Relatório de Projeto

Mestrado em Engenharia Informática – Computação Móvel

Accept Web – Aplicação Web para controlo de enchimento de pré-embalados

João Miguel Moital Dias

Leiria, Setembro de 2016

Relatório de Projeto

Mestrado em Engenharia Informática – Computação Móvel

Accept Web – Aplicação Web para controlo de enchimento de pré-embalados

João Miguel Moital Dias

Projeto de Mestrado realizado sob a orientação do Doutor Marco António de Oliveira Monteiro, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, Setembro de 2016

Agradecimentos

Quero agradecer a todos aqueles que de forma direta, ou indireta, tornaram possível a realização do estágio e a elaboração deste relatório.

Ao Instituto Politécnico de Leiria e aos docentes que contribuíram para a minha formação tanto ao nível da licenciatura como do mestrado, e em especial ao meu orientador, o Doutor Marco Monteiro pela sua ajuda e orientação durante todo o processo relativo ao estágio e à elaboração deste relatório.

A todos os meus colegas e amigos que me acompanharam ao longo do meu percurso académico, pela sua ajuda e auxílio.

A todos os colaboradores da Sinmetro e da Aferymed por me receberem e me ajudarem a crescer a nível profissional e pessoal.

Esta página foi intencionalmente deixada em branco

Resumo

O presente relatório enquadra-se no âmbito da unidade curricular de Projeto do mestrado em Engenharia Informática – Computação Móvel da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

O objetivo deste relatório é descrever o processo referente aos 9 meses do estágio que foi realizado na empresa Sinmetro, Lda e o trabalho desenvolvido nesse estágio. Neste relatório poderão ser encontradas informações sobre o local de estágio, o trabalho desenvolvido e as conclusões tiradas após o término do estágio.

O objetivo do estágio foi o de criar uma aplicação *Web* que facilite a gestão e controlo da produção de produtos pré-embalados por parte dos produtores e fornecedores, focando-se maioritariamente na apresentação de dados visuais na forma de tabelas e gráficos.

Esta aplicação complementa o *software* ACCEPT, criado pela Sinmetro, apresentando muitas das funcionalidades já existentes, transportando-o para o ambiente *Web*.

Palavras-chave: Laravel, Web Site, JavaScript, Framework, Metrologia,Pré-embalados

Esta página foi intencionalmente deixada em branco

Abstract

This report falls within the Project course of the Master's degree in Computer Engineering - Mobile Computing from the School of Technology and Management of the Polytechnic Institute of Leiria.

The objective of this report is to describe all the process regarding the 9 months internship that was held in the company Sinmetro Ltd as well as the work that was developed during the internship. This report presents information about the internship location, the work developed over the internship period and the conclusions drawn after the completion of the internship.

The goal of the internship was to create a Web application whose purpose is to facilitate the management and control of the production of pre-packaged products by the producers and suppliers, focusing mainly on the presentation of visual data in the form of charts and graphs.

This application complements the ACCEPT software, created by Sinmetro presenting many of the existing features and transporting them to the Web environment.

Keywords: Laravel, Web Site, JavaScript, Framework, Metrology,Pre-packaged.

Esta página foi intencionalmente deixada em branco

Índice de Figuras

Índice de Tabelas

Esta página foi intencionalmente deixada em branco

Índice de Excertos de Código Fonte

Esta página foi intencionalmente deixada em branco

Lista de Siglas

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DOM	Document Object Model
HTTP	Hypertext Transfer Protocol
IPQ	Instituto Português da Qualidade
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
MVC	Model-View-Controller
OEE	Overall Equipment Effectiveness
OVM	Organismo de Verificação Metrológica
PHP	PHP:Hypertext Preprocessor
REST	Representational State Transfer
SPA	Single Page Application
SQL	Structured Query Language
VPN	Virtual Private Network
WAMP	Windows, Apache, MySQL, PHP

Esta página foi intencionalmente deixada em branco

Índice

1 – Introdução

O enchimento de produtos pré-embalados é um processo que necessita de obedecer a várias regras e leis de maneira a que seja considerado dentro dos conformes. Todos os lotes que são produzidos devem obedecer a estas regras para que possam seguir para o mercado. Dentro destes lotes, todos os produtos têm uma quantidade nominal definida que é a base para se verificar se o produto obedece ou não às leis.

