar *queries* de maneira a conseguir obter os dados e realizar algumas operações sobre os mesmos. Houve a necessidade de criar uma nova classe

que pudesse ser acessível no módulo isp-ocs-ui (e que contivesse a informação necessária para a construção dos gráficos)

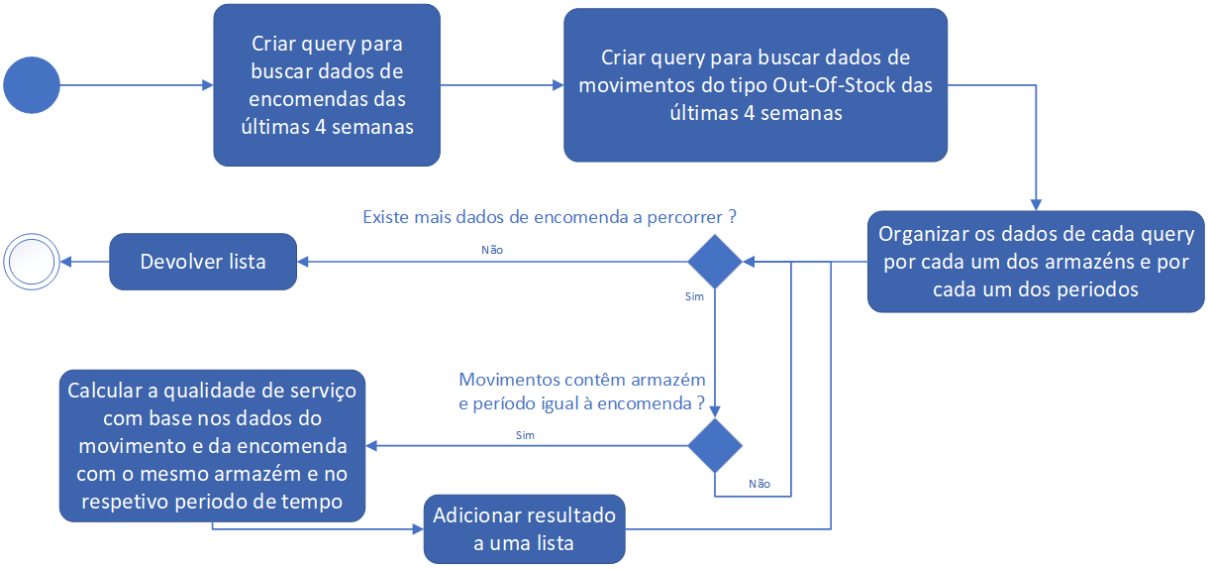


Figura 38 - Processo para calcular o gráfico da Qualidade de Serviço

Posteriormente, no contexto de outra funcionalidade, foi efetuada a atualização do gráfico de "Análise de *Out-of-Stock*". O objetivo deste gráfico circular era o de mostrar os "top 5" produtos com maiores quantidades em falta (considerando o total das 4 semanas anteriores). Para tal, foram efetuadas algumas *queries* para obter os dados necessários (Figura 39).

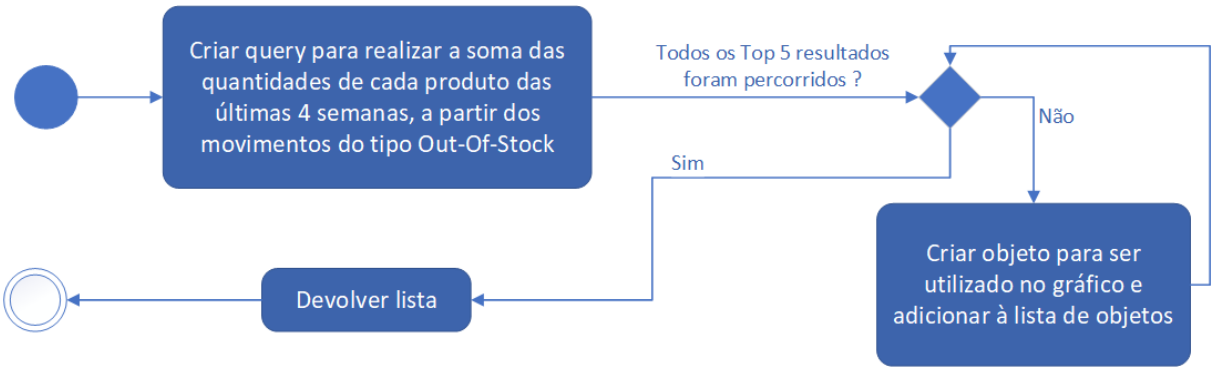


Figura 39 - Processo para calcular o gráfico de Análise de Out-Of-Stock

Finalmente, foi também atualizado o gráfico de Cobertura de *Stock*, tendo com base o último dia das 4 semanas anteriores (Figura 40).

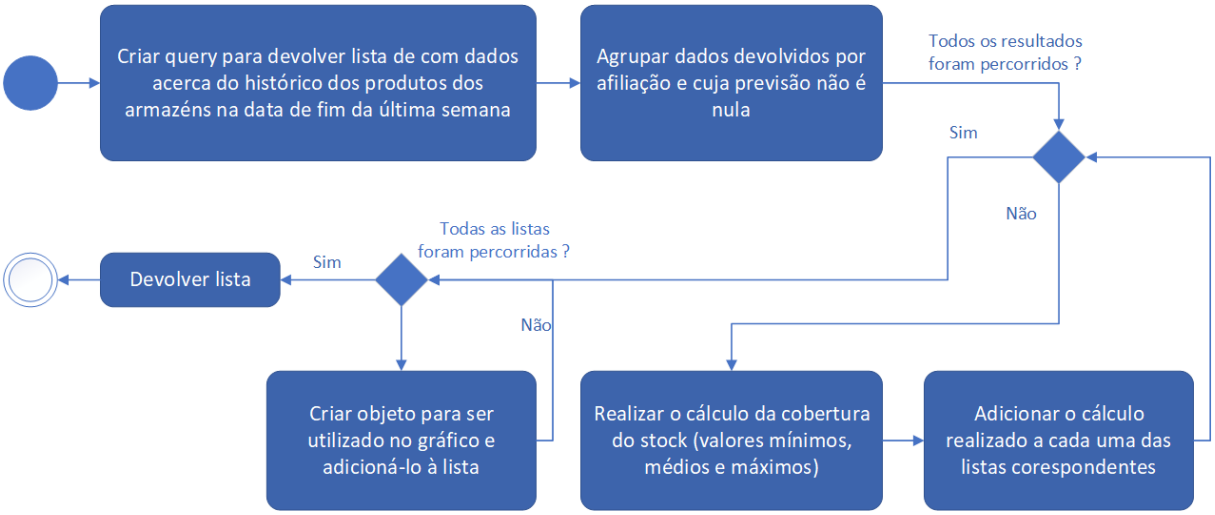


Figura 40 - Processo para calcular o gráfico de Análise de Cobertura de Stock

A página inicial passou a apresentar um *dashboard* com 3 gráficos dinâmicos distintos. Para cada um desses gráficos têm de ser efetuadas múltiplas *queries* à base de dados, bem como fazer o respetivo *rendering*.

À medida que se foi substituindo o conteúdo estático dos gráficos por *queries* de maneira dinâmica, o tempo de espera para o carregamento da página foi aumentando. Inicialmente este aumento não foi muito significativo, devido ao reduzido volume de registos na base de dados. No entanto, esse tempo de espera aumentou consideravelmente quando se passou a utilizar uma base de dados com um maior número de registos (na ordem dos milhões em algumas tabelas).

Para resolver esse problema, a informação que os gráficos necessitavam foi armazenada em novas tabelas da base de dados (home_fact_data e home_product_fact_data). Assim, passaram a ser executadas mais rapidamente as *queries* que obtêm a informação para os gráficos. Para atualizar estas tabelas é necessária a execução de um comando, que por sua vez faz uma chamada a uma API REST. De seguida, o módulo isp-ocs-engine processa esse pedido de atualização das tabelas com os dados referentes à última semana.

Capítulo 6

Conclusão

Fazendo um balanço geral de todas as atividades realizadas durante o estágio, pode-se dizer que foi uma experiência muito enriquecedora quer a nível profissional quer a nível pessoal, e que também permitiu ganhar novas competências técnicas, sobretudo ao nível das cadeias de abastecimento e do ambiente de desenvolvimento Java.

O objetivo principal do projeto proposto pela entidade de acolhimento era o de implementar novas funcionalidades num *software* VMI, denominado OCS, em que se pretende gerir um processo de *cross docking*, onde o fornecedor é o responsável por gerir os armazéns dos seus clientes (retalhistas), de acordo com valores máximos e mínimos definidos para os *stocks*.

O planeamento e as tarefas inicialmente propostas tiveram algumas alterações durante o estágio, sobretudo devido às prioridades que foram sendo definidas pela entidade acolhedora. Ao longo do desenvolvimento do *software* OCS, o gestor de projeto efetuou a gestão e a distribuição das tarefas aos elementos da equipa de desenvolvimento de acordo com as suas disponibilidades e com as prioridades que foram sendo definidas.

Os objetivos principais definidos para o estágio foram integralmente cumpridos uma vez que foram desenvolvidas todas as funcionalidades atribuídas (Figura 19) de maneira funcional. Todas essas funcionalidades encontram-se atualmente em produção. Foram também resolvidos problemas, para os quais não foi atribuída uma tarefa específica, que permitiram garantir o correto desempenho do sistema nos contextos adequados.

Durante o estágio foram sentidas algumas dificuldades para compreender integralmente algumas das funcionalidades solicitadas, tendo obtido os esclarecimentos necessários através de reuniões e de emails. Também surgiram dificuldades a nível técnico, sobretudo no início,

pois nunca tinha trabalhado com algumas das tecnologias presentes no projeto, nomeadamente com o Spring e o Vaadin, e porque não se tinha o conhecimento completo da organização do projeto.

