Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA
DESDE 1911

MESTRADO

GESTÃO EM SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

TRABALHO DE PROJETO

WERESOLV – UM PROJETO DE DESENVOLVIMENTO DE SOFTWARE EM AMBIENTE WEB

HUGO FREDERICO PARDO DE OLIVEIRA CRUZ E SILVA

SETEMBRO - 2013

Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA
DESDE 1911

MESTRADO EM
GESTÃO EM SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO
TRABALHO DE PROJETO

WERESOLV – UM PROJETO DE DESENVOLVIMENTO DE SOFTWARE EM AMBIENTE WEB

HUGO FREDERICO PARDO DE OLIVEIRA CRUZ E SILVA

ORIENTAÇÃO:

PROF. DOUTORA WINNIE PICOTO
ENG. PAULO NINA

SETEMBRO - 2013

RESUMO

Quando uma empresa decide alterar a forma como se visualizam e gerem os dados operacionais, tal operação deve ser efetuada de maneira a que os seus funcionários se adaptem facilmente a essa mudança. É o caso da *Solvay*, empresa do setor químico, onde se iniciou, em Setembro de 2012, um projeto (no qual estive inserido) com vista à criação de uma aplicação *web* que permitisse a visualização, introdução e manipulação de dados fabris a partir de um *browser*, tanto em *desktops* como *tablets*.

É objetivo deste trabalho apresentar e descrever o projeto em que participei durante o meu estágio na *Solvay*. De forma a introduzir as áreas de conhecimento com as quais tive mais contacto no decorrer do projeto, é apresentada uma revisão da literatura nessas mesmas áreas. É objetivo desta revisão criar um *benchmark* que mais tarde permita comparar o que foi realizado durante o projeto com, o que são consideradas pelo conhecimento atual, as melhores práticas destas áreas.

Após a revisão da literatura é apresentado o projeto, descrevendo como foram aplicados princípios ágeis de desenvolvimento (*Extreme Programming*) e os impactos observados da nova aplicação nos métodos de trabalho da fábrica.

Concluo este trabalho com uma avaliação da aplicação e quais as maiores dificuldades encontradas durante o seu desenvolvimento.

Palavras-chave: Desenvolvimento de Software, Desenvolvimento Ágil, *Scrum*, *Extreme Programming*, Gestão de Base de Dados, Design de *Websites*.

Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA
DESDE 1911

ABSTRACT

When a company decides to change the way it views and manages its operational data, said change should be implemented in a way which allows for the company's workforce to easily adapt to the new environment. Such was the case at Solvay, a chemical company, where a project was initiated, in late September 2012, (in which I took part) with the aim of creating a web application that would allow its users to view and manipulate manufactoring data. All feasible through a web browser.

The aim of this work is to present and describe the project in which I participated during my internship at Solvay. To introduce the knowledge areas with which I had most contact during the project, a literature revision is presented. The aim of this revision is to create a benchmark which will help later on to compare what was done during the course of the project with the best practices in these knowledge areas.

Afterwards, I describe the project, detailing how Agile software development principles were applied (*Extreme Programming*) and the impacts the new application had in the factory's workflow.

I conclude this work with an evaluation of the application and issues that arose during its development.

Keywords: Software Development, Agile Development, *Scrum*, *Extreme Programming*, *Database Management*, *Website Design*.

Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA
DESDE 1911

AGRADECIMENTOS

Agradeço em primeiro lugar à Professora Doutora Winnie Picoto pela oportunidade de ingressar num projeto aliciante, bem como a prontidão demonstrada em me orientar neste trabalho. De salientar também a disponibilidade que demonstrou para ajudar sempre que necessário.

Tenho ainda a agradecer à minha família a paciência e apoio que sempre me deram durante este longo percurso. Em especial à Raquel, pela originalidade do nome da aplicação.

Aos Engenheiros Paulo Nina e Rui Rocha por me permitirem recordar o meu primeiro trabalho, no futuro, com algum divertimento e, ao mesmo tempo, um sentimento de realização pessoal.

Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA
DESDE 1911

Índice

1. Introdução

Atualmente, as empresas começam a dar maior importância à gestão e ao consumo da informação que é produzida internamente. A informação é agora encarada como uma vantagem competitiva no apoio às decisões. Existem contudo, empresas que ainda fazem o seu consumo de dados e informação através de meios que se tornam, com o passar do tempo, cada vez mais ultrapassados. Foi o caso da empresa onde decorreu o projeto, a *Solvay*. Na monitorização das suas operações os trabalhadores da fábrica utilizam ficheiros *Excel*, que fazem uma ligação à Base de Dados técnica, *Aspen Process Information Management System* (PIMS), e que com recurso a macros de VBA (*Visual Basic for Applications*) extraem os dados para visualização em *spreadsheets*. Esta abordagem origina a criação, manutenção e difusão de um elevado número de ficheiros, tornando difícil efetuar pequenas alterações num desses ficheiros.

Decidida a mudar esta situação, a equipa de gestão da fábrica iniciou um projeto com vista à criação de uma aplicação em ambiente *web* que permitisse ganhos de tempo na consulta e manipulação de dados, bem como a centralização destas mesmas tarefas. Por último, a aplicação deveria ser compatível com, e acessível a partir de, certos dispositivos móveis e *desktops*, desde que devidamente certificados na rede interna da empresa.

Quando finalizada, a aplicação deveria, além de cumprir os objetivos mencionados, ser replicável noutras fábricas com um mínimo de esforço técnico.

Tendo em conta estes pontos, foi decidido que a aplicação não deveria utilizar qualquer tipo de linguagem que requeresse a instalação de *software* do lado do cliente. Se necessário, apenas deveriam ser necessárias instalações no lado do servidor.

É objetivo deste trabalho apresentar e descrever o projeto em que participei durante o meu estágio na *Solvay*. Após uma pesquisa bibliográfica nas áreas de desenvolvimento de *software*, gestão de dados e *web design*, através da qual se identificam aquelas que são consideradas as melhores práticas nestas áreas de conhecimento, descrevo as diferentes fases do projeto desenvolvido de acordo com princípios de *Extreme Programming*, destacando, em cada etapa, os meus contributos pessoais no desenvolvimento da aplicação construída. A concluir apresento uma avaliação desta aplicação e as dificuldades encontradas durante o projeto.

No capítulo seguinte, é apresentada a pesquisa bibliográfica referente a cada uma das áreas de interesse acima referenciadas. Após esse capítulo é apresentado o trabalho realizado na *Solvay*, enumerando os métodos utilizados ao longo do projeto. Por fim, apresentam-se as conclusões deste trabalho, onde é elaborada uma avaliação da aplicação e se fazem considerações relativas à aplicação de certos aspetos teóricos durante o decorrer do projeto.

2. Revisão da Literatura

2.1. Metodologias de Desenvolvimento

Para um projeto de desenvolvimento de *software* ser bem sucedido, devem ser aplicados no decorrer do mesmo os princípios defendidos por uma dada metodologia de desenvolvimento. A mesma deve adequar-se às características do projeto visto que "utilizar um modelo de ciclo de vida de software adequado pode melhorar a eficiência e eficácia do desenvolvimento de software" (Guntamukkala, et al., 2006, p.266). Na secção seguinte, são caracterizadas as metodologias de desenvolvimento em Cascata (servindo de exemplo das metodologias tradicionais), *Extreme Programming* e *Scrum* (representando as correntes mais ágeis de metodologias de desenvolvimento). É no entanto necessário que antes de se proceder a tal caracterização se aborde o conceito de metodologia de desenvolvimento.

Uma metodologia de desenvolvimento é definida por Ramsin e Paige (2008, p.3) como sendo um "modelo de aplicação de práticas de engenharia de *software*, que tem como objetivo específico providenciar os meios necessários ao desenvolvimento de sistemas de *software* intensivos". Já Avison e Fitzgerald (2003, p.80) definem metodologia de desenvolvimento de *software* como uma "coleção de fases, procedimentos, regras, técnicas, ferramentas, documentação, gestão e treino utilizados no desenvolvimento de um sistema".

2.1.1. Modelo em Cascata

As metodologias de desenvolvimento de *software* tradicionais defendem o uso de "planeamento extensivo, processos codificados e reutilização rigorosa para tornar o desenvolvimento uma atividade eficiente e previsível" (Guntamukkala et al., 2006, p.268).

A caracterização apresentada a seguir foi baseada na sumarização apresentada por Munassar e Govardhan (2010), embora também outros estudos tenham sido consultados (Guntamukkala et al., 2006; Sheffield & Lemétayer, 2013; Avison & Fitzgerald, 2003), onde são expostas as diversas fases do ciclo de vida desta metodologia, as suas fraquezas e vantagens.

Este modelo é composto por sete fases: **(i)** Definição de requisitos do sistema, **(ii)** definição de requisitos de *software*, **(iii)** *design* preliminar, **(iv)** *design* detalhado, **(v)** programação, **(vi)** testes e por fim a **(vii)** manutenção (Avison & Fitzgerald, 2003; Munassar & Govardhan, 2010).

Na definição dos **(i)** requisitos do sistema, é criada uma lista com as ferramentas necessárias para construir o sistema. São exemplo as decisões por um ou outro sistema de gestão de bases de dados (*Data Base Management System* - DBMS) e respetiva Base de Dados (BD). Na fase de definição de **(ii)** requisitos de *software* devem ser apresentadas as expectativas relativamente às funcionalidades da aplicação. Na conceção do **(iii)** *design* preliminar deverá ser estipulado qual o modelo

da aplicação necessário ao cumprimento dos requisitos estipulados nas duas primeiras fases. Na fase de **(iv)** *design* detalhado deve ser estipulada a forma como cada elemento, definido na fase de *design* preliminar, deverá ser implementado. Durante a **(v)** programação deverão ser programadas as especificações presentes no *design* detalhado e após a finalização da fase de programação deve proceder-se à integração de todos os módulos programáticos. Para garantir que a aplicação cumpre os requisitos estipulados, deve proceder-se a **(vi)** testes que ponham à prova cada um dos requisitos. Por fim, resta a fase de **(vii)** manutenção, na qual deverão ser abordados os problemas decorrentes da operação da aplicação e pedidos de melhorias da aplicação que tenham sido submetidos após o seu lançamento.

No modelo em cascata, a equipa de desenvolvimento só poderá passar à fase seguinte de desenvolvimento quando der por terminada a fase atual (Avison & Fitzgerald, 2003, p.79). Por fim, e para que uma fase de desenvolvimento seja finalizada, é feita uma revisão da mesma, averiguando-se se todas as tarefas foram concluídas e que a fase de desenvolvimento atingiu os objetivos estabelecidos, permitindo assim avaliar a progressão do projeto. Algumas vantagens e desvantagens desta metodologia estão enumeradas no **Anexo I.A**.

2.1.2. Metodologias Ágeis

O Desenvolvimento Ágil foi apresentado como uma *framework* de desenvolvimento em 2001, no Utah, no encontro que deu origem ao manifesto onde

estão presentes os doze princípios para um desenvolvimento ágil (Jeffries & Lindstrom, 2004). Esses princípios são: **(i)** satisfação do cliente através da entrega rápida e contínua de *software*, **(ii)** flexibilização da alteração de requisitos, **(iii)** períodos de entrega de algumas semanas a poucos meses, dando preferência a períodos mais curtos, **(iv)** garantir a colaboração entre o cliente e a equipa de desenvolvimento durante o decorrer do projeto, **(v)** desenvolver projetos com base em indivíduos motivados, **(vi)** a transmissão de informação cara a cara, **(vii)** a principal medida de progresso é a entrega de *software* funcional, **(viii)** os processos ágeis promovem um ritmo de desenvolvimento sustentável, **(ix)** a atenção permanente à excelência técnica e um bom desenho da solução aumentam a agilidade, **(x)** simplicidade é essencial, **(xi)** as melhores arquiteturas, requisitos e desenhos surgem de equipas auto-organizadas, **(xii)** a equipa reflete regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias (Beck, et al., 2001).

O desenvolvimento ágil consiste num grupo de metodologias que fazem uso de iterações ao longo do período de desenvolvimento. Nestes, a ideia principal é a minimização de risco, que é posta em prática através da entrega de protótipos operacionais no fim de cada ciclo (*sprint*) de desenvolvimento, habitualmente entre duas a quatro semanas, permitindo assim uma mais fácil avaliação do projeto ao longo do tempo. Entre as metodologias ágeis mais conhecidas serão apresentadas duas: *Scrum* e *Extreme Programming* (*XP*).