As empresas são também responsáveis pelo registo e preservação de registos periódicos aos produtos. Geralmente este é um processo difícil e moroso quando realizado manualmente, sendo que muitas vezes as empresas acabam por deixar de o realizar pois consome demasiado tempo e recursos, sujeitando-se a possíveis coimas caso não realizem esses registos.

Para auxiliar neste processo a Sinmetro criou o *software* ACCEPT, disponível para ambientes *desktop*. Este *software* é constituído por vários módulos, e na sua essência permite às empresas um maior controlo sobre o seu processo de enchimento dos pré-embalados. Após a recolha dos dados referentes à pesagem dos produtos, é possível obter várias informações estatísticas e legais. Com estas informações as empresas conseguem ter a noção do funcionamento das suas linhas de produção ao longo do tempo, permitindo ajustar os processos de enchimento quando necessário. A obtenção dos dados pode ser feita através de introdução manual de valores previamente pesados ou diretamente de balanças existentes ao longo das linhas de produção, agilizando consideravelmente o processo. No entanto, uma vez que o ACCEPT é uma aplicação local, o acesso a estes dados apenas é possível nas instalações físicas das empresas, sendo necessário o uso de *proxies* ou VPNs (*Virtual Private Networks*) para o acesso aos mesmos quando fora da empresa.

Tendo em conta as atuais limitações da versão *desktop* do ACCEPT, a Sinmetro decidiu investir no desenvolvimento de uma aplicação *Web* que permitisse às empresas aceder às

informações internas das suas fábricas através da internet e que fornecesse uma interface mais atual e completa.

Foi neste contexto que o estágio se desenrolou, tendo como objetivo principal a criação de um protótipo funcional que satisfaça as necessidades das empresas no controlo metrológico dos seus produtos de forma ubíqua. Pretende-se assim oferecer as funcionalidades já existentes do *software* de *desktop* num ambiente *web*, proporcionando uma plataforma mais moderna e acessível, dotada de uma interface simples e personalizável, permitindo ao utilizador uma melhor gestão dos processos de enchimento.

Este relatório descreve todo o processo de investigação e desenvolvimento realizados durante os 9 meses de estágio na empresa Sinmetro.

No capítulo 2 (Caracterização do estágio) é feita uma contextualização do estágio descrevendo a empresa de acolhimento, o objetivo do estágio e as condições de desenvolvimento oferecidas pela empresa. O terceiro capítulo (Estado da Arte) apresenta a investigação realizada durante o estágio, focando-se em aspetos como metrologia e *software* de controlo metrológico. No capítulo 4 (Implementação) é apresentada a fase de implementação, descrevendo os requisitos do projeto, a metodologia de desenvolvimento usada, a arquitetura do projeto, as várias tecnologias utilizadas para desenvolver o ACCEPT *Web*, bem como os desafios encontrados no decurso do desenvolvimento do projeto. No capítulo 5 (Resultados e Testes) são apresentados os testes realizados à aplicação e os resultados obtidos na aplicação após a realização do estágio. Finalmente, no capítulo 6 (Conclusão) é feita uma conclusão em relação ao estado do projeto na fase final do estágio.

2 – Caracterização do estágio

Neste capítulo é feita uma apresentação sobre a empresa onde decorreu o estágio, dando a conhecer o seu historial e as condições de trabalho, bem como os objetivos gerais do estágio.

2.1 Empresa

A Sinmetro foi fundada em 2002 e desde essa altura que trabalha com a indústria e serviços na implementação de sistemas de informação, 100% desenvolvidos pela Sinmetro, formação e consultoria em áreas como Investigação & Desenvolvimento Tecnológico (I&DT), Inovação, Melhoria Contínua e/ou Reengenharia de Processos através das metodologias *Lean & 6 Sigma*.

Relativamente ao espaço físico é uma empresa constituída por uma sala de reuniões, uma sala de refeições, a sala de trabalho e casas de banho. Este espaço é geralmente usado por 7 colaboradores. A Sinmetro partilha o espaço com a empresa Aferymed [2], que tem 5 colaboradores, um dos quais é a sócia fundadora das 2 empresas, totalizando 11 pessoas no local de trabalho.