Durante o desenvolvimento das funcionalidades foram realizados testes unitários e de usabilidade, de modo a assegurar a qualidade do *software* desenvolvido e para a detetar de possíveis erros de regressão. Atendendo a que se pretendia colocar o *software* em produção, foi tida uma particular atenção com os testes de *software*, por forma a evitar que os utilizadores se deparem com erros que possam causar bloqueios de processos e a incapacidade de executar as tarefas pretendidas.

Após um período de testes em alguns clientes prioritários, uma nova versão do *software* foi colocada em produção no final do mês de maio. Essa versão já incluía todas as funcionalidades desenvolvidas ao longo do estágio.

A equipa de desenvolvimento do *software* OCS, que continuou a pertencer após a conclusão do estágio, para além de efetuar a manutenção corretiva do programa continuará a efetuar o desenvolvimento de novas funcionalidades. Estão previstas a criação e gestão partilhada dos camiões entre diferentes *suppliers*, a criação do módulo de promoções e a respetiva integração de ficheiros no sistema, o suportar novos perfis de utilizador, e novas opções de análise ou *reporting*. Também se pretende explorar a realização de testes automatizados de UI, possivelmente com a utilização do Selenium [53] no projeto, e ir efetuando a migração dos projetos para as versões mais recentes das tecnologias e *frameworks* utilizadas.

Bibliografia

[1] Z. Lotfi, M. Mukhtar, S. Sahran, and A. T. Zadeh, “Information Sharing in Supply Chain Management,” *Procedia Technol.*, vol. 11, pp. 298–304, Jan. 2013.

[2] D. Prajogo and J. Olhager, “Supply chain integration and performance: The effects of long-term relationships, information technology and sharing, and logistics integration,” *Int. J. Prod. Econ.*, vol. 135, no. 1, pp. 514–522, Jan. 2012.

[3] M. J. T. Claassen, A. J. van Weele, and E. M. van Raaij, “Performance outcomes and success factors of vendor managed inventory (VMI),” *Supply Chain Manag. An Int. J.*, vol. 13, no. 6, pp. 406–414, Sep. 2008.

[4] H. Hammer and C. Bernasconi, “Best Practice in Implementing VMI A recommendation by ECR Europe,” 2016. [Online]. Available: https://www.gs1.ch/docs/default-source/prozesse-dokus/ecr-community---best-practice-in-implementing-vmi.pdf?sfvrsn=2. [Accessed: 02-Oct-2017].

[5] “Global Locations - Interlog Services.” [Online]. Available: http://www.interloggroup.com/en/our-locations/. [Accessed: 10-Mar-2018].

[6] “Home - Interlog Group.” [Online]. Available: http://www.interloggroup.com/en/. [Accessed: 10-Mar-2018].

[7] “Freight Spend Management - Interlog Services.” [Online]. Available: http://www.interloggroup.com/en/our-services/interlog-services/. [Accessed: 10-Mar-2018].

[8] “Supply Chain Soutions - Interlog Solutions.” [Online]. Available: http://www.interloggroup.com/en/our-services/interlog-solutions/. [Accessed: 10-Mar-2018].

[9] C. A. Ullrich, “Introduction to Supply Chain Management,” in *Issues in Supply Chain Scheduling and Contracting*, Wiesbaden: Springer Fachmedien Wiesbaden, 2014, pp.

5–15.

[10] L. M. Ellram and M. C. Cooper, “Supply Chain Management: It’s All About the Journey, Not the Destination,” *J. Supply Chain Manag.*, vol. 50, no. 1, pp. 8–20, Jan. 2014.

[11] J. T. Mentzer *et al.*, “DEFINING SUPPLY CHAIN MANAGEMENT,” *J. Bus. Logist.*, vol. 22, no. 2, pp. 1–25, 2001.

[12] U. M. Apte and S. Viswanathan, “Effective Cross Docking for Improving Distribution Efficiencies,” *Int. J. Logist. Res. Appl.*, vol. 3, no. 3, pp. 291–302, Nov. 2000.

[13] J.-P. Rodrigue, C. Comtois, and B. Slack, *The geography of transport systems*. .

[14] “INTERLOG acquires OCS software and consolidates its VMI expertise.” [Online]. Available: http://www.interloggroup.com/en/interlog-acquires-ocs-software-vmi-expertise/. [Accessed: 17-Mar-2018].

[15] D. Kayal, *Pro Java EE Spring Patterns: Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework*. 2008.

[16] E. Jendrock, R. Cervera-Navarro, I. Evans, and K. Haase, *The java ee 7 tutorial*. 2014.

[17] “Spring Framework.” [Online]. Available: http://spring.io/projects/spring-framework. [Accessed: 16-Mar-2018].

[18] R. Johnson, J. Hoeller, K. Donald, C. Sampaleanu, and ..., “Spring Framework Reference Documentation - spring-framework-reference.pdf.” [Online]. Available: https://docs.spring.io/spring/docs/4.3.9.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf.

[19] “Spring Boot.” [Online]. Available: http://spring.io/projects/spring-boot. [Accessed: 16-Mar-2018].

[20] P. Webb, D. Syer, J. Long, S. Nicoll, and ..., “Spring Boot Reference Guide - spring-boot-reference.pdf.” [Online]. Available: https://docs.spring.io/spring-boot/docs/1.5.4.RELEASE/reference/pdf/spring-boot-reference.pdf.

[21] “Goals and Philosophy - Vaadin Framework 8 | Vaadin 8 Docs | Vaadin.” [Online]. Available: https://vaadin.com/docs/v8/framework/introduction/intro-goals.html. [Accessed: 08-Mar-2018].

[22] “Overview - Vaadin Framework 8 | Vaadin 8 Docs | Vaadin.” [Online]. Available: https://vaadin.com/docs/v8/framework/introduction/intro-overview.html. [Accessed: 08-Mar-2018].

[23] “Overview - Vaadin Framework 8 | Vaadin 8 Docs | Vaadin.” [Online]. Available: https://vaadin.com/docs/v8/framework/architecture/architecture-overview.html. [Accessed: 08-Mar-2018].

[24] “Overview - Vaadin Framework 8 | Vaadin 8 Docs | Vaadin.” [Online]. Available: https://vaadin.com/docs/v8/framework/application/application-overview.html. [Accessed: 08-Mar-2018].

[25] “Maven – Welcome to Apache Maven.” [Online]. Available: https://maven.apache.org/. [Accessed: 06-Jul-2018].

[26] “Maven – Introduction.” [Online]. Available: https://maven.apache.org/what-is-maven.html. [Accessed: 06-Jul-2018].

[27] “Jersey.” [Online]. Available: https://jersey.github.io/. [Accessed: 20-May-2018].

[28] M. Rouse, “What is open API (public API)? - Definition from WhatIs.com.” [Online]. Available: https://searchmicroservices.techtarget.com/definition/open-API. [Accessed: 10-Sep-2018].

[29] M. Rouse, “What is REST (REpresentational State Transfer)? - Definition from WhatIs.com,” 2018. [Online]. Available: https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer. [Accessed: 10-Sep-2018].

[30] Red Hat, “What is Object/Relational Mapping?,” *Hibernate Overview*, 2014. [Online]. Available: http://www.hibernate.org/about/orm. [Accessed: 12-Mar-2018].

[31] D. Graham, E. Van Veenendaal, and I. Evans, *Foundations of software testing: ISTQB*

certification. 2008.

[32] “JUnit - About.” [Online]. Available: https://junit.org/junit4/. [Accessed: 06-Jul-2018].

[33] “What is Scrum?” [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Accessed: 25-Aug-2018].

[34] “Scrum: A new perspective on agile development · bounswe/bounswe2016group6 Wiki · GitHub,” 2016. [Online]. Available: https://github.com/bounswe/bounswe2016group6/wiki/Scrum:-A-new-perspective-on-agile-development. [Accessed: 30-Aug-2018].

[35] “Scrum Guide | Scrum Guides.” [Online]. Available: https://www.scrumguides.org/scrum-guide.html. [Accessed: 25-Aug-2018].

[36] M. Rouse, “What is integrated development environment (IDE)? - Definition from WhatIs.com,” 2016. [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/integrated-development-environment. [Accessed: 17-Jul-2018].

[37] M. Rouse, “What is Java IDE? - Definition from WhatIs.com,” 2016. [Online]. Available: https://www.theserverside.com/definition/Java-IDE. [Accessed: 13-Aug-2018].

[38] “Git - About Version Control.” [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. [Accessed: 13-Aug-2018].

[39] “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 13-Aug-2018].

[40] “About – TortoiseGit – Windows Shell Interface to Git.” [Online]. Available: https://tortoisegit.org/about/. [Accessed: 26-Aug-2018].

[41] “Sourcetree | Free Git GUI for Mac and Windows.” [Online]. Available: https://www.sourcetreeapp.com/. [Accessed: 26-Aug-2018].

[42] “Redmine plugins, hosting, solutions - Easy Redmine.” [Online]. Available: https://www.easyredmine.com/. [Accessed: 26-Aug-2018].

[43] “Easy Redmine 2018 features - Easy Redmine.” [Online]. Available: https://www.easyredmine.com/software/easy-redmine. [Accessed: 26-Aug-2018].

[44] “iText PDF, easy PDF generation for Java or .NET developers | iText.” [Online]. Available: https://itextpdf.com/. [Accessed: 16-Apr-2018].

[45] “Apache PDFBox | A Java PDF Library.” [Online]. Available: https://pdfbox.apache.org/. [Accessed: 16-Apr-2018].

[46] “Jaspersoft® Studio | Jaspersoft Community.” [Online]. Available: https://community.jaspersoft.com/project/jaspersoft-studio. [Accessed: 16-Apr-2018].