2.1.2.1. Scrum

O ciclo de vida da metodologia *Scrum* é composto por três fases: pré-jogo, desenvolvimento e pós-jogo (Abrahamsson et al., 2002). Está presente no **Anexo I.B** uma esquematização desta metodologia. Com base no que é exposto por Ramsin e Paige (2008), bem como noutros estudos (Abrahamsson et al., 2002; Guntamukkala et al., 2006) segue-se uma breve explicação do ciclo de vida da metodologia *Scrum*.

A fase **pré-jogo** tem como principal função preparar tudo o que seja necessário à iniciativa de desenvolvimento e está subdividida em duas subfases sobrepostas: planeamento e arquitetura. Na subfase de planeamento deve ser produzida uma lista (*backlog*), a qual é gerida pelo dono do produto, com os requisitos (funcionais e não funcionais) do sistema, riscos, consumo de recursos e previsões de tempo associados ao projeto. Com a ajuda do cliente, o *backlog* do produto deverá ser constantemente atualizado com os erros a corrigir e melhoramentos a realizar ao longo do projeto. Nesta fase é também necessária a formação de equipas de desenvolvimento, cada uma delas gerida por um *Scrum Master* e composta por cinco a dez elementos. Na subfase de arquitetura deverá ser identificada a estrutura do sistema. Caso seja necessário, o *backlog* do produto deve ser atualizado, tornando-o adequado à estrutura que foi desenhada (Ramsin & Paige, 2008; Guntamukkala et al., 2006).

Na fase de **desenvolvimento** (também conhecida como fase de jogo) procede-se ao desenvolvimento iterativo e incremental do sistema. No final de cada *sprint*, deve

ser lançado um novo executável do sistema, já com as novas funcionalidades implementadas. Esta fase é composta por três subfases: **(i)** planeamento do *sprint*, **(ii)** desenvolvimento e **(iii)** revisão do *sprint*. Durante o **(i)** planeamento, deve ser estabelecido um objetivo (*sprint goal*) para esse *sprint* (quantificada pelos elementos presentes no *backlog*). Após esta tarefa, é criada uma lista (*sprint backlog*) derivada do *backlog* do produto, onde são introduzidos os elementos que serão desenvolvidos no *sprint* em questão (Guntamukkala et al., 2006). Caso se conclua que já não são necessários mais *sprints*, então a fase de desenvolvimento é declarada como finalizada e dá-se início à fase pós-jogo (Ramsin & Paige, 2008). Durante o **(ii)** desenvolvimento de cada *item* do *sprint backlog* tomam lugar reuniões diárias em que todos os membros das equipas de desenvolvimento podem comparecer, evitando-se reuniões que excedam os quinze minutos de duração (Abrahamsson et al., 2002). Nesta reunião, todos os membros devem responder a três perguntas: 'O que fizeram ontem?', 'O que vão fazer hoje?' e 'Depararam-se com algum obstáculo?' (Ionel, 2008, p.438).Com as respostas, os S*crum masters* das equipas ajudam a remover os obstáculos encontrados pelas equipas e promover assim a eficiência e produtividade das mesmas dentro do projeto (Ramsin & Paige, 2008). Na reunião de **(iii)** revisão do *sprint* é feita uma demonstração do executável. Procede-se também, nesta reunião, à avaliação do progresso alcançado pelo *sprint,* averiguando-se se se cumpriu o objetivo estipulado e atualizando o *backlog* do produto em conformidade, assinalando quais os requisitos cumpridos, adicionar novos requisitos, erros encontrados ou melhoramentos necessários (Abrahamsson et al., 2002).

Por fim, na fase de pós-jogo deverá proceder-se ao teste da última versão integrada e criar documentação orientada para o cliente. De seguida deve introduzir-se o produto final no ambiente de trabalho do cliente. Por fim, restará apenas testar a aceitação do sistema por parte de todos os seus utilizadores (Abrahamsson et al., 2002). Algumas vantagens e desvantagens desta metodologia são enumeradas no **Anexo I.C**.

2.1.2.2. *Extreme Programming* (*XP*)

Uma outra metodologia que emprega princípios de desenvolvimento ágil é o *XP*, uma metodologia assente em quatro valores base para responder à necessidade de melhorar a qualidade do *software* e dar resposta a requisitos em constante mudança: comunicação, simplicidade, *feedback* e coragem (Jeffries & Lindstrom, 2004). Tal como o nome indica, se uma prática de desenvolvimento é considerada adequada, então a mesma deve ser levada ao extremo: se testar é bom, então que toda a gente teste, se refatorização de código corrige erros, então que se refatorize em todos os momentos, se a simplicidade é boa, então que se simplifique o máximo que não ponha em causa o funcionamento da aplicação, entre outras máximas (Beck et al., 2001). Segue-se uma explicação desta metodologia, apoiada no estudo presente em Ramsin e Paige (2008), não esquecendo outros estudos acerca desta metodologia (Jeffries & Lindstrom, 2004; Munassar & Govardhan, 2010; Guntamukkala et al., 2006; Abrahamsson et al., 2002).

A metodologia *XP* está dividida em seis fases: **(i)** exploração, **(ii)** planeamento, **(iii)** iterações até à primeira versão, **(iv)** implementação, **(v)** manutenção e **(vi)** morte. A sequência desta metodologia de desenvolvimento encontra-se exposta no **Anexo I.D**.

A fase de **(i)** exploração está subdividida em três atividades: formação da equipa de desenvolvimento, elaboração das primeiras *user stories* e a criação da metáfora do sistema. Durante a formação da equipa deve ser nomeado um 'treinador', que tem de monitorar e facilitar o trabalho da equipa de programadores, bem como garantir uma participação ativa do cliente ao longo do projeto. Na elaboração de *user stories* - descrição de uma funcionalidade que o sistema deva possuir, de acordo com o ponto de vista do cliente e escrita na terminologia do próprio – cada uma deve permitir fazer uma estimação do tempo necessário à sua integração e, conforme forem efetuadas alterações ou adições ao sistema as mesmas devem ser refletidas na lista de *user stories* atualizada. Por fim, resta a elaboração da metáfora do sistema, que consiste numa descrição sucinta e clara de como o sistema deverá funcionar, normalmente elaborada com recurso a analogias. A existência de um protótipo, pode facilitar esta tarefa (Ramsin & Paige, 2008).

Na fase de **(ii)** planeamento é efetuada uma estimativa do tempo de desenvolvimento, uma priorização das *user stories* e o agendamento do primeiro lançamento. Deve subdividir-se todas as *user stories* que excedam três semanas e juntar aquelas que se preveja que durem menos de uma semana. Com estas estimativas, o cliente deve priorizar as *user stories* de acordo com o seu valor para o negócio. Posteriormente, é agendada uma data para a finalização da primeira iteração

de desenvolvimento e a partir desta, estabelecer uma referência para as restantes iterações do projeto (Ramsin & Paige, 2008).

Terminado o planeamento, prossegue-se para a fase de **(iii)** iterações até a primeira versão, funcional do sistema. Esta fase é composta por um planeamento inicial da iteração seguido pelo seu desenvolvimento, repetindo-se estes processos ao longo do projeto. No início de cada iteração, a lista de *user stories* é atualizada de maneira a acomodar novos requisitos ou problemas detetados em iterações anteriores. Após a elaboração da lista os programadores deverão traduzi-las em tarefas que de seguida deverão ser escolhidas pelos mesmos. A cada iteração o conhecimento da velocidade da equipa deverá permitir elaborar uma lista adequada às capacidades de trabalho da mesma, suprimindo assim grande parte do risco inerente ao projeto. Quando se der por concluído o planeamento da iteração, poderá dar-se início ao desenvolvimento de cada uma das tarefas, num processo de repetição diária de certos procedimentos metodológicos (Ramsin & Paige, 2008).

Seguidamente, serão descritos os procedimentos metodológicos utilizados no desenvolvimento de cada tarefa. As **reuniões diárias** informais entre membros do projeto onde devem ser expostos problemas e soluções encontrados durante o dia anterior. A prática de uma **programação em ambiente de propriedade conjunta**, onde o código pertence à equipa e não a um programador específico, o que dá liberdade a qualquer programador de alterar o código conforme julgar necessário para o mesmo se tornar mais simples e compreensível, bem como a familiarização de toda a equipa com o código. Outra prática que deverá ser levada a cabo é a **integração contínua** do

código, que consiste em fazer uma junção entre o código recentemente criado e o código mais antigo de uma aplicação, tendo que primeiro passar por uma diversidade de testes criados a partir das *user stories* incluídas na iteração. Pretende-se com isto que mais tarde no projeto não se perca demasiado tempo a unir o código das diversas equipas. Ajuda também a que exista sempre uma versão atualizada e executável da aplicação. A adoção de um **desenvolvimento guiado por testes**, derivados das user stories elaboradas, que deverão garantir a fiabilidade do que é programado e dar confiança ao utilizador de que a aplicação funcionará corretamente quando acabada. A **simplificação do *design*** também deve ser tida em conta durante todo o esforço de desenvolvimento. A opção pelo mais simples é quase sempre melhor que o mais complexo. Será assim mais fácil testar código novo e refatorizar o mesmo. Já a **programação em pares**, como o nome o indica, significa que a codificação do sistema deve ser feita por equipas de duas pessoas, cada uma com o seu papel. Enquanto uma programa, a outra deve sugerir modificações ao código, invertendo os papéis decorrido algum tempo. Outra prática encorajada pelo *XP* é a utilização de ***coding standards*** que deverão impossibilitar a alguém da equipa dizer quem criou uma linha específica de código (comprimento de identações, funções a usar, nomenclatura de variáveis, etc) . Outra prática importante é a **refatorização de código**, em que mesmo que o código funcione, deverá ser alterado sempre que possível com vista a minimizar redundâncias programáticas. O propósito desta prática é o de tornar o código mais simples, facilitando a sua leitura e tolerância a mudanças futuras. Por último, o princípio da **semana de quarenta horas**. Se numa dada semana se observar a exceção

à regra de um ou outro programador ficar horas extra no local de trabalho, o mesmo não deve suceder na semana seguinte de forma a garantir que estão descansados (Munassar & Govardhan**,** 2010; Abrahamsson et al., 2002; Jeffries & Lindstrom, 2004).

Após a fase de desenvolvimento segue-se a **(iv)** implementação do sistema, após uma verificação e validação completa do mesmo, em que se procede à sua implementação no ambiente do cliente. A usual documentação e treino são tidas como indispensáveis, tal como em todas as fases de implementação (Ramsin & Paige, 2008).

A fase de **(v)** manutenção, a penúltima no processo *XP*, consiste na integração das *user stories* que não chegaram a ser implementadas durante as três fases anteriores. É em tudo semelhante aos processos das fases que a precederam, com a diferença de cada uma das novas funcionalidades ser integrada num sistema que já está operacional, sendo assim um caminho para uma versão mais completa do que se tinha estipulado no início do processo (Ramsin & Paige, 2008).

Por fim, chega a oficialização da **(vi)** morte do projeto, altura em que a evolução do sistema é considerada desnecessária ou impossível de realizar. Não dispensa as habituais resoluções finais de âmbito legal, financeiro e social (Abrahamsson et al., 2002). Algumas vantagens e desvantagens da metodologia *XP* são enumeradas no **Anexo I.E**.

2.2. Gestão de Dados

Nos primórdios do processamento de dados computorizado, os computadores eram utilizados exclusivamente para cálculos científicos e de engenharia. De forma a serem úteis ao ambiente de negócios, era necessário que os computadores pudessem armazenar, manipular e recolher grandes quantidades de dados. Foi para esse efeito que surgiram os primeiros sistemas computorizados de processamento de ficheiros (Hoffer et al., 2006). Este tipo de abordagem funcionava para esquemas de negócio simples, mas à medida que a complexidade das operações aumentava surgiam certas dificuldades com esta abordagem: a dependência entre programa e dados, a duplicação de dados, a limitação da partilha de ficheiros e longos períodos de desenvolvimento e manutenção (Hoffer et al., 2006). A **dependência entre progama e dados** ocorre quando um dos ficheiros que é acedido por diversas aplicações é alterado, sendo assim necessária a alteração dessas aplicações para efetivar corretamente as alterações. É frequente a ocorrência de erros durante estas mudanças. A **duplicação de dados** é outra ocorrência frequente neste tipo de aplicações, visto que sendo desenvolvidas de forma independente, irão existir situações em que o mesmo dado será utilizado por aplicações diferentes. No entanto, cada aplicação terá o seu ficheiro com a repetição desse dado, visto não existir ligação entre as mesmas. A **limitação da partilha** de ficheiros entre utilizadores devido às políticas de privacidade de ficheiros de cada aplicação. A necessidade de cada aplicação ser desenvolvida desde o zero conduz, por seu lado, a **longos períodos de**

desenvolvimento. No seu conjunto, os fatores anteriores levam também a que seja necessária uma **extensa manutenção** deste tipo de aplicações, conduzindo muitas vezes a que "quase 80% do orçamento total para o desenvolvimento de sistemas de informação seja dedicado a manutenção" (Hoffer, et al., 2006, p.12).