A Sinmetro é a empresa responsável pelo desenvolvimento do sistema ACCEPT. Este é um *software* de registo, medição e controlo da qualidade e eficiência da produção das empresas. É uma solução integrada para a gestão de dados no chão da fábrica que tem por objetivos maximizar a capacidade das linhas de produção, minimizar ou eliminar tarefas de suporte e falhas na rastreabilidade, bem como fornecer um "Painel de Bordo" de *Key Performance Indicators* (KPI) que avaliem a eficiência e eficácia das operações. Este sistema foi criado em 2001 com o módulo de controlo estatístico do peso em linhas de enchimento testado na fábrica da Compal em Almeirim e Lactogal na Tocha, e contempla até à data 6 módulos padrão (Peso, Qualidade, OEE, Micro, Mobile e Sensorial) adaptados de acordo com as

necessidades do cliente. O sistema ACCEPT está atualmente em 49 fábricas em Portugal e Espanha, no setor alimentar, automóvel e indústria química [26].

2.2 Objetivo do estágio

O objetivo do estágio foi o desenvolvimento de uma aplicação *Web* cujo objetivo é permitir às empresas verificar em tempo real e em qualquer lugar, o estado das suas linhas de produção de pré-embalados, permitindo acesso a dados estatísticos e legais detalhados. De certa forma, pretende-se reproduzir as funcionalidades já existentes no ACCEPT *desktop* para a plataforma *Web* permitindo um acesso de forma ubíqua às funcionalidades já existentes. Com esta transição para um ambiente *online* pretende-se também tirar partido das vantagens inerentes a este modelo de disponibilização de *software*, como por exemplo a fácil gestão de versões entre os utilizadores e a maior facilidade nas condições de licenciamento.

2.3 Condições de Trabalho

O trabalho realizado no decorrer do estágio foi desenvolvido num computador portátil disponibilizado pela Sinmetro com o sistema operativo Windows 10.

No início do estágio foi instalado um servidor PHP (*PHP:Hypertext Preprocessor*) para o desenvolvimento do ACCEPT Web alojado com recurso ao *software* WAMP (Windows, Apache, MySQL, PHP) [1]. Este servidor estava presente num computador da empresa que era acedido através da rede local.

Numa fase inicial do processo de desenvolvimento a troca de dados entre o servidor PHP e a base de dados era realizada com recurso a uma API (*Application Programming Interface*) REST (*Representational State Transfer*). Esta API era acedida através de um *software* de *proxy* desenvolvido pela Sinmetro e estava alojada também no computador do servidor PHP. Foram também usadas várias bases de dados com dados reais de clientes Sinmetro para validar o funcionamento da aplicação e a representação dos dados.

Para a escrita de código foi escolhido o editor de texto Sublime Text 2 [3] e como ferramenta de gestão de projeto foi usado o Asana [19] uma vez que era a ferramenta já utilizada na empresa.

3 – Estado da Arte

Neste capítulo é feita uma introdução à Metrologia, focando principalmente a Metrologia Legal de pré-embalados e são apresentadas algumas ferramentas para o controlo metrológico de pré-embalados existentes no mercado.

3.1 Metrologia

Segundo o documento de vocabulário internacional de metrologia [27], a metrologia é a "*ciência da medição e das suas aplicações*" e "[…] *engloba todos os aspetos teóricos e práticos da medição, qualquer que seja a incerteza de medição e o campo de aplicação*".

O objetivo da metrologia é garantir o rigor nas medições feitas, recorrendo a unidades padronizadas para a comparação de grandezas da mesma natureza. Este rigor é geralmente garantido pela calibração dos equipamentos de medição para a grandeza a medir.

A metrologia pode ser dividida em 3 áreas (Metrologia Científica, Metrologia Industrial e Metrologia Legal), sendo que no âmbito do estágio foi feito um estudo sobre a Metrologia Legal, no contexto dos pré-embalados, apresentado na secção seguinte.

3.1.1. Controlo Metrológico de pré-embalados

Em 1989, a Organização Internacional de Metrologia Legal (OIML) lançou uma recomendação que descreve os procedimentos a adotar aquando do controlo metrológico dos pré-embalados [28]. Este documento tornou-se a referência para muitas das leis existentes sobre o controlo de pré-embalados, sendo em 2004 lançada uma revisão do mesmo [29].

Segundo o decreto lei nº 199/2008 de 8 de outubro de 2008 [30], um produto pré-embalado é um "[…] *produto cujo acondicionamento foi efectuado antes da sua exposição para venda ao consumidor em embalagem que solidariamente com ele é comercializada, de tal modo que a quantidade de produto contido na embalagem tenha um valor previamente escolhido e não possa ser alterada sem que a embalagem seja aberta ou sofra uma alteração perceptível.".* Significa isto que é considerado pré-embalado o produto que seja embalado antes da sua

venda ao consumidor e sem a intervenção ou presença do mesmo e que tenha uma quantidade nominal fixa que não possa ser modificada sem que a integridade da embalagem seja violada. Os artigos publicados neste decreto-lei dizem apenas respeito a todos os produtos pré-embalados que apresentem uma quantidade nominal igual ou superior a 5 gramas ou 5 mililitros e igual ou inferior a 10 quilogramas ou 10 litros que se destinem ao consumo dentro de União Europeia.