[47] “GitHub - LibrePDF/OpenPDF: OpenPDF is a free Java library for creating and editing PDF files with a LGPL and MPL open source license. OpenPDF is based on a fork of iText. We welcome contributions from other developers. Please feel free to submit pull-requ.” [Online]. Available: https://github.com/LibrePDF/OpenPDF. [Accessed: 16-Apr-2018].

[48] “GitHub - flyingsaucerproject/flyingsaucer: XML/XHTML and CSS 2.1 renderer in pure Java.” [Online]. Available: https://github.com/flyingsaucerproject/flyingsaucer. [Accessed: 11-Apr-2018].

[49] “Thymeleaf.” [Online]. Available: https://www.thymeleaf.org/. [Accessed: 11-Apr-2018].

[50] T. Uhrig, “Generating PDFs with Java, Flying Saucer and Thymeleaf (Part 1) – Thomas Uhrig,” 2017. [Online]. Available: https://tuhrig.de/generating-pdfs-with-java-flying-saucer-and-thymeleaf/. [Accessed: 10-Apr-2018].

[51] T. Uhrig, “Generating PDFs with Java, Flying Saucer and Thymeleaf (Part 2) – Thomas Uhrig,” 2018. [Online]. Available: https://tuhrig.de/generating-pdfs-with-java-flying-saucer-and-thymeleaf-part-2/. [Accessed: 10-Apr-2018].

[52] M. Collovati, “Enhanced Window Opener | Vaadin Directory | Vaadin,” 2018. [Online]. Available: https://vaadin.com/directory/component/enhanced-window-opener. [Accessed: 30-Apr-2018].

[53] “Selenium - Web Browser Automation.” [Online]. Available: https://www.seleniumhq.org/. [Accessed: 09-Sep-2018].

[54] “Git - Stashing.” [Online]. Available: https://git-scm.com/book/en/v1/Git-Tools-Stashing. [Accessed: 15-Aug-2018].

[55] M. Fowler, “P of EAA: Data Transfer Object.” [Online]. Available: https://martinfowler.com/eaaCatalog/dataTransferObject.html. [Accessed: 08-Sep-2018].

Anexos

Anexo 1: Planeamento inicial do estágio

Tabela 4 - Planeamento inicial do trabalho

TABLE C – WORK PLAN (9 MONTHS)

ACTION ITEMS (ADD LINES IF NEEDED)	BEGIN DATE	END DATE
Training in business context and technology	01/10/2017 (indicative)	31/10/2017
Analyzis product requirements	01/11/2017	31/12/2017
Iterative development cycle of feature : algorithm to compute order proposals by otimizing truck load	01/01/2018	15/02/2018
Iterative development cycle of feature : provide inventory visibility	16/02/2018	31/03/2018
Iterative development cycle of feature : support connectivity with legacy systems	01/04/2018	30/04/2018
Iterative development cycle of feature : measuring performance of VMI software	01/05/2018	31/05/2018
Deployment guides	01/06/2018	30/06/2018
		30/06/2018

Anexo 2: Lista de classes com testes agendados

Tabela 5 – Lista inicial de classes com testes para corrigir e/ou realizar (1ª Fase)

Package	Classe	Estado
net.interlog.isp.ocs.engine.dao	AffiliationDAOImplTest.java	OK
net.interlog.isp.ocs.engine.dao	AssortmentDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	BrandDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	DayOffDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	ExclusionPeriodDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	InactivityReasonDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	InventoryDataDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	InventoryDataLineDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	LineDateDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	MarketDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	MovementDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	OriginDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	ProductDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	ProductFamilyDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	PromotionalCampaignDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	PromotionalCampaignLineDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	ReplenishmentDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	ReplenishmentLineDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	SeasonalityDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	TransportModeDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	WarehouseDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	WarehouseMarketParamsDAOImplTest.java	TO DO
net.interlog.isp.ocs.engine.dao	WarehouseReplenishmentDAOImplTest.java	TO DO

Tabela 6 – Lista inicial de classes com testes para corrigir e/ou realizar (2ª Fase)

Package	Classe	Estado
net.interlog.isp.ocs.engine.business	AffiliationValidatorTest.java	OK
net.interlog.isp.ocs.engine.business	BrandValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	DayOffValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	EdiProductLinkValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ExcludedDayValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	MarketValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	MissingReasonValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	MovementMappingValueLineValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	MovementMappingValueValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	MovementValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	OrderNumberingRangeValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	OrderProposalLineValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	OrderProposalValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	OriginValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ProcessTrackingActionValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ProcessTrackingValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ProductFamilyValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ProductValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	PromotionalCampaignLineValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	PromotionalCampaignValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	ReferenceValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	SeasonalityValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	TransportModeValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	WarehouseMarketParamsValidatorTest.java	TO DO
net.interlog.isp.ocs.engine.business	WarehouseValidatorTest.java	TO DO

Anexo 3: Reporting Requirements

Auteur	Date	Objet	Vers.
Claude SOURTI	01/12/17	Initial version	v1.0
Claude SOURTI	12/12/17	Add quality service analysis	V1.1
Claude SOURTI	02/01/18	net service level + home	v1.2
Claude SOURTI	18/01/18	Add out of stock analysis	v1.3
Claude SOURTI	05/03/208	Add optimal logistics unit analysis	v1.4

Period definition

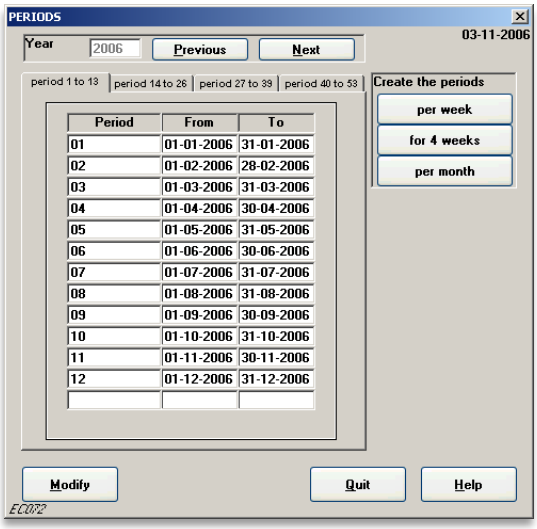
Use case

Business case

The year can be split into periods to do the analysis :

- **[By week**] divide the year into 52 or 53 weeks.
- **[For 4 weeks**] divide the year with one period every 4 weeks.
- [**By month**] divide automatically the year in 12 periods that fit with civil months.

See screenshot below of the screen to define periods in current OCS version.



Business rules

Periods are defined overall for the application.

Only one period type can be defined at a time.

The start of a period must follow the end of the previous one. There must be no missing date.

After the user has selected the period type (per week, for 4 weeks, per month), the system pre-fills the periods.

When the user clicks on “Modify” button, the system updates this data in the database.

Development goal

1) Add sub-menu “Définir les périodes” in main menu “Reporting”.

2) Add a view and new API REST methods to manage the periods.

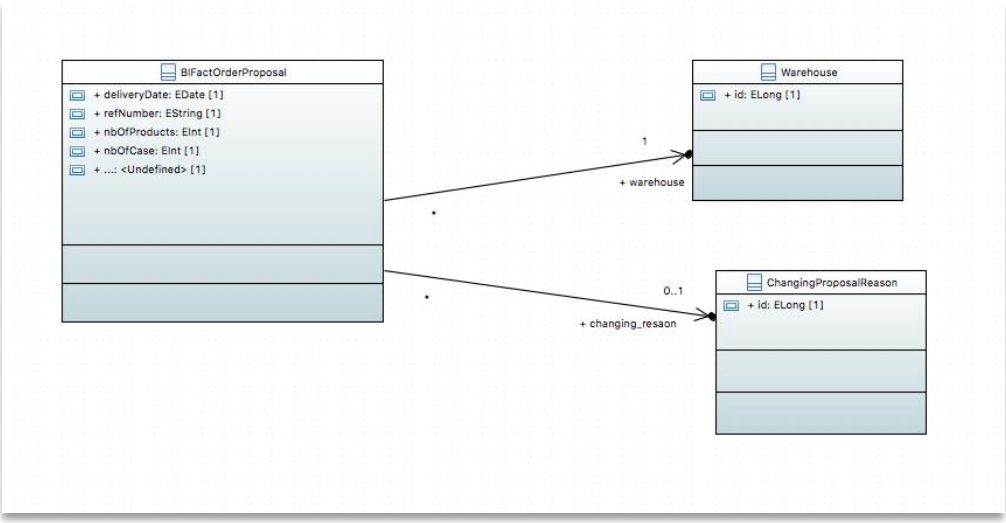
Service quality measurement

Use case

Business rules

This analysis is used to measure the quality of service with as main indicator the availability (in percentage) being the ratio between OOS (Out-Of-Stock, missing quantities) and delivered quantities.

Considering the class diagram below, the quality of service is computed from the contents of “BIFactOrderProposal”. This entity contains the order proposal history. It is populated by OCS when an order proposal (OrderProposalEntity) is validated by the user.



The table below shows the correspondence of all the entity's business data.

Attribute Name	Correspondence	Comment
deliveryDate (*)	Delivery date of order proposal	
warehouse (*)	Warehouse for which the order proposal was created	Link to an existing warehouse
refNumber (*)	Reference number of the order proposal	
changingReason	Cause of change if the proposal has been changed	Link to ChangingProposalReason
nbOfProducts	Number of products in the order proposal	(integer : 3 characters)
nbOfCases	Number of cases in the order proposal	(integer : 3 char.)
nbFloorPallets	Number of floor pallets	(integer : 3 char.)
nbUnitOfPallet	Equivalent of units of pallet	(integer : 3 char.)
nbMixedPallets	Number of mixed pallets	(integer : 3 char.)
nbPoints	Equivalent of points	(integer :10 char.)
weight	Weight of goods (kg) in the order proposal	(integer : 7 char.)
originalDeliveryDate	Original delivery date in case of the proposal has been changed	
refOrder	Order reference number	(String : 10 char.)