Para solucionar estes problemas surgiram as aplicações apoiadas em BD's. Segundo Navathe e Elmasri (2002, p.24), uma abordagem apoiada em BD consiste no uso de uma "coleção de dados relacionados", dados esses que são "factos conhecidos que podem ser guardados e que possuem um significado implícito" (definição de uma BD). Os autores explicam também que a criação e manutenção da BD pode ser feita manualmente ou então com a ajuda de um sistema de gestão de base de dados (*Data Base Management System* - *DBMS*). Um DBMS é um "sistema de *software* que facilita os processos de definição, construção e manipulação das BD's para várias aplicações" (Navathe & Elmasri, 2002, p.25). A conjugação de uma BD e um *DBMS* origina um sistema de base de dados (Navathe & Elmasri, 2002). A sua integração no desenvolvimento de aplicações traz diversos benefícios: independência entre programa e dados, duplicação/redundância de dados planeada, fácil partilha de dados, maior produtividade durante o desenvolvimento de aplicações, aplicação de regras na escolha de nomes e convenções de acesso, manipulação e proteção dos dados, tempos de resposta de consulta de dados mais curtos, manutenção reduzida e por último, uma melhor qualidade de dados (Hoffer et al., 2006). Foi a este último ponto dada grande importância no projeto e como tal será abordado a seguir.

"A qualidade dos dados produz sérias consequências, de enorme significância, para a eficiência e eficácia de organizações e negócios" (Batini & Scannapieca, 2006, p.2). Um exemplo é o impacto negativo que uma marca tem nos seus clientes caso um erro na sua BD origine o envio de *newsletters* múltiplas vezes aos mesmos clientes. De forma a evitar este e outros erros é necessário dar atenção às diversas dimensões de qualidade dos dados – *accuracy*, *completeness*, *currency*, *timeliness*, *volatility*, *consistency*, *accessibility* e *quality of information sources*[1]. Estas dimensões são apresentadas de seguida de acordo com a explicação presente em (Batini & Scannapieca, 2006). Para efeito de exemplos irá tomar-se **A'** como a representação em dados de uma realidade **A**.

Começando por ***accuracy*** terão de ser abordados os dois tipos que compõem esta dimensão: sintática e semântica. O tipo sintático avalia se um dado está dentro do domínio de valores presentes em A (**A' ∈ ʊ**). Por exemplo, caso **A'** = C# quando **A** = VB.NET, **A'** estaria sintaticamente correto visto representar uma linguagem de programação de servidor. Quando se fala no aspeto semântico, **A'** não estaria correto visto não ter sido essa a linguagem de programação utilizada no projeto que realizei. Na avaliação semântica, apenas uma qualificação booleana faz sentido (correto ou incorreto).

A segunda dimensão, ***completeness***, diz respeito à presença de valores nulos em campos da base de dados. Desde que representem fielmente a realidade, o uso de

[1] De forma a não originar erros de tradução mantiveram-se os nomes em Inglês

valores nulos num registo leva a que esse mesmo registo seja completo. Caso esta representação da realidade não seja a correta, quando **A'** = *NULL* e **A** = 6789, o registo estará incompleto.

Relativamente às dimensões temporais – ***Currency***, ***Volatility*** e ***Timeliness*** – poderá dizer-se que as mesmas estão intimamente ligadas à evolução dos dados de uma BD ao longo do tempo. ***Currency***, por exemplo, diz respeito à rapidez com que os dados são atualizados na BD quando os mesmos sofrem mudanças em **A**. Imagine-se o caso de uma pessoa que mudou de residência e pretenda alterar a sua residência no *website* de uma empresa que lhe envia semanalmente uma *newsletter*. Caso a alteração tenha sido efetuada antes do próximo envio, o grau de *currency* é normal/elevado, em oposto com um grau reduzido se essa mesma alteração não tiver sido efetuada a tempo. ***Volatility*** aborda a validade dos dados que variam, com alguma frequência no tempo, e se essa mesma validade é transposta para a BD. Quanto maior o período em que um registo se mantém válido, melhor. Deverá ser preocupação dos programadores de uma aplicação apoiada por uma BD garantir que os dados utilizados são válidos. A última dimensão temporal, ***Timeliness***, avalia a atualidade dos dados do ponto de vista do utilizador. Não só precisam ser actuais como disponibilizados atempadamente ao utilizador. Por exemplo, imagine-se que um dado curso só é introduzido na lista de cursos disponibilizados pela faculdade no último dia de matrículas. Apesar de ser bastante atual, esta lista foi atualizada demasiado tarde para influenciar certos alunos a escolher um dado curso ou optar por

outra faculdade que o oferecesse. Caso a lista fosse atualizada a tempo de influenciar a maioria dos candidatos a matricularem-se no referido curso poderia afirmar-se que o grau de *timeliness* dos dados era adequado.

A dimensão de ***Consistency***, debruça-se sobre a validade relacional de um dado registo na BD. Esta dimensão é influenciada pela estruturação de restrições (*Integrity Constraints*) dentro da BD. Caso não existam restrições de integridade, poderá existir um registo na tabela 'Motas' com um campo #Rodas = 6. Tal seria uma óbvia má representação da realidade, que seria impossível de acontecer caso uma restrição fosse imposta no campo #Rodas. Uma simples forma de o fazer seria através do comando `ALTER TABLE #Rodas ADD CONSTRAINT chk_#Rodas CHECK (#Rodas IN ('2', '3', '4'))`. Outro tipo de restrição é a ligação entre campos de diferentes tabelas, as chamadas chaves (primárias e estrangeiras), que limitam os valores de uma tabela aos já existentes noutra tabela.

A dimensão que mede o grau de ***acessibility*** tem como função "medir a capacidade de o utilizador aceder aos dados independentemente da sua cultura, estado físico/incapacidades e tecnologias disponíveis" (Batini & Scannapieca, 2006, p.34).

Por último, e em especial em ambientes *web*, é preciso ter em conta a ***quality of information sources***, que avalia se "uma dada fonte de informação oferece dados verdadeiros, reais e credíveis" (Batini & Scannapieca, 2006, p.35).

2.3. Ambiente Web

No projeto, que é apresentado no capítulo seguinte, o principal objetivo era a criação de uma aplicação *web* com vista a melhorar os processos de trabalho na fábrica. Esta aplicação deveria ser acedida através de um *browser*, tanto num *desktop* como num *tablet*. De entre os benefícios deste tipo de aplicação contam-se dois que foram também identificados por Froehlich et al. (1999, p.457) no seu caso de estudo: "devido a não necessitar de instalação, um interface de utilizador *web* elimina as dores de cabeça inerentes a instalações e atualizações do *software*, bem como permitir o acesso a partir de qualquer lugar em qualquer dispositivo" e o facto de a aplicação possuir o seu próprio caminho para a resolução de problemas, o tempo associado a essa tarefa é relativamente diminuído. Mas antes de elaborar mais sobre os benefícios de uma aplicação *web* é necessário referenciar certos termos e conceitos, essenciais no âmbito do projeto de que fiz parte. De forma a não dispersar desnecessariamente irei abordar sucintamente os conceitos de *e-commerce* e *m-commerce*, quais os tipos de portais que existem e um *benchmarking* de construção de *websites*.

2.3.1. *E-commerce* e *m-commerce*

Segundo o pensamento seguido por Mcconnell e Stephen (1997, p.10), *e-commerce* é a "aplicação de tecnologias de informação, especificamente a inteligência e conectividade dos computadores, na resolução dos problemas do comércio", sendo esses mesmos problemas as "barreiras à circulação da informação que compõe o

comércio". Já Froehlich et al. (1999, p.458) definem *e-commerce como* "a entrega de processos de negócio e aplicações através da *web*". Sendo assim, poderá aferir-se que *e-commerce* é a aplicação de tecnologias de informação num negócio através da facilitação da circulação de informação através da *Internet*. No caso específico da aplicação que desenvolvi, *IntraBusiness* será a definição mais adequada, visto aplicar-se a "todas as atividades organizacionais que envolvam a troca de produtos, serviços ou informação entre as várias unidades e indivíduos numa organização" (Picoto, 2012, Slide 11).

Sendo verdade que "parece existir entendimento alargado de que *m-commerce* é o uso de dispositivos móveis para comunicar e levar a cabo transações" (Balasubramanian et al., 2002, p.349), não existe uma definição formal do termo. Balasubramanian et al.(2002) conceptualizam *m-commerce* como um "fenómeno" em que exista **(i)** comunicação unidirecional ou interativa entre pessoas e objetos inanimados, **(ii)** uma das partes esteja conectada através de um dispositivo móvel, **(iii)** a capacidade de manter a comunicação mesmo que uma das partes se desloque de um ponto geográfico para outro. **(iv)** Se a comunicação é feita entre pessoas então, pelo menos uma delas terá de estar a tentar alcançar algum proveito económico dessa mesma comunicação. Seguindo Clarke III (2001, p.41), *m-commerce* deve ser definido como "a capacidade de comprar bens em qualquer lugar a partir de um dispositivo *wireless* conectado à *Internet*". Já Smith (2006, p.682) define *m-commerce* como a "compra e venda de bens e serviços através de dispositivos móveis *wireless*, tais como

telemóveis e PDA's.". Sendo assim, entende-se por *m-commerce* a possibilidade, de em qualquer lugar e a qualquer hora, uma pessoa/objeto trocar ou comprar bens e/ou serviços com/a outra(s) pessoa(s)/objeto(s). Estando incluída na categoria de serviços, a visualização de dados empresariais.

2.3.2. Portais *Web*

A aplicação desenvolvida, no seu todo, toma a forma de um portal *web* com acesso restrito à *Intranet* da organização, onde é possível reunir informações, tanto da BD técnica da fábrica (PIMS) como de uma base de dados *SQL SERVER* 2008 (de ora em diante denominada por Vega), sendo a última para uso exclusivo da aplicação. Existem diversos tipos de portal no *e-commerce*, sendo os mais importantes os portais **(i)** comerciais, **(ii)** publicitários, **(iii)** pessoais, **(iv)** móveis, onde é possível visualizar a informação de um dado portal num formato compatível com dispositivos móveis, e os **(v)**corporativos, categoria em que se insere o portal que se construiu ao longo do projeto (Turban et al., 2010). Os portais corporativos possuem duas vertentes: informacional e colaborativos. Nos portais informacionais o objetivo é possibilitar a navegação e consulta de todos os dados de teor informativo da empresa. Já os colaborativos dão apoio ao trabalho interno de uma organização, possibilitando a partilha de informação entre fornecedores, gestores e funcionários de uma organização. Este tipo de portais, sendo que usam as redes internas de comunicações, deverão fazer uso dos grupos de utilizadores já existentes na *Intranet* da organização, garantindo que os utilizadores possam utilizar um sistema em que o *login* que usam

para iniciar sessão no seu computador seja o mesmo que é utilizado pela aplicação ao verificar a veracidade desse mesmo *login*. É assim mais fácil garantir que o utilizador se lembra da *password* e torna também desnecessária a construção de um sistema de *login* completo para a aplicação. É também uma das principais funcionalidades deste tipo de portal, e que foi aplicado na aplicação desenvolvida. Um sistema em que os níveis de acesso de cada utilizador apenas lhe permitem consultar ou editar os dados que à sua função dizem respeito, transpondo assim a hierarquia organizacional, de forma eficiente, para o ambiente virtual. Estas características influenciam a forma como uma organização opera e transmite a informação entre os diversos níveis hierárquicos.

Segundo Fan e Tsai (2010, p.1144), *web design* diz respeito ao "ambiente e canal de interface através do qual o utilizador e um computador trocam informação, para que o primeiro possa visualizar, procurar e introduzir dados" no sistema. Tendo este aspeto bem presente, é importante que o *design* seja adequado às necessidades do utilizador, pois caso não o seja, será difícil para o mesmo consultar, modificar ou introduzir os dados que pretende, podendo tornar-se numa experiência mais maçadora para o utilizador.