Este decreto-lei refere também que antes de um produto ser comercializado deverá ser feito um controlo metrológico de forma a garantir que os produtos obedecem às leis impostas. Após a sua aceitação, o produto deverá ser submetido a um controlo metrológico anual.

Para além do controlo metrológico, a entidade responsável pelo embalamento ou importação dos produtos deverá conservar o registo das medições periódicas dos mesmos, procedendo às correções e ajustes necessários ao cumprimento da lei. O registo destas medições é geralmente feito através de um processo manual de inserção de valores para uma folha de cálculo num computador, ou através do registo dos valores em papel. Existem também balanças preparadas para recolher os dados das pesagens e imprimir os resultados das várias pesagens para um talão que contém alguns dados estatísticos (ver capítulo 3.2.1). Os instrumentos de medição usados para o registo das medições deverão ter um grau de sensibilidade e precisão adequado ao tipo de produto a controlar. É importante que os responsáveis pelas medições tenham um sistema que lhes permita monitorizar e controlar o processo produtivo ao nível das quantidades e estabelecer os limites de controlo.
Estes documentos deverão ser conservados durante:

- Um ano para produtos com prazo de validade até 3 meses;
- Três anos para produtos com prazo de validade entre 3 e 18 meses;
- Cinco anos para produtos com prazo de validade superior a 18 meses.

O controlo metrológico dos pré-embalados deverá ser feito por uma entidade externa qualificada pelo Instituto Português da Qualidade (IPQ), conhecida como Organismo de Verificação Metrológica (OVM) [31]. Durante o controlo metrológico, um OVM deverá:

- Avaliar os modelos estatísticos usados no controlo do processo de enchimento/embalamento;
- Analisar os registos das medições periódicas;

- Realizar os ensaios de verificação metrológica de acordo com a Portaria nº1198/91 de 18 de Dezembro;
- Emitir um certificado caso o processo demonstra estar conforme.

Na Portaria nº1198/91 de 18 de Dezembro [32] é definido o regulamento do controlo metrológico das quantidades dos produtos pré-embalados. O controlo metrológico do produto é realizado nas instalações do respetivo responsável (podendo os produtos ser retirados do armazém ou diretamente da linha de produção) e pode seguir a abordagem destrutiva ou não destrutiva conforme o tipo de produto, dando-se sempre preferência à abordagem não destrutiva. Na abordagem destrutiva é necessário abrir a embalagem do produto para que se possa ter acesso apenas ao peso real do produto. Por exemplo, para determinar o peso escorrido de uma lata de salsichas é necessário abrir a embalagem e escorrer o líquido no seu interior. No final de um ensaio destrutivo o produto não poderá ser comercializado. Na abordagem não destrutiva é possível determinar o peso do produto através da subtração da tara da embalagem vazia ao peso da embalagem cheia. Uma vez que a embalagem não é violada neste ensaio, o produto poderá depois ser comercializado, consoante o resultado favorável ou desfavorável do controlo metrológico.

De forma determinar a tara a usar para nos ensaios é seguida a seguinte logica:

- É usada a média da tara de 10 unidades caso a massa da tara represente um valor inferior a 10% da massa bruta do produto;
- É usada a média de 20 unidades caso a massa da tara represente um valor superior a 10% da massa bruta do produto e o desvio padrão da massa da tara for inferior ou igual a um quarto do erro admissível por defeito dos pré-embalados;
- É usada a tara individual dos produtos nos restantes casos.

Para que o processo de enchimento/embalamento de um produto possa ser considerado conforme, deverá obedecer às seguintes regras:

- O seu conteúdo efetivo não deve ser inferior, em média, à quantidade nominal nele marcado;
- Apenas 2.5% dos produtos (1 em 40) dentro de um lote podem ter uma quantidade nominal abaixo do erro admissível;
- Nenhum pré-embalado deve ter um erro, por defeito, superior ao dobro do erro admissível.

Na tabela 1 está apresentado o erro admissível conforme a Portaria nº1198/91 de 18 de Dezembro [32].