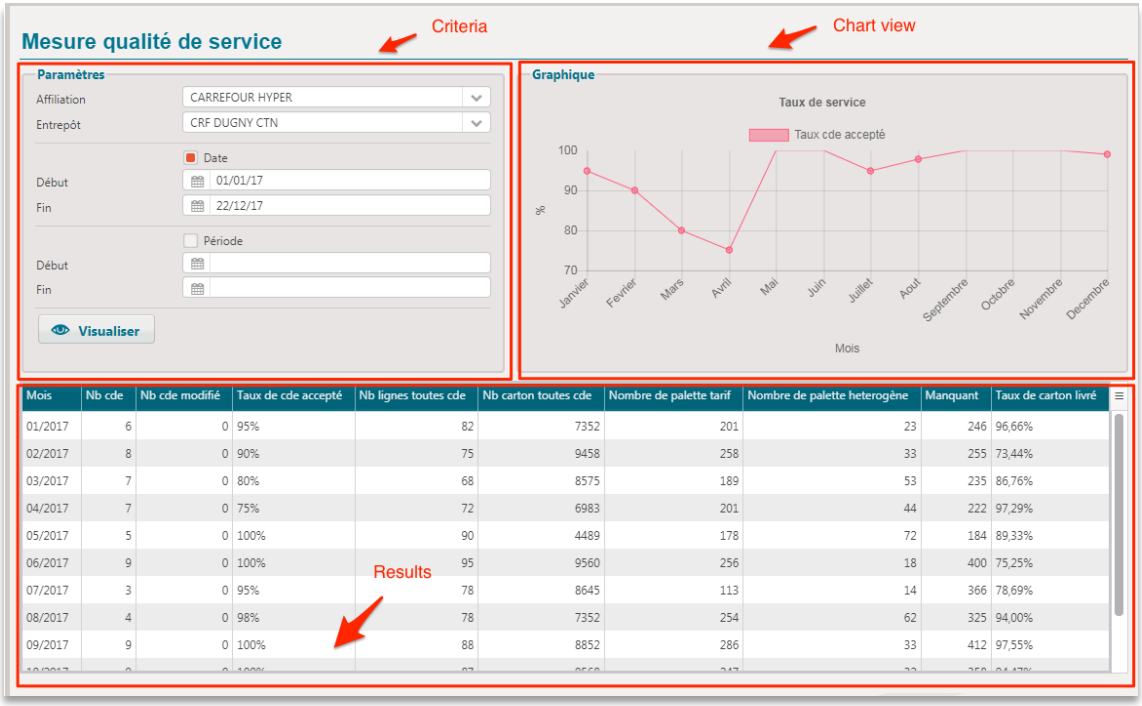
creationDate	Order proposal creation date	
switchDate	Date on which the order proposal has been changed	
origNbFloorPallets	Original number of floor pallets	
origNbUnitOfPallet		
origNbMixedPallets		
origNbPoints		

(*) Business unique key of this entity.

Business case #1 - Quality service analysis

From a sub-menu “Mesure de la qualité de service” in main menu “Reporting”, this view allows the user to display a service quality report for one or more warehouses after selecting some criteria.

See below the representation of this report with 3 parts.



When the user accesses to the view, chart and results list are not displayed. The user must enter the criteria to view the report.

Criteria area

The criteria to be defined are as follows:

- Affiliation: a selected value in the list of the affiliation’s set up.
- Warehouse: a warehouse defined in the system whose the affiliation is equal to the value defined above. It is possible to choose all the warehouses (value = “Tous”) of the selected affiliation.
- Exclusively select start and end months or start and end user-defined periods. In all case, this defines the reporting period.

The required fields are: affiliation, warehouse, reporting period (by month or by user-defined period of time).

Result area

The result is displayed as a grid (customizable and exportable). See development guidelines

The data are aggregated by warehouse and occurrences of reporting period. Every fact data (an occurrence of the object “BIFactOrderProposal”) is aggregated according to two dimensions:

- warehouse
- delivery date grouped by period of time: date included in the selected user-defined period of time or in the selected month

For each aggregate, the data in the table below are calculated according to the rules defined in the 2nd column.

Attribute name	Aggregate rule	Colum name in French	Comment
warehouseCode	Code of the warehouse aggregated	Code de l'entrepôt	
warehouseName	Name of the warehouse aggregated	Nom de l'entrepôt	
Month	Month (format mm/yyyy)	Mois	Only displayed if the user selected the month criteria
Period	Period (format id-period/yyyy)	Période	Only displayed if the user selected the period criteria
nbOfOrders	Number of orders in the time period	Nbre de Cdes	
nbOfOrdersChanged	Number of orders in the time period where attribute "changingReason" is not empty	Cdes modif.	
rateOfOrdersAccepted	$tx = \left(\left(100 - \frac{nbOfOrdersChanged}{nbOfOrders}\right)\right) * 100$	Tx Cdes accept.	
totalOfProducts	Sum of nbOfProducts	Nbre de lignes Toutes Cdes	
totalOfCases	Sum of nbCases	Nbre Car. Toutes Cdes	
totalUnitOfPallet	Sum of nbUnitOfPallet	Nbre Pal. Tarif	
totalMixedPallets	Sum of nbMixedPallets	Nbre Pal. Hétéro.	
totalOfMissingCases	Sum of missing cases in the period (*)	Manquants	
rateOfGrossServiceLevel	$tx = \left(\left(100 - \frac{totalOfMissingCases}{totalOfCases}\right)\right) * 100$	Tx Car. Livr.	
totalOfProviderMissingCases	Sum of missing cases in the period linked to the provider ([2])	Manquants four.	
rateOfNetServiceLevel	$tx = \left(\left(100 - \frac{totalOfProviderMissingCases}{totalOfCases}\right)\right) * 100$	Tx Net Car. Livr.	

(*) The missing quantity (in terms of case) is determined by selecting the movements (`MovementEntity`) of type `MovementTypes.OUT_OF_STOCK` and `MovementTypes.PROMOTIONAL_OUT_OF_STOCK` for the time period and selected warehouse. The warehouse is deduced from the assortment link to the movement. An assortment links a product and a warehouse. The amount of movement is expressed in cases (`MovementEntity.quantity`).

([2]) Apply the same rule as defined below, but select only movements for which the reason (`MissingReasonEntity`) whose responsibility is attributed to the provider (supplier).

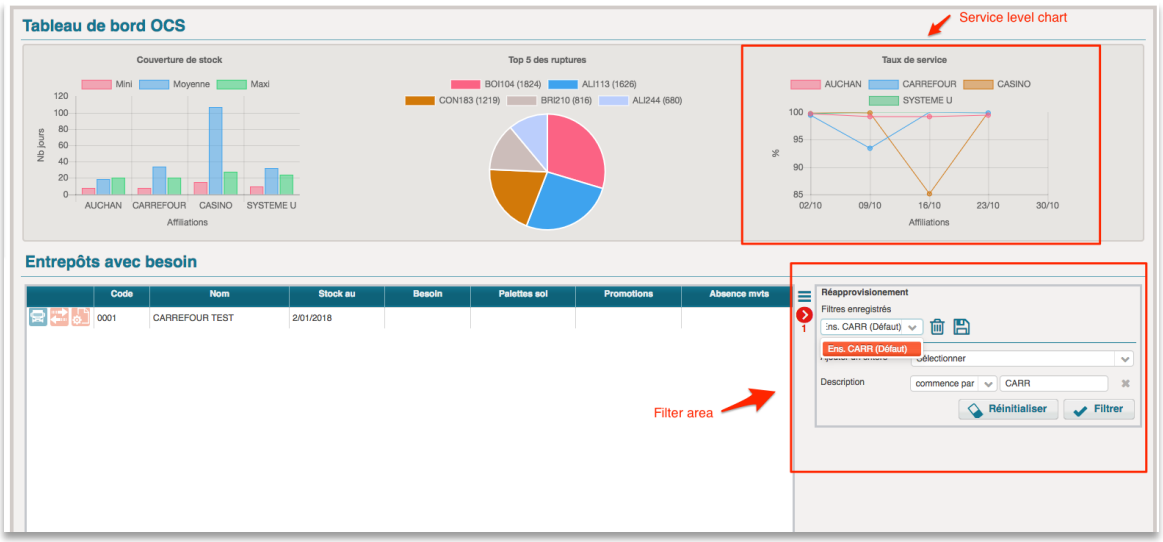
Chart area

The chart have to show the evolution of the service level rate over the reporting period:

- X-Axis: reporting period (by month ou by user-defined period of time)
- Y-Axis: service level in percentage
- Data series: one by selected warehouse
- Legend of data series: name of warehouses

Business case #2 - Quality service analysis

On the home page (*homelistview*), a chart displays the rate evolution over the last 4 weeks group by affiliation. This rate represents gross service level in considering the warehouses in the system.



The warehouses are filtered by applying the default filter defined by the user on the list “Warehouses with needs”. If no default filter is set by the user, the system takes all the warehouses defined.

Development goal

1) Create entity BIFactOrderProposal and DAO layer to create, update, delete it.

2) Add API REST methods to compute quality service analysis

3) Develop view to display “Business case #1 - Quality service analysis” from the existing view named “QualityServiceReportView”. For the grid, you have to replace generic type (String) by an xxxType object.

4) Develop chart to display “Business case #2 - Quality service analysis”.