Existem várias formas de avaliar o *design* de um *website*, começando pelo seu sistema de navegação e a quantidade de informação disponibilizada (Sumpton, 2013) ou o rácio de sucesso, número de cliques e tempo necessários ao utilizador para completar uma dada tarefa. Métodos que foram utilizados como parte da avaliação da

navegabilidade de *websites* no caso de estudo elaborado por Fang et al. (2012). Foi no entanto identificada por Cebi (2013) alguma dispersão nos métodos de avaliação utilizados por diversos autores. No seu estudo, elaborou uma compilação das mesmas, presente no **Anexo I.F**. Para além destes fatores, é preciso ter sempre em conta que tipo de utilizadores irão utilizar o *website* e que dificuldades cada tipo de utilizador poderá ter ao utilizar o mesmo. Jans et al. (2013) defendem, assumindo ao mesmo tempo as limitações do estudo que efetuaram, que desde que não existam grandes distrações no centro de uma página (gráficos, texto extensivo, etc), os utilizadores mais velhos não apresentam grande dificuldade na identificação inicial e utilização de partes cruciais de uma página localizadas nas suas periferias (Ex: Canto superior esquerdo **Anexos II.B, D, E, F** e **G**).

3. PROJETO

O projeto objeto deste trabalho foi desenvolvido na *Solvay*, empresa do setor químico, que emprega presentemente 200 pessoas na sua fábrica da Póvoa de Santa Iria. Nestas instalações, os engenheiros e operários monitorizam as operações através de sistemas informáticos avançados da empresa *Aspen Tech*. Contudo, os trabalhadores não dispensam consultas personalizadas dos dados mais críticos para os processos de fabricação. Para o efeito, foram criadas pequenas aplicações, algumas em ambiente *web,* pelo departamento de sistemas de informação (*Solvay Information Services* - SIS) e outras com recurso a macros VBA em *Excel*. Estas aplicações em *Excel*

assemelham-se ao que está presente no **Anexo II.E**. Com vista a melhorar a consulta de dados na fábrica, a gestão da organização decidiu dar início ao projeto 'Automatização do Reporting Fabril', do qual fiz parte e descrevo nas secções seguintes.

3.1. Método de Desenvolvimento

No início do projeto não foi tomada uma decisão formal sobre qual a metodologia de desenvolvimento a adotar no mesmo. No entanto, foi rejeitada desde início a hipótese de se adotarem processos de desenvolvimento similares aos de um desenvolvimento em cascata. Devido a experiências passadas com o SIS, os clientes consideravam esta metodologia como arcaica e utópica. Ao longo do projeto foram sendo adotados processos característicos de metodologias ágeis, sendo que após um período de adaptação os mesmos se tornaram prática corrente.

Na fase de **(i) exploração** do projeto formou-se uma equipa de três elementos, um programador (eu próprio), o chefe de uma unidade produtiva e um gestor de otimização de processos industriais. Durante as primeiras duas semanas estudou-se o ambiente de visualização de dados (*Reporting*) na empresa. A mesma tinha duas vertentes: a utilização de *software Aspen Tech* (PIMS) para consultas aprofundadas e aplicações *ad-hoc* (*Macros Excel*) para consultas menos exigentes, em termos de volume de dados. Definiu-se também a metáfora para o sistema: uma aplicação, independente do dispositivo utilizado (à exceção de *smartphones*), que permitisse a

gestão e visualização de dados de forma centralizada e fácil de manusear pelos utilizadores. Deveria também não ser necessário qualquer envolvimento por parte do departamento SIS no que respeitasse a implementações futuras e manutenção da aplicação. Tendo em consideração esta metáfora, as funcionalidades que foram definidas de início e o estudo do ambiente de *reporting* chegou-se à conclusão que a melhor opção seria a criação de uma aplicação *web*, acedida através de um *browser*, apoiada pela base de dados Vega.

Durante o **(ii) planeamento**, a priorização e previsão temporal das *user stories* foi feita informalmente pelos clientes. A lista criada foi sendo cumprida ao longo do projeto com exceção de dois *sprints* onde se discutiu e optou pelo prolongamento dos mesmos de forma a não prejudicar o produto final. Os ciclos de desenvolvimento iniciaram-se sempre pela tradução da *user story* em pseudo-código ou tarefas. Darei o exemplo da consulta de folhas de marcha (**Anexo II.E**). Numa primeira fase, deve ser recolhido o nome da folha de marcha e data. De seguida dar início a recolha de informação estrutural (pontos de medida, limites, descrições) em Vega e por último recolher os valores de cada período no PIMS.

Na fase de **(iii) desenvolvimento**, foram sendo discutidas com os clientes durante as reuniões diárias de cada ciclo, customizações à medida que cada tarefa era programada. Sempre que cada tarefa fosse sendo programada a mesma era testada, por todos os membros da equipa, de forma a garantir a sua correta execução e máxima simplificação. Era com isto objetivo do projeto garantir uma progressiva

melhoria nos tempos de resposta da aplicação. Sendo que se trata de uma aplicação *web*, a integração do código foi efetuada tanto em funcionalidades ainda em desenvolvimento como em funcionalidades já em ambiente de produção. O mesmo se aplicou na refatorização de código, na medida em que cada vez que se programou código mais eficaz para uma dada operação – mesmo quando a mesma já tinha sido utilizada em ciclos anteriores – esse mesmo código foi implementado no ciclo atual e em código já em produção. Tanto na implementação contínua como na refatorização de código já em produção procedeu-se com cuidado na forma como essas alterações foram efetuadas. Só assim foi possível não perturbar a experiência dos utilizadores no usufruto das funcionalidades já implementadas.

A utilização destas técnicas proporcionou à equipa reduzir o tempo de manutenção da aplicação. Essa manutenção diz respeito principalmente aos objetos da aplicação (**Anexos II. D** e **H**), armazenados em Vega e tal como foi definido pela metáfora do sistema, essa manutenção é efetuada pelos utilizadores através da aplicação criada. Sendo que o desenvolvimento ainda não terminou, a 'morte' do projeto ainda se encontra em espera, estando previstas outras funcionalidades, como a criação de diários de fabricação (presentemente em desenvolvimento), introdução e visualização de consumos energéticos afetos às operações de fabricação, ligações ao ERP (*Enterprise Resource Planning*) para introdução de dados de produção e por último, apresentações de resultados financeiros.

Os diários de fabricação consistem em formulários impressos em papel, preenchidos pelos operários da empresa durante cada turno (três por dia). Nestes diários são preenchidos campos referentes a alguns pontos de medida que existem no PIMS, outros que não existem de todo, e observações de texto livre referentes a eventos importantes ocorridos durante o turno. A funcionalidade, presentemente em desenvolvimento, permitirá introduzir estes dados em Vega, permitindo consultar o diário de um dado turno com maior facilidade. Será também possível realizar pesquisas avançadas que no formato em papel se tornam tarefas de horas ou até impossíveis. Relativamente à apresentação dos dados financeiros, será feita uma ponte entre a aplicação desenvolvida e uma outra aplicação *web*, externa, que permite criar e visualizar relatórios financeiros através de um *browser*.

3.2. Gestão de Dados

A política utilizada tanto na esquematização de Vega como na programação da aplicação foi sempre pautada pela preocupação com a qualidade dos dados. Foi com essa preocupação em mente que se criaram contingências para mitigar o risco de não cumprimento dos *standards* identificados na revisão de literatura sobre gestão de dados. São exemplo destas contingências as restrições de integridade criadas em Vega. Passo a explicar o exemplo da criação de um campo booleano: atribuindo ao número **0** o valor de 'falso' e ao número **1** 'verdadeiro'. Sendo que a programação da aplicação não permite a adição de outros valores senão estes, optou-se por proteger Vega contra futuras alterações no código que possibilitassem o envio de outros valores. Tal

foi feito criando uma restrição em Vega para o campo em questão, que limita os valores aceites a 0 e 1. Outra prática foi a utilização de transações e tentativas (*Transactions* e *TRY*'s respetivamente) de forma a garantir que, quando uma *query* de inserção de dados em múltiplas tabelas é enviada para o servidor, apenas quando todos os *INSERT*'s são bem sucedidos é que os mesmos serão executados. Tal é feito através de um bloco similar a

```
BEGIN TRY BEGIN TRANSACTION INSERT INTO tabela 1(...) VALUES(...) INSERT INTO tabela 2(...) VALUES(...) COMMIT END TRY BEGIN CATCH ROLLBACK PRINT 'Mensagem de erro personalizada/Código do erro' END CATCH.
```

Com o uso de comandos como o anterior e de restrições em tabelas consegue-se garantir que as medidas de ***Consistency***, ***Completeness*** e ***Accuracy* sintática** tenham coeficientes elevados. A possibilidade de os utilizadores inserirem novos valores e editar os já presentes em Vega garantem que os coeficientes de ***Accuracy* semântica**, ***Currency***, ***Timeliness*** e gestão de ***Volatility*** se aproximem de valores desejados, à medida que o uso da aplicação vai aumentando. A segurança na ***Accessibility*** dos dados é garantida pelos níveis de segurança presentes em Vega e no tratamento aplicado aos mesmos pela aplicação. Por último, a ***Quality of Information Sources*** é o único campo que foge ao controlo do trabalho realizado pela equipa de projeto visto a fonte de informação ser uma BD paralela, PIMS. No entanto, caso os dados do PIMS sejam verificados, considerados fiáveis e utilizados nas operações diárias da fábrica, não há razão para colocar em questão a fiabilidade dos mesmos no âmbito do projeto.

3.3. Ambiente *Web*

O *design* da aplicação foi construído de forma a melhor servir os interesses dos utilizadores, tendo em conta as métricas presentes no **Anexo I.F**.

Começando pelo **menu de navegação** (**Anexo II.A**), manteve-se a mesma estrutura de navegação que já existia, e era utilizada diariamente, na consulta de ficheiros de fabricação. Apenas a secção de administração não figura nessas pastas, sendo no entanto auto-explicativa quanto ao seu propósito. Através deste menu, os utilizadores não são restringidos a uma navegação fixa entre páginas - visto o menu ser carregado no início de cada página é possível alterar um ficheiro para as mudanças serem visíveis em toda a aplicação. Podem desta maneira passar diretamente da área de administração para a de fabricação e observar as alterações que porventura tenham efetuado. Outro fator importante é o de cada página possuir o seu *header* específico, permitindo ao utilizador saber sempre em que zona do *site* está (**Anexo II**).

Relativamente às divisórias de **execução de comandos** - presentes nos cantos superiores esquerdos dos **Anexos II. B**, **D**, **E**, **F** e **G** - as mesmas foram aí colocadas para tornar mais fácil a ambientação de novos utilizadores ao portal – habituados aos ficheiros de *Excel* com menus de execução similarmente posicionados. Com o mesmo princípio em mente, foram criados os botões de seleção de *Tags* individuais (presentes nos **Anexos II. B**, **D**, **G** e **H**), fazendo a replicação de uma seleção em árvore de *Tags* individuais presente no *Tag Browser* do PIMS – *software Aspen Tech*. Outro aspeto

importante para garantir a coerência numa aplicação *web* é a **coloração** dos objetos de cada página. O **Anexo II** apresenta o modelo criado. Aos botões de execução inicial, manipulação temporal de gráficos e navegação temporal de folhas de marcha foi atribuída a cor azul. Para botões de exportação, navegação temporal de gráficos e adição ou remoção de elementos em Vega optou-se pelo verde e vermelho. Por fim, botões que tenham como funcionalidade alterar o estado de um objeto (**Anexos II. C** e **F**) entre ligado ou desligado terão uma cor laranja. Este procedimento tem como objetivo que futuras funcionalidades sejam melhor compreendidas pelos utilizadores, que poderão associar a uma dada cor uma operação similar à de botões de páginas aos quais os utilizadores já se encontram familiarizados. A melhoria da comunicação entre os utilizadores e a equipa do projeto foi possível com a programação de *links* dinâmicos que fazem uma ligação ao *GMAIL* corporativo. A estes *links*, é associada uma mensagem pré-definida - para reportar erros e criação de registos de utilizador em Vega - e com um destinatário, também ele já definido. É assim comunicado ao utilizador quem o poderá ajudar em situações futuras, ao mesmo tempo que a equipa de projeto reduz o tempo de correção dos erros reportados. Outro aspeto que foi chamado à atenção por ambos os clientes, dizia respeito ao **número de cliques** necessários para se chegar à página pretendida. Com o menu da aplicação, basta colocar o rato sobre a área que pretende abrir até chegar à página que pretende e, por último, clicar na mesma (**Anexo II.A**). Depois, na maioria dos casos, bastará escolher a data do pedido (se necessário) e o elemento que pretende executar. Era considerado importante, limitar sempre que possível, o número de cliques necessários para chegar

a uma dada página (regra de três cliques). Por outro lado, e graças à programação da aplicação, conseguiram-se também alcançar algumas economias de tempo relativamente ao tempo de resposta. Olhando para o **Anexo II.K** é possível concluir que apenas as funcionalidades que manipulam um maior volume de dados (**Anexos II. E** e **F**) excedem um tempo médio de execução de dois segundos. Isto garante que os tempos que levam um utilizador a concluir uma dada tarefa sejam também eles reduzidos.