Quantidade nominal do pré-embalado (grama ou mililitro)	**Erros admissíveis por defeito**	
	Percentagem	Em massa ou volume (grama ou mililitro)
Até 50	9,0	-
De 50 a 100	-	4,5
De 100 a 200	4,5	-
De 200 a 300	-	9,0
De 300 a 500	3,0	-
De 500 a 1000	-	15,0
De 1000 a 10000	1,5	-

Tabela 1 - Erros admissíveis por defeito nos conteúdos efetivos. Adaptado da portaria nº1198/91 de 18 de Dezembro [32].

Após um controlo metrológico com resultado positivo é emitido um certificado e o produto deverá ser remetido a um novo controlo metrológico até ao dia 31 de dezembro do ano seguinte. Após a obtenção do certificado, poderá ser colocado no rótulo da embalagem a marca de conformidade apresentada na figura 1. A colocação desta marca é opcional para a empresa embaladora e garante ao consumidor que o produto obedece aos critérios legais existentes. Caso se opte pela colocação da marca, esta deverá estar situada no mesmo campo de visão da quantidade nominal, repetindo-se sempre que quantidade nominal for indicada na embalagem. Apesar da colocação da marca de conformidade estar dependente de um resultado positivo do ensaio de controlo metrológico, algumas empresas colocam a marca nos seus produtos mesmo que este resultado do ensaio tenha sido negativo ou mesmo sem terem realizado o ensaio (geralmente por desconhecimento), o que constitui um ato ilegal nos termos da lei. A marca de conformidade deverá ter no mínimo 3 mm de altura, e deverá manter todas as proporções caso sofra mudanças de tamanho.

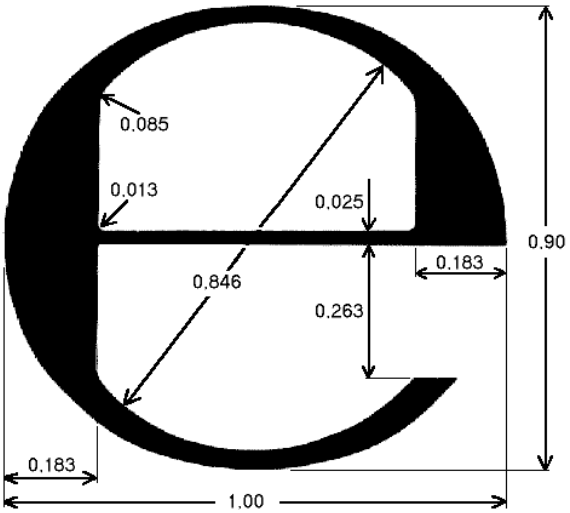


Figura 1 - Marca de conformidade "e". Imagem retirada do site do instituto de pesquisa técnica da Suécia [33]

Se um produto não obedecer aos critérios legais, deverão ser tomadas as medidas necessárias para corrigir os erros detetados. Neste caso, poderão ser feitos os ajustes na própria hora e refeito o ensaio até que se obtenha um resultado positivo. Nos casos em que este cenário é impraticável, deverá ser feito um controlo metrológico numa data futura.

Muitas vezes o certificado do controlo metrológico é requerido pelos clientes (como por exemplo cadeias de hipermercados), servindo como condicionante para a sua comercialização por parte do mesmo.

3.2 *Software* Controlo Metrológico

Uma vez que o registo de medições por parte dos embaladores/produtores é obrigatório por lei, é útil recorrer a ferramentas para auxiliar neste processo. Consoante o ritmo de produção de cada fábrica, poderão existir alternativas mais eficientes que outras. Para empresas produtoras com uma produção horária relativamente alta, o registo de medições em papel torna-se um processo pesado em termos de registo e armazenamento. Além disso, a procura de informação pode tornar-se também bastante difícil e morosa. Cada embalador ou produtor poderá ter um método preferido para efetuar o registo, armazenamento e análise estatística de dados de medições de pré-embalados. O *software* torna-se assim numa ajuda que permite otimizar a recolha, tratamento estatístico e pesquisa dos registos. Nesta secção são apresentadas apenas algumas das alternativas conhecidas.