To display result area use a class that extends AbstractISPListView<xxxType>. See example OrderProposalFormEditView.

Out of stock analysis

Use case

Business rules

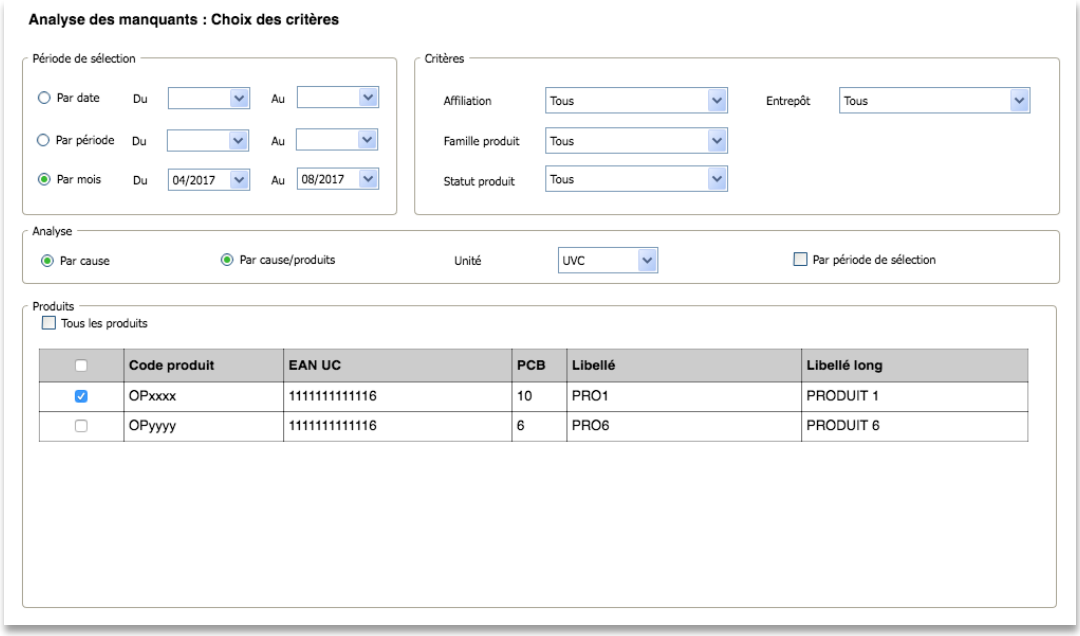
This analysis is used to review the different reasons for missing items (Out-Of-Stock). The analysis can be done according to several axes using different criteria.

The out of stock analysis is computed from the contents of “Movement” entity. The missing quantity (in terms of case) is determined by selecting the movements (MovementEntity) of type `MovementTypes.OUT_OF_STOCK` (for trading status = standard) and `MovementTypes.PROMOTIONAL_OUT_OF_STOCK` (for trading status = promotional).

Business case #1 – Out of stock analysis

From a sub-menu “Analyse des manquants” in main menu “Reporting”, this view allows the user to display the report of missing items for one or more warehouses, or product families after selecting some criteria.

The first screen allows the user to define criteria to apply to the report (see the sreen template below).



Criteria selection

The user must choose at least one of these date criteria :

- By date : select of start and end date in a calendar (start date must be less than the end date).

- By period : select of start and end period in the user-defined periods of time.
- By month : select start and end month.

To limit the report, the user can choose :

- one affiliation or all (French = Tous). If he chooses one affiliation, the list of warehouses under must be filter considering the affiliation selected.
- one family product or for all.
- one trading status (cf. `TradingStatusTypes`) or all.

Next the user can choose the type of analysis : by reason or by reason and product. He must choose the unit for computing and displaying quantities : in consumer unit (UVC) or in case (Carton). The option per selecting period (par période de sélection) specifies if the data will be aggregated by period or cumulated over the defined selecting period.

Next the user can choose to apply the analysis to all products (by selecting the option “tous les produits”, by default this option is enabled) or select a list of products. To select a the list of products, the user can uncheck the previous option. In this case, the list of products is displayed and he can use the filter (to enable in this case) to restrict the list. The list of columns by default is : product code (French = Code produit), EAN CU (EAN UC), Number CU per case (PCB), short description (Désignation), long description (Libellé long). The user can add another attribute of the product by using the option provided for this purpose.

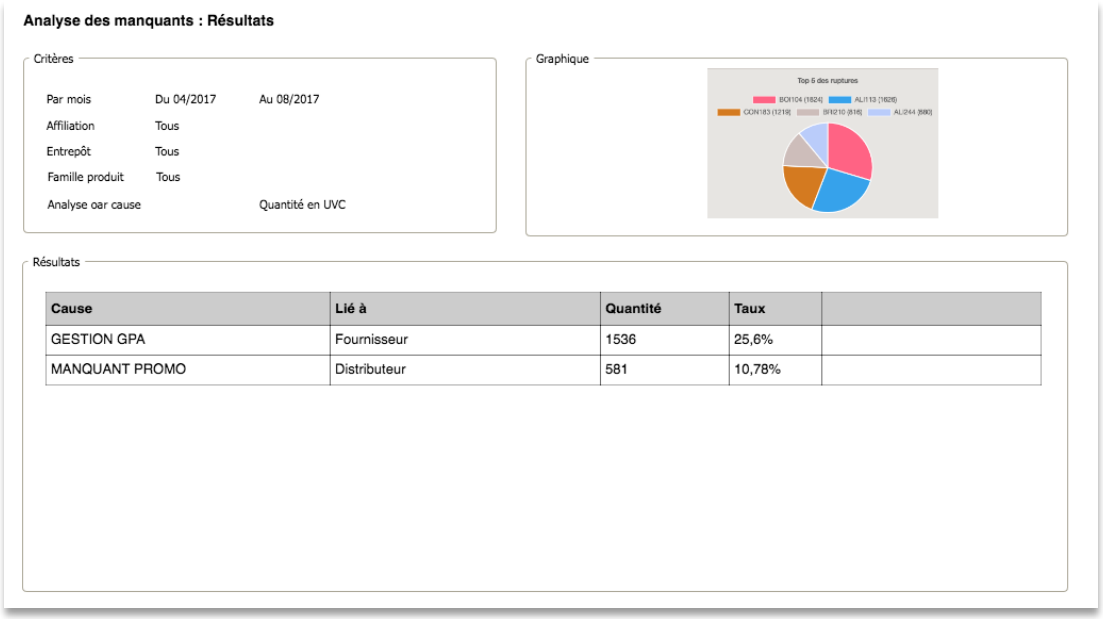
After having chosen the criteria, the user validates his selection by clicking on the visualize button (Visualiser) and then redirected to the page of results (see below the template of this page).

Result view of analysis by reason

The result is displayed as a grid (customizable and exportable). At the top of the screen, there are a summary of the selected criteria and a graphic area.

The warehouse is deduced from the assortment link to the movement. An assortment links a product and a warehouse. The family product is deduced from the product link to the assortment related to the movement.

The movement quantity is expressed in cases (`MovementEntity.quantity`).



The data are aggregated by missing reason linked to movements of “out-of-stock” type and occurrences of reporting period in case of option per selecting period, otherwise the data are cumulated over the period. Every fact data (an occurrence of the object “Movement” where type = `MovementTypes.OUT_OF_STOCK` or `MovementTypes.PROMOTIONAL_OUT_OF_STOCK`) is aggregated according to the dimensions:

- missing reason
- ([1]) movement date grouped by period of time: date included in the selected user-defined period of time or in the selected month or date

(1) Only if the option per selecting period has been selected.

For each aggregate, the data in the table below are calculated according to the rules defined in the 2nd column.

Attribute name	Aggregate rule	Colum name in French	Comment
Reason	Description of the missing reason aggregated	Raison	
Responsibility	Responsibility of the missing reason	Lié à	
Month	Month (format mm/yyyy)	Mois	Only displayed if the user selected the month criteria and option per selecting period
Period	Period (format id-period/yyyy)	Période	Only displayed if the user selected the period criteria and option per selecting period
missingQuantity	Sum of movement quantity for the current reason	Quantité	Expressed in the unit selected in the previous screen
rateOfMissingQuantity	$\text{tx} = \left(\frac{\text{MissingQuantity}}{\text{TotalOfMissingQuantity}}\right) * 100$	Taux	The calculations are based on the unit selected previously
totalOfMissingQuantity	Sum of all missing quantities	Total	Expressed in the unit selected in the previous screen

The graphic area displays the distribution of reasons by rate.

Result view of analysis by reason

The result is still displayed as a grid (customizable and exportable).

The data are aggregated by product (deduced from the assortment link to the movement), and by missing reason linked to movements of "out-of-stock" type and occurrences of reporting period in case of option per selecting period, otherwise the data are cumulated over the period. Every fact data (an occurrence of the object "Movement" where type = `MovementTypes.OUT_OF_STOCK` or `MovementTypes.PROMOTIONAL_OUT_OF_STOCK`) is aggregated according to the dimensions:

- missing reason
- product
- ([1]) movement date grouped by period of time: date included in the selected user-defined period of time or in the selected month or date

(1) Only if the option per selecting period has been selected.
For each aggregate, the data are calculated according to the same rules defined above. The product information will be added :

- Product code (Code produit)
- EAN (EAN)
- Number CU per case (PCB)
- Short description (Désignation)
- Long description (Libellé long)
- Trading status (statut commercial)

The user can add another attribute of the product by using the option provided for this purpose in the grid.

The graphic area displays “top 5” of products with the largest quantities of missing items. For each product, display the product and code and missing quantity expressed in the unit selected. The values of chart are given the rate.

Business case #2 – Out of stock analysis

On the home page (*homelistview*), a chart displays the top 5 of rate products with the largest quantities of missing items over the last 4 weeks. For each product, display the product code and missing quantity expressed in the unit selected (default : case). The values of chart are given the rate.