Outro aspeto que foi tido em conta ao longo do desenvolvimento da aplicação foi a articulação entre objetos de visualização (tabelas, gráficos, botões, etc). Exemplo disso é a ligação permanente entre dados da tabela e gráfico do **Anexo II.F**, que ajuda visualmente o utilizador a localizar as *Tags* que adicionou ou não ao gráfico, sabendo sempre o que está a visualizar. Essa mesma ajuda visual ocorre com as imagens presentes no mesmo anexo e no **Anexo II.E** - coloração das células da folha de marcha - na identificação de valores fora do normal. Acrescendo a estas ajudas o facto de os dados visuais poderem ser exportados para ficheiros *Excel* e PDF é possível: melhorar o grau de partilha de informação dentro da empresa e garantir que nesses ficheiros também estão presentes as ajudas visuais à identificação de situações anormais.

Por último, e como normalmente acontece, as caixas de diálogo são uma forma de dar a conhecer ao utilizador as mensagens de erro quando um pedido falhou, avisar que a sessão de *login* expirou ou que os seus pedidos e consequentes operações tenham sido bem sucedidos. É muito importante que um utilizador saiba quais os

resultados que as suas interações com o interface têm. Esta particularidade aumenta a curva de aprendizagem de cada utilizador evitando que, à medida que o tempo avança, ocorram repetições de erros.

3.4. Impactos

Após a finalização do projeto, estava à disposição dos utilizadores a visualização e manipulação de dados relativos ao dia-a-dia da fábrica, de análises laboratoriais, de produções diárias e semanais de produtos finais, de pontos de medida individuais, da gestão de dados visualizáveis (**Anexos II. D** e **H**), da gestão de utilizadores (criação, edição de níveis de segurança e departamentos, etc) e introdução e cálculo de consumos de energia nas propriedades da empresa. Graças à aplicação, criaram-se economias de tempo nos tempos de consulta e gestão da informação visto já não ser necessária a gestão e procura de diversos ficheiros bem como, os tempos de resposta na consulta de dados serem muito mais curtos. A forma como a utilização de cores é feita permite também uma maior rapidez na identificação de situações anormais que antes não era, de todo, feita. A exportação destes dados operacionais permitiu também grandes ganhos de tempo no que diz respeito à preparação de reuniões dos órgãos de gestão da fábrica e também em cálculos com recurso a dados consultados. A exportação de dados permitiu também o melhoramento na gestão e partilha da informação, algo que era bastante difícil de executar num ambiente de gestão de ficheiros individuais. No início do projeto era também pouco prático visualizar dados a partir de um local fora das instalações, o que agora é possível através dos gráficos e

tabelas criados em ambiente *web*. Foi assim facilitada a gestão *off-site* de situações anormais, não sendo necessário no futuro ter alguém nas instalações a reportar um problema a outra pessoa que não esteja presente. Agora basta indicar a fonte de dados a consultar a um elemento que esteja no exterior, para este perceber como estão a decorrer as operações fabris e identificar possíveis causas de situações anómalas.

No seu todo, a aplicação garantiu uma migração bem sucedida, de um ambiente em que a consulta de dados era feita através de ficheiros *Excel* para um ambiente *web*, suportado por uma BD (Vega).

4. Conclusões

No decurso do projeto, deparei-me com algumas dificuldades na aplicação dos conhecimentos obtidos ao longo do mestrado no trabalho prático. Na sua grande maioria, se não na totalidade, essas dificuldades surgiram devido a disputas organizacionais com o departamento SIS.

Visto não ter sido formalmente escolhida uma metodologia de desenvolvimento para o projeto, existiu um período inicial de aprendizagem em que o projeto passou por uma fase de *code-and-fix* antes de se adotarem métodos mais ágeis de desenvolvimento. No entanto, este facto não influenciou o período de desenvolvimento do projeto, uma vez que esse período de aprendizagem sobrepôs-se

a um período de conflito com o departamento SIS. Este conflito (razão pela qual o período de desenvolvimento do projeto não foi afetado pelo período de aprendizagem), criou alguns obstáculos organizacionais à equipa do projeto se deparou, alguns destes enumerados a seguir.

Ao longo dos anos, o SIS foi escolhido para criar aplicações para a introdução e visualização de dados no PIMS, entre outras aplicações. Apesar de sempre cumprirem com a entrega destas mesmas aplicações, os gestores da fábrica optaram por uma mudança na forma como se deveria proceder em projetos de desenvolvimento de *software*. A experimentação iniciou-se com o projeto em que estive envolvido. O que resultou desta mudança foi um debate sobre se pessoas externas ao SIS deveriam ter a possibilidade de desenvolver aplicações de uso operacional. Durante este debate, o SIS nem sempre deu o apoio necessário à equipa do projeto - leia-se por apoio a instalação de um servidor virtual e uma base de dados (que mais tarde veio a ser Vega) - para que esta conseguisse iniciar o seu trabalho. Para se perceber o real alcance deste obstáculo, todo o código foi escrito, e continua a ser escrito, através de *notepad* visto que fora do SIS "não é permitido o uso de ferramentas de desenvolvimento de *software*" (seria apenas necessária a instalação de uma versão *Express* do *Visual Studio 2012*).

Fora estes dois de obstáculos, a aplicação é considerada uma boa mudança pelos seus utilizadores e um sucesso crescente pela equipa de gestão do projeto. Poupança de tempo e simplicidade de uso são tidos como os principais benefícios alcançados.

Estas razões fazem da aplicação uma ótima candidata a implementação noutras fábricas espalhadas pelo globo, hipótese que está já a ser debatida.

No futuro, outras funcionalidades serão implementadas, permitindo consultar dados fabris e financeiros da fábrica, fazendo assim a ligação entre o plano operacional e o de resultados económicos, tudo num mesmo local virtual. Será assim possível alcançar uma ainda maior centralização da comunicação de dados, dentro da realidade *Solvay*.

5. Bibliografia

- Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J., 2002. Agile Software Development Methods: Review and Analysis. *VTT Publications,* pp. 1-99.
- Balasubramanian, S., Peterson, R. A. & Jarvenpaa, S. L., 2002. Exploring the Implications of M-Commerce for Markets and Markets. *Journal of the Academy of Marketing Science,* pp. 348-361.
- Batini, C. & Scannapieca, M., 2006. *Data Quality - Concepts, Methodologies and Techniques.* s.l.:Springer.
- Beck, K. et al., 2001. *Principles behind the Agile Manifesto.* [Online] Disponível em: http://agilemanifesto.org/principles.html [Acedido a 11 de Setembro de 2013].
- Cebi, S., 2013. Determining importance degrees of website design parameters based on interactions and types of websites. *Decision Support Systems,* pp. 1030-1043.
- Clarke III, I., 2001. Emerging Value Propositions for M-commerce. *Journal of Business Strategies,* pp. 133-148.
- Fang, X. et al., 2012. A Data-Driven Approach to Measure Web Site Navigability. *Journal of Management Information Systems,* pp. 173-212.
- Fan, W.-S. & Tsai, M.-C., 2010. Factors driving website success - the key role of Internet customisation and the influence of website design quality and Internet marketing strategy. *Total Quality Management & Business Excellence,* pp. 1141-1159.
- Fitzgerald, G. & Avison, D. E., 2003. Where Now for Development Methodologies?. *Communications of the ACM,* pp. 78-82.
- Froehlich, G., Hoover, H. J., Liew, W. & Sorenson, P. G., 1999. Application Framework Issues When Evolving Business Applications for Electronic Commerce. *Information Systems,* pp. 457-473.
- Guntamukkala, V., Wen, H. J. & Tarn, M., 2006. An Empirical Study of Selecting Software Development Life Cycle Models. *Human Systems Management,* p. 265–278.
- Hoffer, J. A., Prescott, M. B. & McFadden, F. R., 2006. *Modern Database Management.* s.l.:Pearson Prentice Hall.
- Jans, M. E., Olmsted-Hawala, E. & Bergstrom, J. R., 2013. Age-Related Differences in Eye Tracking and Usability Performance: Website Usability for Older Adults. *International Journal of Human-Computer Interaction,* pp. 541-548.
- Jeffries, R. & Lindstrom, L., 2004. Extreme Programming And Agile Software Development Methodologies. *Information Systems Management,* pp. 41-52.
- Mcconnell, Stephen, 1997. *The OGM/CommerceNet Joint Electronic Commerce Whitepaper,* s.l.: Object Management Group.
- Munassar, N. & Govardhan, A., 2010. A Comparison Between Five Models of Software Engineering. *International Journal of Computer Science Issues,* pp. 94-101.

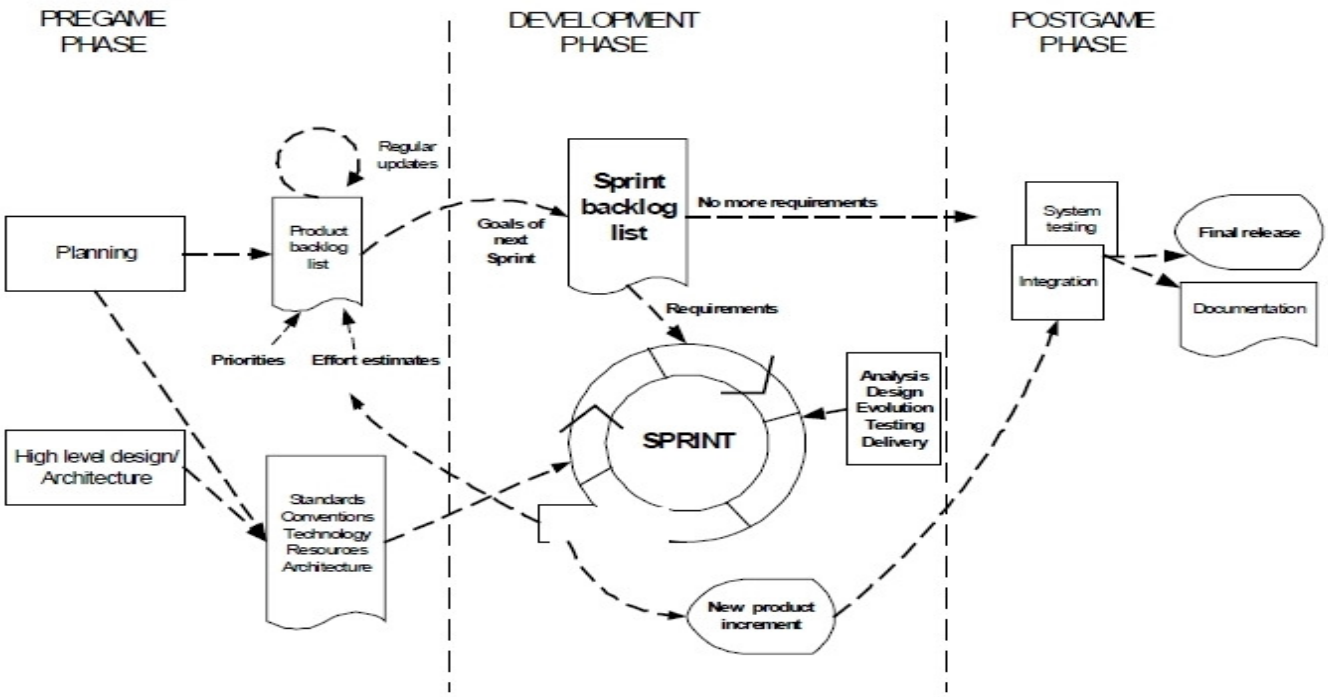
- Navathe, S. B. & Elmasri, R., 2002. *Fundamentals of Database Systems.* s.l.:Addison Wesley Longman.
- Picoto, W., 2012. *Fev. 2012.* Lisboa: Instituto Superior de Economia e Gestão, 47 Slides, color.
- Ramsin, R. & Paige, R. F., 2008. Process-Centered Review of Object Oriented Software Development Methodologies. *ACM Computing Surveys,* pp. 1-89.
- Sheffield, J. & Lemétayer, J., 2013. Factors Associated with the Software Development Agility of Successful Projects. *International Journal of Project Management,* pp. 459-472.
- Smith, A. D., 2006. Exploring M-commerce in Terms of Viability, Growth and Challenges. *International Journal of Mobile Communications,* pp. 682-703.
- Sumpton, P., 2013. Is it time to look at your website?. *Builders Merchants Journal,* pp. 19-19.
- Turban, E. et al., 2010. *Electronic Commerce 2010: A Managerial Perspective.* s.l.:Prentice Hall.
- Ionel, N., 2008. Critical Analysis of the Scrum Project Management Methodology. *Annals of the University of Oradea, Economic Science Series,* pp. 435-441.