This rate represents gross service level in considering the warehouses in the system.

The out of stock items depend on the warehouses that are filtered by applying the default filter defined by the user on the list “Warehouses with needs”. If no default filter is set by the user, the system takes all the warehouses defined.

Development goal

1) Add API REST methods to compute out of stock analysis

2) Develop view to display “Business case #1 – Out of stock analysis”.

3) Develop chart to display “Business case #2 – Out of stock analysis” in order to replace the static chart in homelistview.

Optimal logistics unit analysis

Use case

Business rules

This analysis is used to define the “right” replenishment unit for assortments (assortment = product in a warehouse) considering the last outgoings and according to the goals of quality of service.

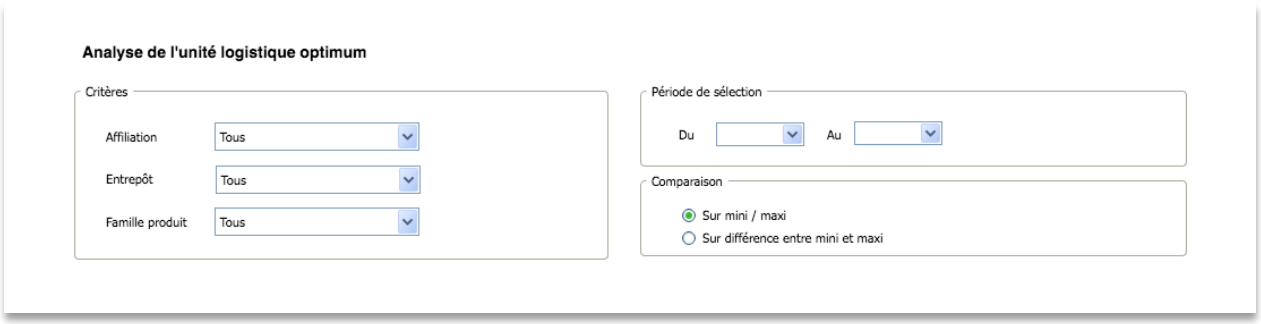
The report is created from the outputs of the selected period and takes only into account the standard products.

The analysis is applied to the assortments of selected warehouses.

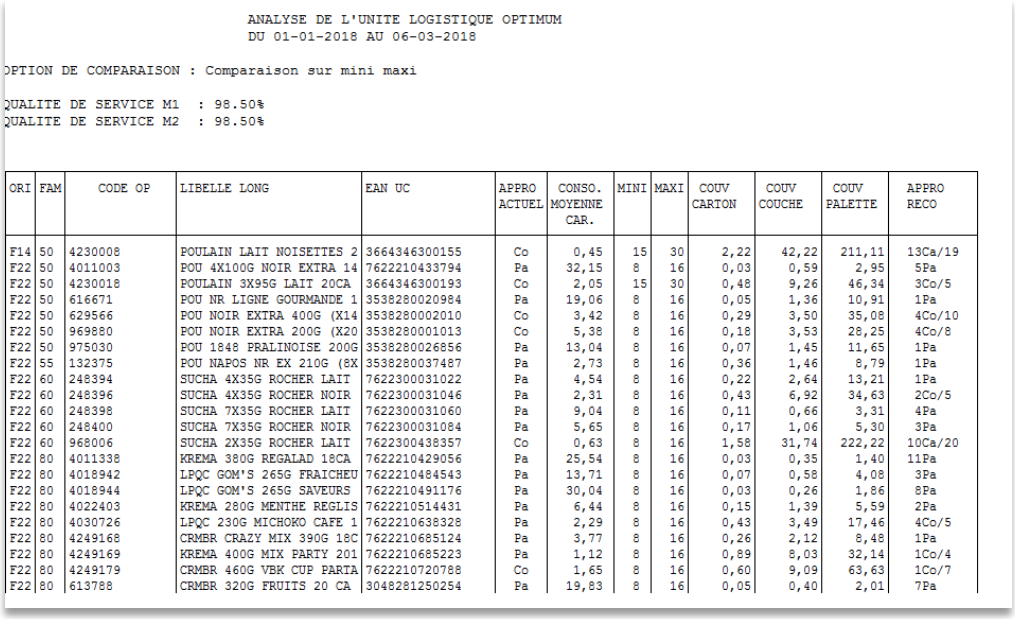
Business case #1 – Optimal logistics unit analysis

From a sub-menu “Analyse de l’unité logistique optimum” in main menu “Reporting”, this view allows the user to display the report of optimal logistics unit analysis for one or more warehouses, or product families after selecting some criteria.

The first screen allows the user to define criteria to apply to the report (see the sreen template below).



Example of the result.



ANALYSE DE L'UNITE LOGISTIQUE OPTIMUM
DU 01-01-2018 AU 06-03-2018

OPTION DE COMPARAISON : Comparaison sur mini maxi

QUALITE DE SERVICE M1 : 98.50%
QUALITE DE SERVICE M2 : 98.50%

ORI	FAM	CODE OP	LIBELLE LONG	EAN UC	APPRO ACTUEL	CONSO. MOYENNE CAR.	MINI	MAXI	COUV CARTON	COUV COUCHE	COUV PALETTE	APPRO RECO
F14	50	4230008	POULAIN LAIT NOISETTES 2	3664346300155	Co	0,45	15	30	2,22	42,22	211,11	13Ca/19
F22	50	4011003	POU 4X100G NOIR EXTRA 14	7622210433794	Pa	32,15	8	16	0,03	0,59	2,95	5Pa
F22	50	4230018	POULAIN 3X95G LAIT 20CA	3664346300193	Co	2,05	15	30	0,48	9,26	46,34	3Co/5
F22	50	616671	POU NR LIGNE GOURMANDE 1	3538280020984	Pa	19,06	8	16	0,05	1,36	10,91	1Pa
F22	50	629566	POU NOIR EXTRA 400G (X14	3538280002010	Co	3,42	8	16	0,29	3,50	35,08	4Co/10
F22	50	969880	POU NOIR EXTRA 200G (X20	3538280001013	Co	5,38	8	16	0,18	3,53	28,25	4Co/8
F22	50	975030	POU 1848 PRALINOISE 200G	3538280026856	Pa	13,04	8	16	0,07	1,45	11,65	1Pa
F22	55	132375	POU NAPOS NR EX 210G (8X	3538280037487	Pa	2,73	8	16	0,36	1,46	8,79	1Pa
F22	60	248394	SUCHA 4X35G ROCHER LAIT	7622300031022	Pa	4,54	8	16	0,22	2,64	13,21	1Pa
F22	60	248396	SUCHA 4X35G ROCHER NOIR	7622300031046	Pa	2,31	8	16	0,43	6,92	34,63	2Co/5
F22	60	248398	SUCHA 7X35G ROCHER LAIT	7622300031060	Pa	9,04	8	16	0,11	0,66	3,31	4Pa
F22	60	248400	SUCHA 7X35G ROCHER NOIR	7622300031084	Pa	5,65	8	16	0,17	1,06	5,30	3Pa
F22	60	968006	SUCHA 2X35G ROCHER LAIT	7622300438357	Co	0,63	8	16	1,58	31,74	222,22	10Ca/20
F22	80	4011338	KREMA 380G REGALAD 18CA	7622210429056	Pa	25,54	8	16	0,03	0,35	1,40	11Pa
F22	80	4018942	LPQC GOM'S 265G FRAICHEU	7622210484543	Pa	13,71	8	16	0,07	0,58	4,08	3Pa
F22	80	4018944	LPQC GOM'S 265G SAVEURS	7622210491176	Pa	30,04	8	16	0,03	0,26	1,86	8Pa
F22	80	4022403	KREMA 280G MENTHE REGLIS	7622210514431	Pa	6,44	8	16	0,15	1,39	5,59	2Pa
F22	80	4030726	LPQC 230G MICHOKO CAFE 1	7622210638328	Pa	2,29	8	16	0,43	3,49	17,46	4Co/5
F22	80	4249168	CRMBR CRAZY MIX 390G 18C	7622210685124	Pa	3,77	8	16	0,26	2,12	8,48	1Pa
F22	80	4249169	KREMA 400G MIX PARTY 201	7622210685223	Pa	1,12	8	16	0,89	8,03	32,14	1Co/4
F22	80	4249179	CRMBR 460G VBK CUP PARTA	7622210720788	Co	1,65	8	16	0,60	9,09	63,63	1Co/7
F22	80	613788	CRMBR 320G FRUITS 20 CA	3048281250254	Pa	19,83	8	16	0,05	0,40	2,01	7Pa

For each assortment, the data in the table below are calculated according to the rules defined in the 2nd column.

Attribute name	Calculated rule	Colum name in French	Comment
Warehouse code	Warehouse code of the assortment	Code entrepôt	
Warehouse description	Warehouse description of the assortment	Nom entrepôt	
Product code	Product code associated to the assortment	Code produit	
Product long description	Product long description	Libellé long	
Product EAN	EAN CU of the product	EAN UC	
	(*) All attributes of product can be added with the option "Configurer" of the grid		By default, these attributes are hidden.
Current replenishment type	Replenishment unit defined	Appro. Actuel	Displayed translates values : Pallet, Layer, Case
Average output	Calculated from the number of working days. Used existing method (see below 1)	Conso. Moy. (Car.)	Expressed in number of cases
Minimum stock	Minimum stock of days defined for the assortment	Mini	Number of days
Maximum stock	Maximum stock of days defined for the assortment	Maxi	Number of days
Case coverage	$\text{cover} = \left(\frac{\text{Product.nbCuPerCase}}{\text{Average output}}\right)$	Couv. Carton	Expressed in days, round with 2 decimals
Layer coverage	$\text{cover} = \left(\frac{\text{Product.nbCasePerLayer}}{\text{Average output}}\right)$	Couv. Couche	Expressed in days, round with 2 decimals
Pallet coverage	$\text{cover} = \left(\frac{\text{Product.nbCasePerPallet}}{\text{Average output}}\right)$	Couv. Palette	Expressed in days, round with 2 decimals
Recommended replenishment unit	See below 2.	Appro. Reco	

1. Used method `AssortmentBusiness. computeAverage(AssortmentEntity assortment, List<MovementEntity> mvts, Date refDate, int nbHistoricalDays)` with :
    a. *refDate* = End date define for the period of analysis,

b. *nbHistoricalDays* = number of days between start and end date of analysis period.
c. *mvts* can load using method `IMovementDAO.findByAssortmentIdAndType(Long idAssortment, MovementTypes type, 52)`.

2. Recommended unit is computed in two steps: 1. identify the optimum replenishment unit, 2. calculate the optimum number of this unit.
   a. Identify the optimum replenisheement unit
      i. Select the comparison method:

If comparison = min-max (OCS will compare the cover of the various replenishment units with the min and the max in order to choose the optimal logistic unit):

CompMax = `Assortment.getMaxStockDays()`

CompMin = `Assortment.getMinStockDays()`

If comparison = difference between min and max (OCS will compare the cover of the various replenishment units with the difference between min and max):

CompMax = `Assortment.getMaxStockDays()` - `Assortment.getMinStockDays()`

CompMin = CompMax

      ii. Select the replenishment unit:

By default, selected replenishment unit = Pallet.

If pallet coverage is greater than *CompMax*, selected replenishment unit = layer.

If layer coverage is greater than *CompMax*, selected replenishment unit = case.

   a. Calculate the optimum number of this unit

Calculate the recommended unit requirement to get as near as possible to CompMax.

Recommended number of units (rounded up to the nearest integer):

$$\text{nb of units} = \left(\frac{\text{CompMax}}{\text{Coverage of unit selected}}\right)$$

Rules:

- If united selected = pallet, then coverage = pallet coverage, united selected = layer then coverage = layer coverage, …

If Recommended Unit = 0, then Recommended Unit = 1

Anexo 4: EDI interfaces

E.D.I. Interfaces

Requirements

Auteur	Date	Objet	Vers.
Claude SOURTI	04/10/17	Initial version	v1.0
Claude SOURTI	31/10/17	Add EDI / OCS products links	v1.1
Claude SOURTI	15/01/18	Add locations links	v1.2

Movements Types

Use case

Business case

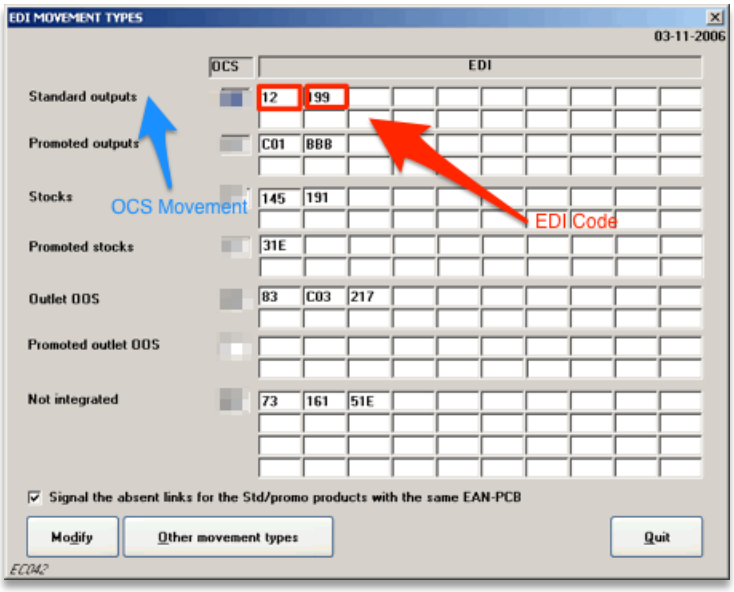
Each movement type (from EDI message), has a specific codification to identify it. According to the customer's needs, specific codes can be added. Each EDI movement type is converted into OCS codes. So, several EDI codes can be grouped in only one OCS movement type.

It is possible to indicate in 'on error' messages the links not created for the standard and promoted products that have the same EAN and PCB.

Business rules

- An EDI code can only be used for one OCS code.
- An EDI code is an alphanumeric string with a maximum of 3 characters.
- OCS codes have the following values and correspond to Enum « MovementTypes » (see isp-ocs-common project) :
  - Stocks levels (`MovementTypes.STOCK_LEVEL`) (French translation : Stocks standards)
  - Promoted Stocks (`MovementTypes.PROMOTIONAL_STOCK_LEVEL`) (Stocks promotionnels)
  - Outputs (`MovementTypes.OUTGOING`) (= Sorties standards)
  - Promoted Outputs (`MovementTypes.PROMOTIONAL_OUTGOING`) (Sorties promotionnelles)
  - Outlet Out Of Stock (`MovementTypes.MISSING`) (Manquants)
  - Promoted Out Of Stock (`MovementTypes.PROMOTIONAL_MISSING`) (Manquants promotionnels)
  - In Transit (`MovementTypes.IN_TRANSIT`) (En cours)
  - Promoted In Transet (`MovementTypes.PROMOTIONAL_IN_TRANSIT`) (En cours promotionnel)
  - Reserved Preorders (`MovementTypes.PRE_ORDER`) (Pré-commandes de réservation)
  - Not Integrated (`MovementTypes.NOT_INTEGRATED`) (Non intégrés)

Below a screenshot of the existing one (But instead of 20 text fields by OCS code, you can imagine a dynamic system to add an EDI code).



Development goal

1) View to manage these code's settings and store its.

2) A service to load these settings in an object of List<MovementMappingValue> (used in process engine to integrate movement informations). One instance of MovementMappingValue for each EDI movement corresponding to an OCS code.

 This list must be managed in cache (https://github.com/jhalterman/expiringmap) that be updated when a value has changed. See VisibilityCacheManager in iso-engine project to have an example of use.

EDI – OCS Products links

Use case

Business case

This table allows having a connection between all the EDI products and the products processed by OCS. It allows controlling the OCS product OP code from several EDI criteria.

The following items from EDI files used to find an OP code:

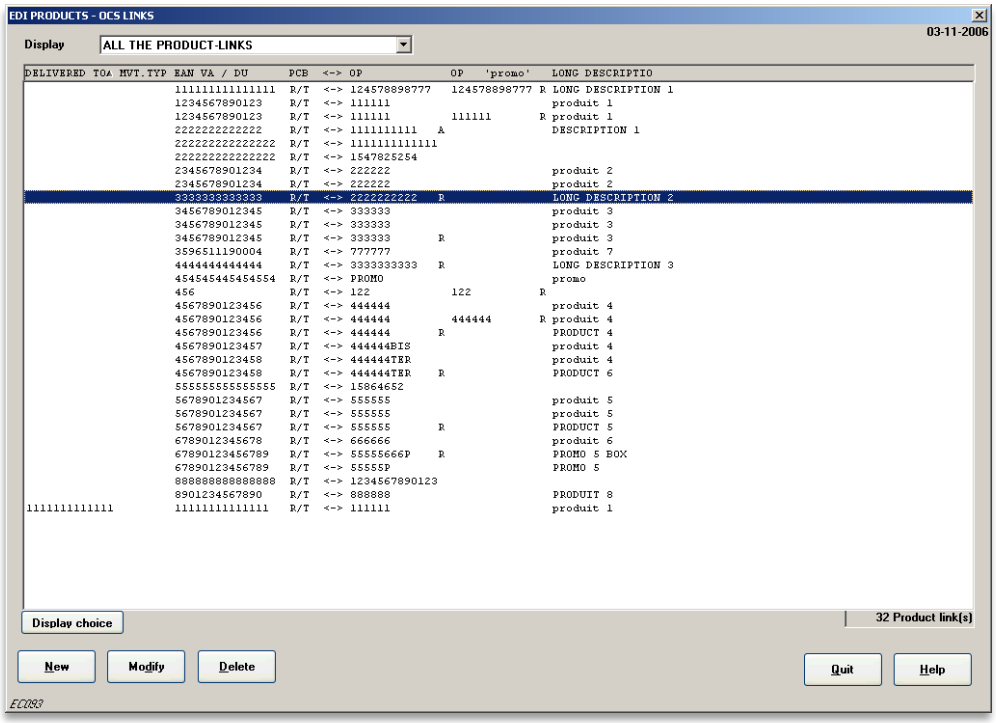
- Delivered to
- Movement type
- EAN + PCB or EAN DU (delivery unit) + DU type (C = TDU, P = Pallet)

In the EDI files, an OP code is identified by a DU EAN or an EAN code + RSU/TDU. The DU EAN code is optional, but if it exists, it has priority.

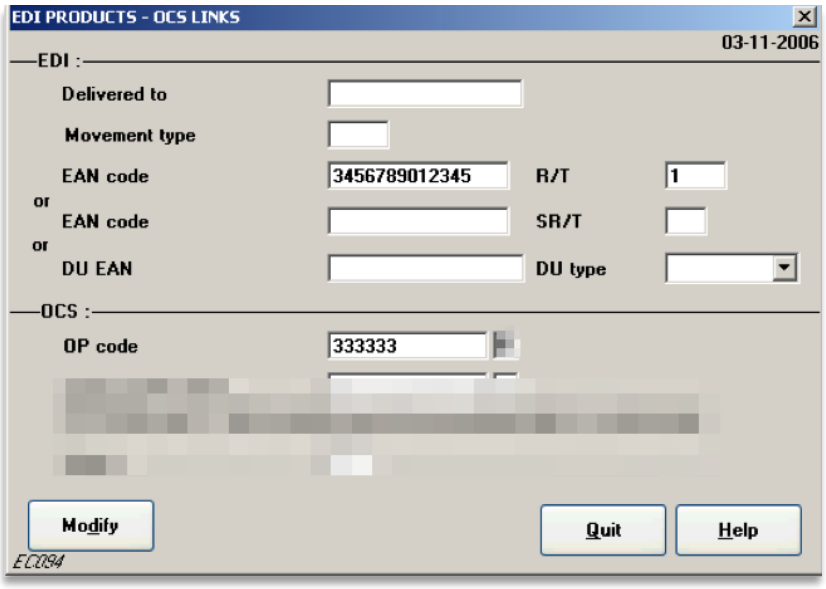
Business rules

This list allows to see mapping EDI – OCS created for products links.