6. ANEXOS I – Revisão Bibliográfica

A. Vantagens e desvantagens da metodologia de desenvolvimento em cascata (Munassar & Govardhan, 2010)

Vantagens	Desvantagens
Fácil entendimento e implementação	Idealizado, não reflete a realidade
Identifica metas e produtos finais	Incapacidade de conhecer todos os requisitos no início do desenvolvimento
Orientado para a documentação	Entrega de *software* tardia leva ao descobrimento de erros já numa fase avançada do projeto
Funciona relativamente bem em equipas fracas	Difícil e caro efetivar mudanças na documentação
Definição clara de responsabilidades	Elevados custos de gestão levam a que não seja adequado para pequenas equipas de desenvolvimento

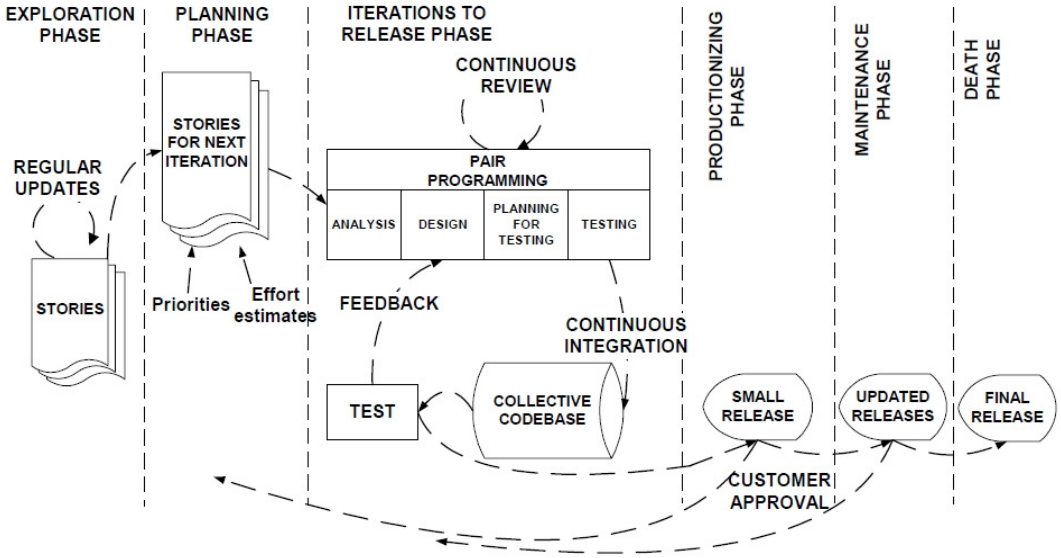
B. O ciclo de vida da metodologia de desenvolvimento *Scrum* (Abrahamsson et al., 2002)



C. Vantagens e desvantagens da metodologia de desenvolvimento *Scrum* (Ionel, 2008)

Vantagens	Desvantagens
Lançamento de novas versões funcionais	Difícil de planear projetos pouco definidos
Permite mudanças de direção no projeto	Alterações e reuniões diárias aumentam os custos do projeto
Alteração de requisitos facilitada pelo envolvimento dos clientes	Depende largamente da boa comunicação entre membros da equipa de projeto
Partilha de conhecimento	Necessidade de ter o cliente disponível para testar mensalmente
Bom ambiente de aprendizagem	Não é fácil de implementar em grandes projetos

D. O ciclo de vida da metodologia de desenvolvimento *XP* (Abrahamsson et al., 2002)



E. Vantagens e desvantagens da metodologia de desenvolvimento *XP* (Munassar & Govardhan, 2010)

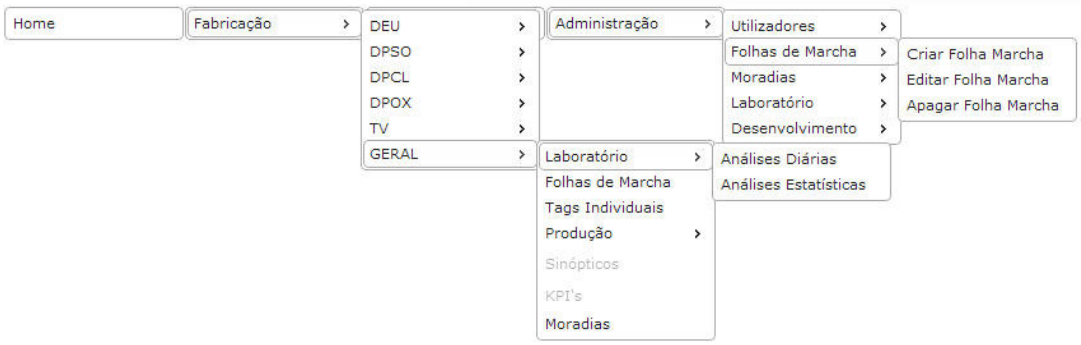
Vantagens	Desvantagens
Apropriado para projetos de média e pequena dimensão	Difícil de aplicar a grandes projetos
Garantem coesão da equipa de projeto	Necessita experiência para não degenerar em *code-and-fix*
Dá grande importância ao produto final	Pares de programação são caros
Iterações permitem correção de erros	A construção de casos de teste é um trabalho difícil e especializado
Desenvolvimento por testes garante qualidade	

F. Parâmetros de *design* de *websites* (Cebi, 2013)

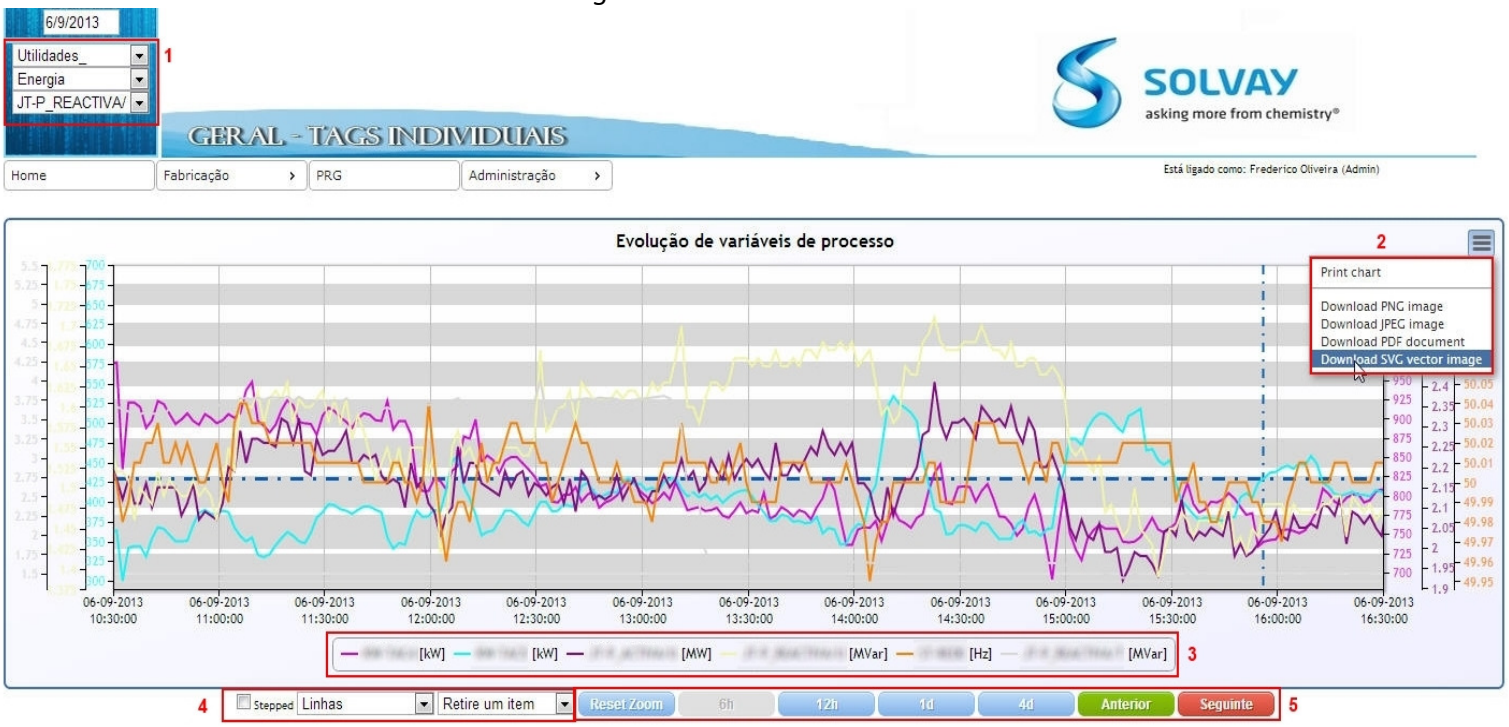
	Main design parameters		Sub-design parameters		Explanation
Website design parameters	C1	Usability			
			C_{11}	Ease of use	The users should reach its aim in short time while using the site first time
			C_{12}	Ease of learning	The users should be adapted the site in short time
			C_{13}	Memorability	The users should remember the functions presented by the site
	C_2	Visual aspects			
			C_{21}	Layout	The site should present good visual organization
			C_{22}	Graphics	The site should present good tonality
			C_{23}	Text	The site should present readable font
	C_3	Technical adequacy			
			C_{31}	System availability	The site must be reached any time
			C_{32}	Speed	The site should provide quick loading, accessing, and using
			C_{33}	Accessibility	The site should provide easy access to materials
			C_{34}	Navigation	The site should provide easy navigation to reach services
	C_4	Content			The site should satisfy users' expectations
	C_5	Security			
			C_{51}	Reliability	The service protect users from hackers' attack while downloading a file or surfing
			C_{52}	Accuracy	The service provide correct information
			C_{53}	Privacy	The site protects users' information
	C_6	Communication			
			C_{61}	Contact info	The site should provide contact addresses and phone numbers
			C_{62}	Online help	The site should provide an assistance service through phone or internet
			C_{63}	Responsiveness	The site should handle user's problems and return to users in a short time
	C_7	Prestige			
			C_{71}	Reputation	The site should be well known
			C_{72}	Sustainability	The site should guarantee to serve for a long time
			C_{73}	Currency	The site should provide continuous improvement