The four first columns display to the EDI configuration. The three last columns display the recorded OCS data.



A form view (see below a screenshot of existing view) allow to create or update a link between a product coming from EDI and a product recorded in OCS



Usually, enter the EAN code, the RSU/TDU value (PCB), the OP code. OP code have to exist in OCS data product. The long description display in the list (see previous page) is deduced from the product with the same OP.

It is possible to create links between OCS OP codes and RSU EAN or DU TDU EAN or DU pallet EAN codes coming from EDI.

If the EAN code is entered, the RSU.TDU value must be entered.

If the DU EAN is entered, the DU type (Case or Pallet, see enum `UnitTypes`) must be entered.

Development goal

1) Build the entity to store these links.
2) Develop layers DAO (with JUnit), service et API to manage the links.
3) Develop list and form view to create and edit a link. Add delete function to remove a link from the list.

Locations links

Use case

Business case

This table allows, from a secondary “delivered to” EAN code, to find the primary “delivered to” and then to select the impacted warehouse.

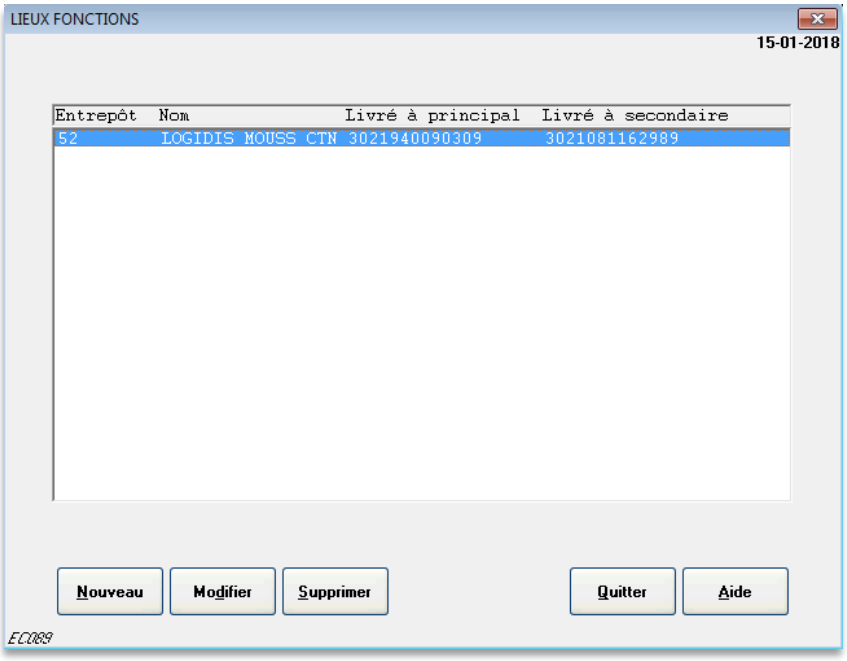
It is used when converting EDI movements and confirmation orders.

Business rules

The properties of a location link are :

- Warehouse : a warehouse defined in the system.
- Secondary code : It defines the code link to the primary warehouse. This code is an alphanumeric string with a maximum of 20 characters.

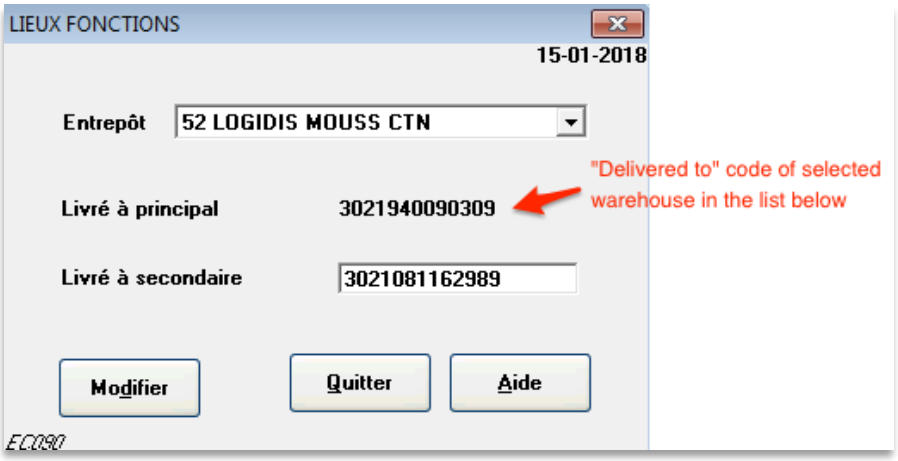
Below a screenshot of the existing form to list locations links.



Because a secondary can only be set once in the system, the secondary code is unique business key.

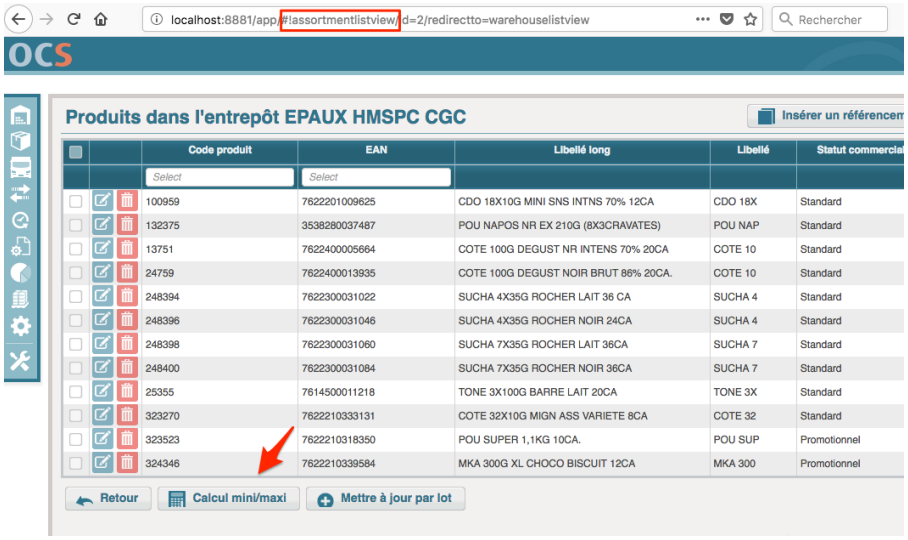
Development goal

1) Build the entity to store these links.
2) Develop layers DAO (with JUnit tests), service et API to manage the links.
3) Add sub-menu “Lieux fonctions” in main menu “Paramètres/Interfaces EDI”.
4) Develop list and form view to create and edit (see the screenshot below for an example of the layout, and for French terms) a link. Add delete function to remove a link from the list.

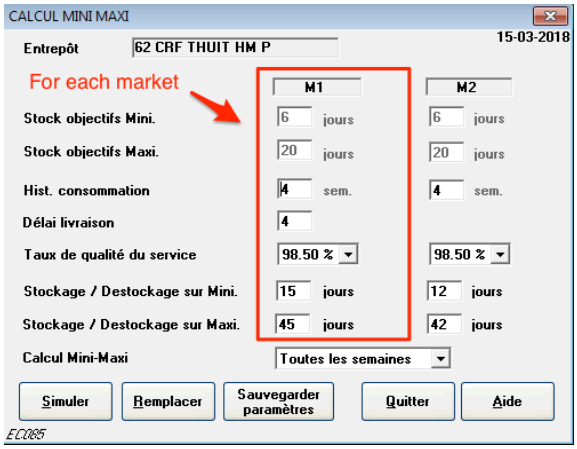


Calcul Min/Max

In the assortment edit view of a warehouse (see screenshot below), the button “Calcul Min/Max” allow to optimize the settings of min/max stock days.



The first step of this feature is to input some parameters to define the goal of optimization. When the user selects the button “Calcul Min/Max”, the user can define the parameters in a view as defined below.



Entrepôt : Warehouse code and name. Read only.

For each market define in the system for the selected warehouse (see entity: *WarehouseMarketParamsEntity)* :

- Stock objectifs Mini. : WarehouseMarketParamsEntity.minStockDays. Read only.
- Stock objectifs Maxi. : WarehouseMarketParamsEntity.maxStockDays. Read only.

- Hist. consommation (*) : By default 4 weeks.
- Délai de livraison (*) : By default WarehouseEntity. replLeadTime
- Taux de qualité de service (*) : an objective of service level, a percentage defined by the user in a predefined list with these 5 values (98.5, 99.0, 99.5, 99.75, 99.99)
- Stockage/Destockage sur Mini. (*) : security stock on mini target.
- Stockage/Destockage sur Maxi. (*) : security stock on maxi target.
- Calcul Mini-Maxi (*) : defined a frequency to update automatically the settings of min/max. 4 values : Every each week, Every 2 weeks, Every 3 weeks or Every 4 weeks.

The first task is to implement a view to input these settings. At the current time, you have to implement only the button “Sauvegarder paramètres” for saving the settings with a (*) in a new entity.

Following, we work together to add the result (a list of assortment) after the computing of min/max values when the user selects “Simuler”.