7. ANEXOS II – Projeto

A. Menu de navegação

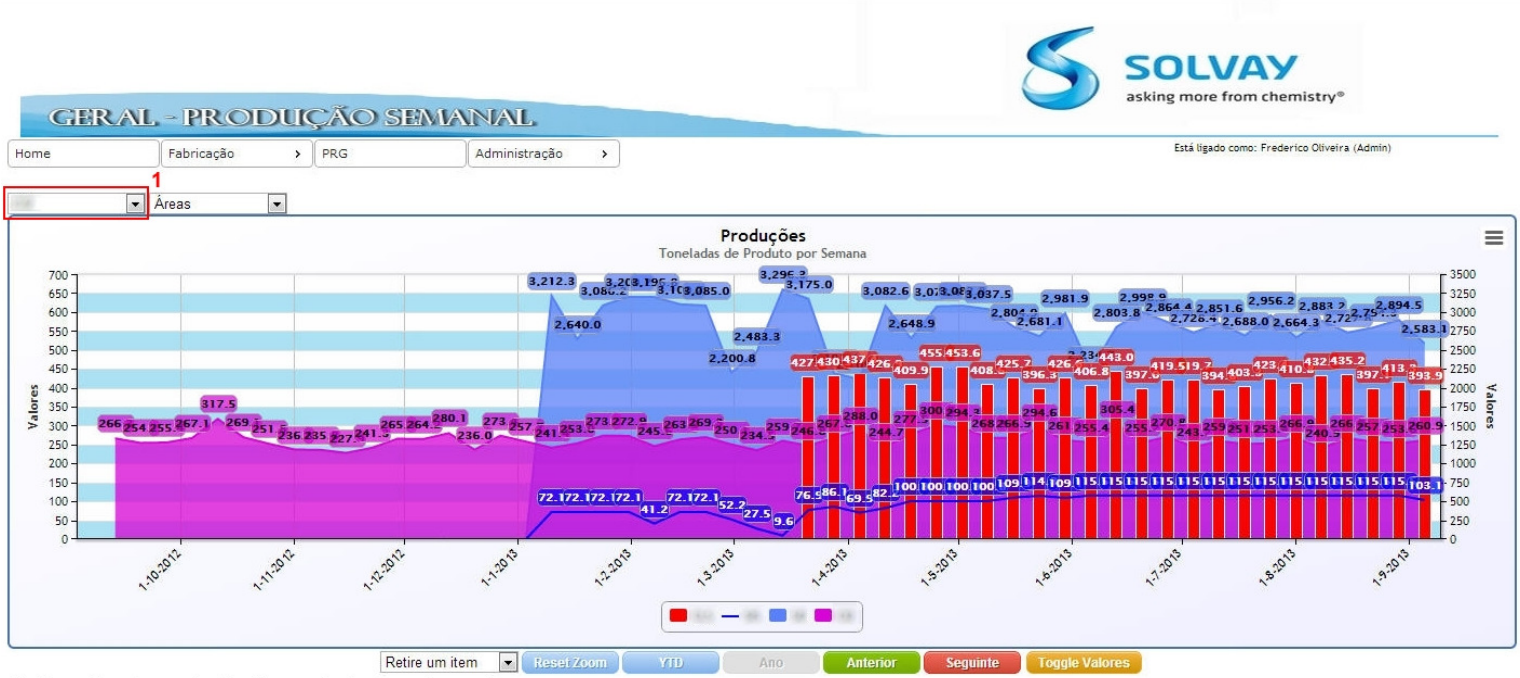


B. Consulta de *Tags* individuais



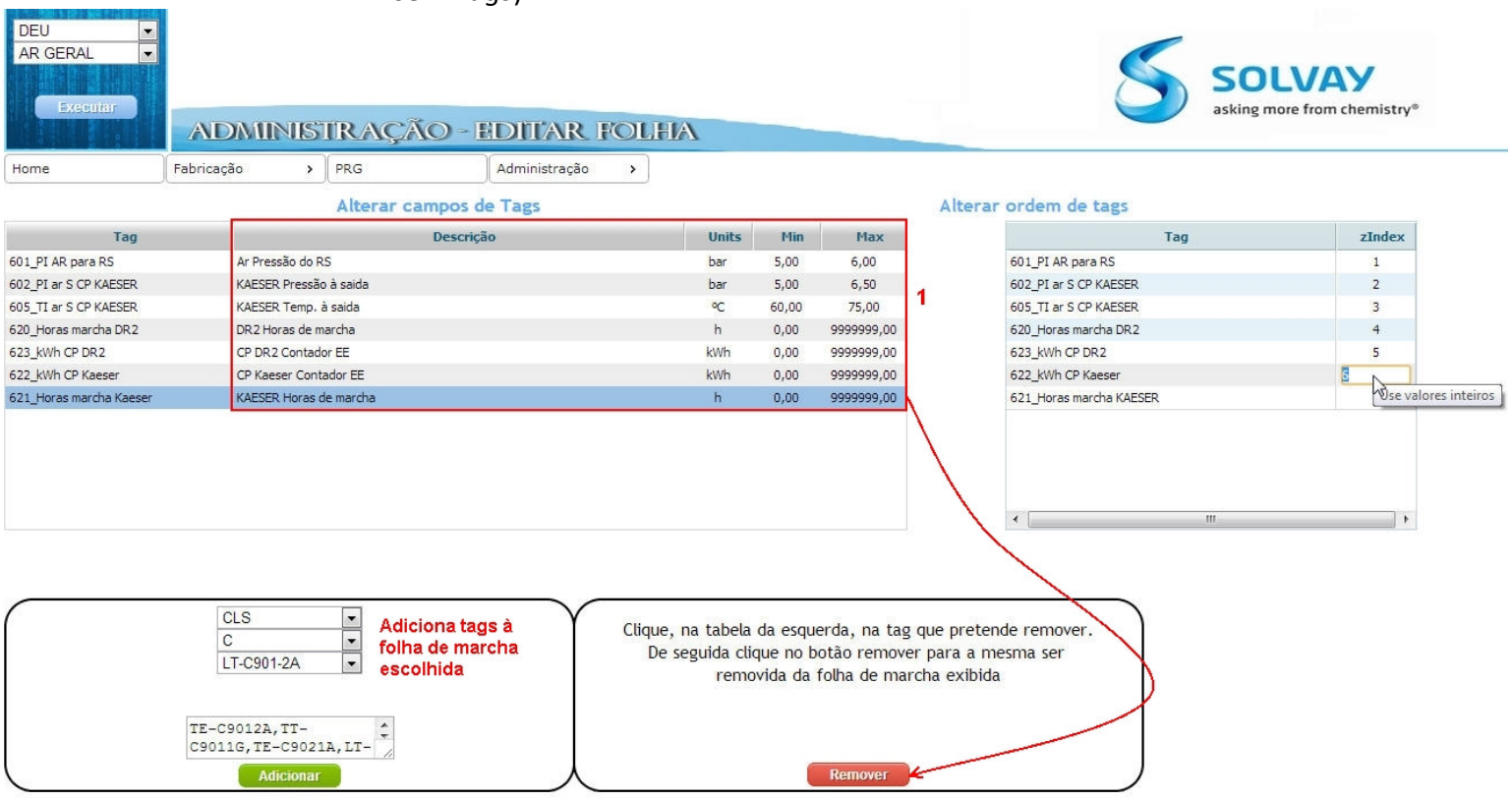
1 - Escolher tags individualmente, numa replicação da navegação por pastas similar à já utilizada noutras aplicações da empresa; 2 - Opções de exportação do gráfico; 3 - Clicando num elemento da legenda causará que a linha correspondente seja escondida; 4 - Escolher se a nova linha de gráfico virá em escada, em linhas/colunas/áreas e por último apagar definitivamente uma série do gráfico; 5 - Navegação temporal dos dados

C. Produções Semanais



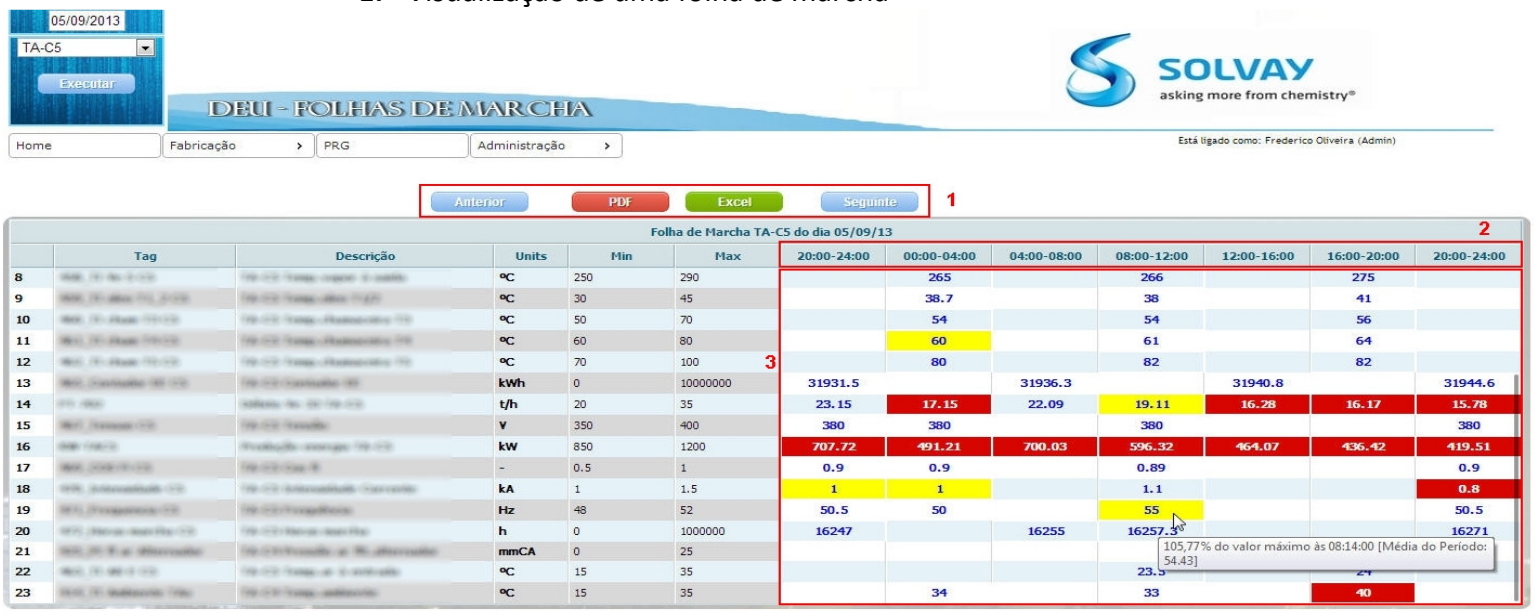
1 - Escolha da produção de um dado departamento

D. Edição de folhas de marcha (Apenas necessária uma folha de marcha sem *Tags*)



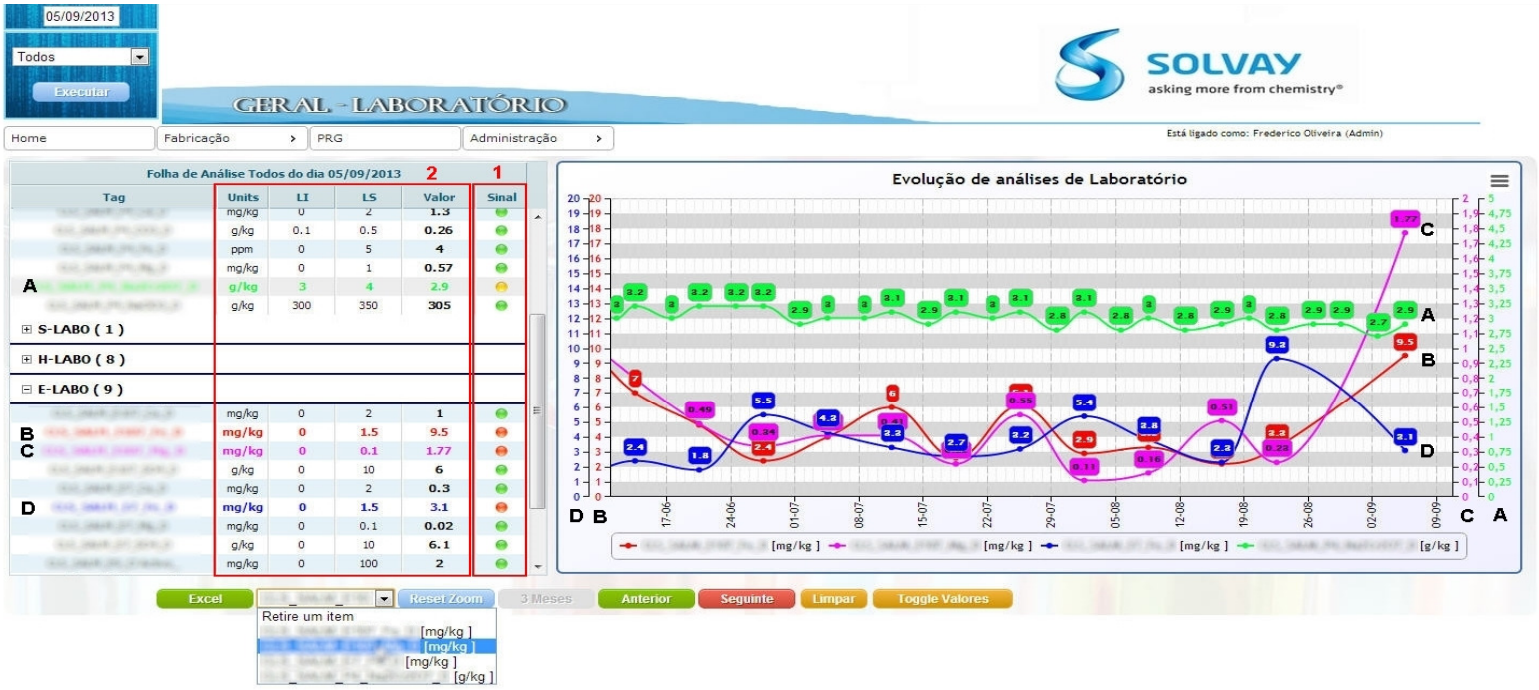
1 - Campos editáveis (entrar em modo edição através de duplo clique na célula que se pretende editar)
Após a execução de um update é sempre feito um refresh das tabelas. Assim o utilizador tem sempre a informação actualizada

E. Visualização de uma folha de marcha



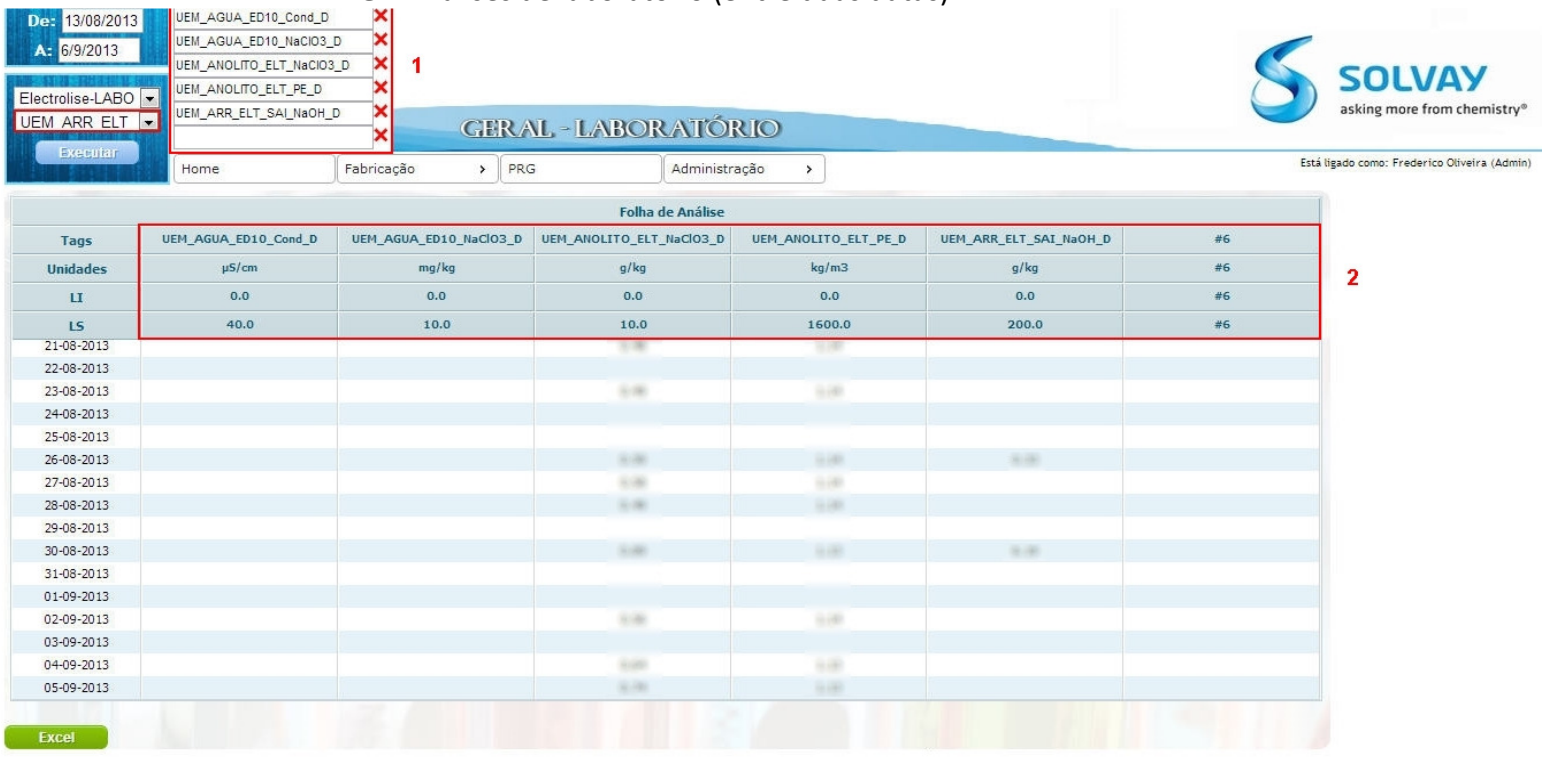
1 - Navegação para o dia anterior/seguinte e opções de exportação da tabela; 2 - Períodos de tempo (começa nas 20 horas do dia anterior ao pedido);
3 - Tal como está exemplificado na legenda da linha 19, existem outros dados informativos adicionais à disposição dos utilizadores;
Se por exemplo ainda fosse 19 horas do dia 05/09/2013 a última coluna estaria vazia e, a coluna das 16h-20h teria (caso existisse) o último valor observado entre as 16h e as 19h (hora a que o pedido foi feito)

F. Análises de laboratório diárias



Valores de análises agrupados por secções de fabricação com alertas visuais 1 para valores fora dos limites estabelecidos (LI e LS); Cores de Tags escolhidas com a mesma cór na tabela e gráfico (dados de gráfico representam 3 meses passados a partir da data do pedido); Quando uma Tag é eliminada do gráfico, essa mesma Tag voltará à sua cor normal (preto) na tabela
2 - Valores também presentes em VEGA

G. Análises de laboratório (entre duas datas)



1 - Não é permitida a repetição de Tags, bem como se pretender substituir alguma bastará apagá-la da input box e escolher outra, e assim será sempre a primeira caixa vazia a receber o valor; 2 - Valores também presentes em VEGA

H. Edição de dados de laboratório

SOLVAY
asking more from chemistry®

ADMINISTRAÇÃO - TAGS LABORATÓRIO

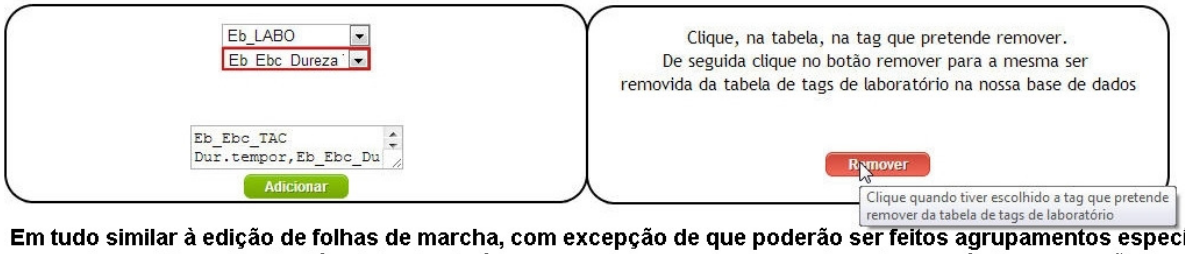
Home | Fabricação › | PRG | Administração ›

1

CLS

Alterar campos de Tags de Laboratório

Departamento	Secção	Tag	Descrição	Units	Min	Max
F (1)						
CLS	F	CLS_SALM_FRIG_NaClO3_D	NaClO3 na Salmoura Frigorifica	g/l	0,0000	5,0000
H (31)						
P (11)						
CLS	P	CLS_SALM_P2_Ca_D	Ca na Salmoura do P403 - P2	ppm	0,0000	5,0000
CLS	P	CLS_SALM_P3A_P3B_Ca_D	Ca na Salmoura s. FL - P3	ppm	0,0000	2,0000
CLS	P	CLS_SALM_P4_Ca_D	Ca na Salm. do P005 - P4	mg/kg	0,0000	2,0000
CLS	P	CLS_SALM_P4_CO3_D	CO3 na Salm. do P005 - P4	g/kg	0,1000	0,5000
CLS	P	CLS_SALM_P4_Fe_D	Fe na Salm. Do P005 - P4	ppm	0,0000	5,0000
CLS	P	CLS_SALM_P4_Massa Vol_D	Massa Vol. na Salm. do P005 - P4	kg/l	0,0000	1500,0000
CLS	P	CLS_SALM_P4_Mg_D	Mg na Salm. do P005 - P4	mg/kg	0,0000	1,0000



Em tudo similar à edição de folhas de marcha, com excepção de que poderão ser feitos agrupamentos específicos (edição de campo secção com valores diferentes dos já existentes fará com que um novo agrupamento surja na próxima execução da gestão de dados de laboratório);
1 - Campos editáveis de cada Tag

I. Exportação de tabelas para *Excel*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Folha de Marcha TA-C5 do dia 05/09/13														
2		Tag	Descrição	Units	Min	Max	20:00-24:00	00:00-04:00	04:00-08:00	08:00-12:00	12:00-16:00	16:00-20:00	20:00-24:00	Média	Variância
3	1	[illegible]	[illegible]	bar	25	35		32		32		33		32,33333333	0,33333
4	2	[illegible]	[illegible]	bar	8	12		10,5		10,3		10,2		10,33333333	0,02333
5	3	[illegible]	[illegible]	bar	0,5	2		1,3		1,3		1,2		1,266666667	0,00333
6	4	[illegible]	[illegible]	bar	1	3		1		1		1		1	0
7	5	[illegible]	[illegible]	bar	4	6		4,5		3,4		3		3,633333333	0,60333
8	6	[illegible]	[illegible]	bar	9	12		8,5		8,4		8,4		8,433333333	0,00333
9	7	[illegible]	[illegible]	°C	300	360		350		350		350		350	0
10	8	[illegible]	[illegible]	°C	250	290		265		266		275		268,6666667	30,3333
11	9	[illegible]	[illegible]	°C	30	45		38,7		38		41		39,23333333	2,46333
12	10	[illegible]	[illegible]	°C	50	70		54		54		56		54,66666667	1,33333
13	11	[illegible]	[illegible]	°C	60	80		60		61		64		61,66666667	4,33333
14	12	[illegible]	[illegible]	°C	70	100		80		82		82		81,33333333	1,33333
15	13	[illegible]	[illegible]	kWh	0	10000000	31931,5		31936,3		31940,8		31944,6	31938,3	32,06
16	14	[illegible]	[illegible]	t/h	20	35	23,15	17,15	22,09	19,11	16,28	16,17	15,78	18,53285714	9,07556
17	15	[illegible]	[illegible]	V	350	400	380	380		380			380	380	0
18	16	[illegible]	[illegible]	kW	850	1200	707,72	491,21	700,03	596,32	464,07	436,42	419,51	545,04	15021
19	17	[illegible]	[illegible]	-	0,5	1	0,9	0,9		0,89			0,9	0,8975	2,5E-05
20	18	[illegible]	[illegible]	kA	1	1,5	1	1		1,1			0,8	0,975	0,01583
21	19	[illegible]	[illegible]	Hz	48	52	50,5	50		55			50,5	51,5	5,5
22	20	[illegible]	[illegible]	h	0	1000000	16247		16255	16257,3			16271	16257,575	99,5892
23	21	[illegible]	[illegible]	mmCA	0	25								#DIV/0!	#DIV/0!
24	22	[illegible]	[illegible]	°C	15	35				23,5		24		23,75	0,125
25	23	[illegible]	[illegible]	°C	15	35		34		33		40		35,66666667	14,3333
26															

Exemplo de uma folha de marcha exportada para Excel através da aplicação. Em tabelas de análises diárias de laboratório, apenas aparecem valores para os agrupamentos expandidos. No caso específico do ANEXO II. f. os agrupamentos 'S-Labo' e 'H-Labo' não seriam exportados

J. Exportação de tabelas para PDF

Folha de Marcha TA-C5 do dia 05/09/13												
	Tag	Descrição	Units	Min	Max	20:00-2...	00:00-0...	04:00-0...	08:00-1...	12:00-1...	16:00-2...	20:00-2...
1			bar	25	35		32		32		33	
2			bar	8	12		10.5		10.3		10.2	
3			bar	0.5	2		1.3		1.3		1.2	
4			bar	1	3		1		1		1	
5			bar	4	6		4.5		3.4		3	
6			bar	9	12		8.5		8.4		8.4	
7			ºC	300	360		350		350		350	
8			ºC	250	290		265		266		275	
9			ºC	30	45		38.7		38		41	
10			ºC	50	70		54		54		56	
11			ºC	60	80		60		61		64	
12			ºC	70	100		80		82		82	
13			kWh...	0	10000000	31931.5		31936.3		31940.8		31944.6
14			t/h	20	35	23.15	17.15	22.09	19.11	16.28	16.17	15.78
15			V	350	400	380	380		380			380
16			kW ...	850	1200	707.72	491.21	700.03	596.32	464.07	436.42	419.51
17			-	0.5	1	0.9	0.9		0.89			0.9
18			kA	1	1.5	1	1		1.1			0.8
19			Hz	48	52	50.5	50		55			50.5
20			h	0	1000000	16247		16255	16257.3			16271
21			mm...	0	25							
22			ºC	15	35				23.5		24	
23			ºC	15	35		34		33		40	

O mesmo que foi explicado no anexo anterior acontece na exportação para formato PDF. Agrupamentos que não estejam expandidos não serão exportados para PDF

K. Tempos de resposta

	II.B	II.C	II.E	II.F	II.G
#obs	30	15	30	30	15
Média (em segundos)	0,900	0,753	2,062	5,200	1,676