3.2.1. *Software* de Balanças

Um dos métodos para o controlo e registo das medições feitas pelas produtoras ou embaladoras é o uso de *software* integrado em balanças. Estas balanças permitem imprimir talões com a estatística de várias medições feitas. Por exemplo, a balança "FPVO-Waage FKTF" [34] é uma balança que obedece às normas de aferição de pré-embalados, permitindo, através de uma impressora externa própria, obter os dados estatísticos das últimas 45 pesagens. A figura 2 apresenta um exemplo de um talão impresso com as estatísticas das medições feitas.

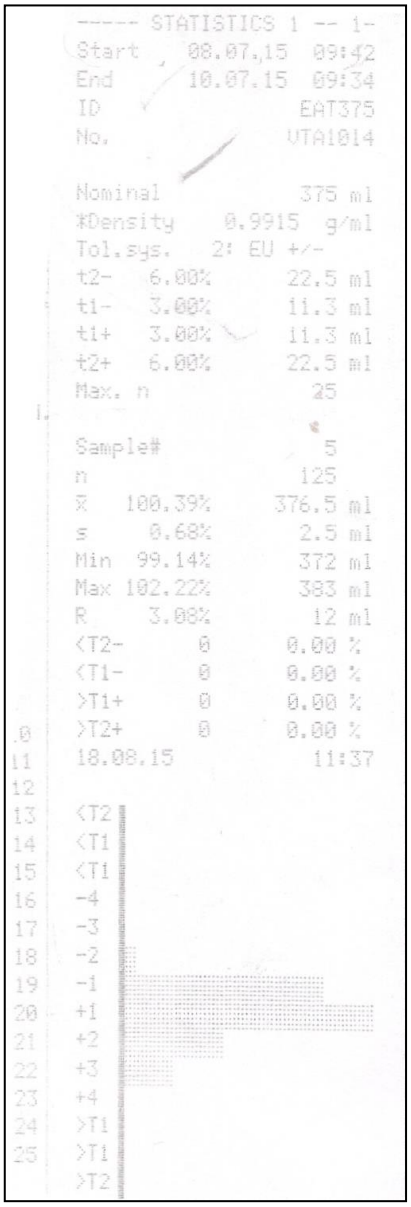


Figura 2 - Talão impresso por uma balança com dados estatísticos sobre pesagens realizadas

Este método, além de bastante limitado em termos do tamanho de amostra e das estatísticas mostradas, obriga ao armazenamento dos talões impressos, tornando-se assim mais difícil a procura pela informação necessária, uma vez que os talões estão em formato de papel.

3.2.2. Microsoft Excel

Outra das alternativas conhecidas é o uso da ferramenta *Microsoft Excel*. Por norma, o uso desta ferramenta passa pela medição/pesagem de um produto e pela introdução do valor observado no *Excel*. Existe também a possibilidade de importar os valores diretamente da balança para o *Excel*, tornando o processo relativamente mais rápido uma vez que não é necessário introduzir os dados manualmente.

Os ficheiros *Excel* variam entre as empresas, sendo que as funcionalidades implementadas nestes ficheiros (como por exemplo gráficos, cálculo de estatísticas, etc.) dependem das necessidades das mesmas.

As vantagens do uso desta ferramenta são o armazenamento digital (sendo mais fácil armazenar e procurar dados) e o facto de ter custos de manutenção reduzidos.

3.2.3. ACCEPT

ACCEPT é uma aplicação para controlo e gestão da produção das fábricas desenvolvida pela Sinmetro na linguagem de programação Delphi, focada na metrologia de pré-embalados. Esta aplicação está disponível para o sistema operativo *Windows* e responde às várias necessidades detetadas no chão de fábrica[1], permitindo alertar os operadores de situações não conformes e antecipá-las através da identificação de tendências ou padrões das linhas de produção. O ACCEPT é atualmente usado por diversas empresas [26] de vários setores da indústria (água, café, refrigerantes, vinho, etc.).

Os principais objetivos a cumprir pelo sistema ACCEPT são:

- Garantir a qualidade da produção em todas as fases do processo;
- Maximizar a capacidade das linhas de produção;
- Reduzir custos e não conformidades;

[1] Local onde é feita a produção de uma empresa

- Desmaterializar processos e libertar recursos humanos para outras tarefas;
- Fornecer um painel de indicadores de desempenho que avaliam a eficiência e eficácia das operações;
- Integrar informação, facilitando a análise de correlações dificilmente detetáveis no circuito atual em papel e bases de dados independentes.

Esta aplicação possui várias características tais como:

- Totalmente em português;
- Tratamento estatístico adaptado às necessidades específicas de cada perfil de utilizador;
- Adaptável à realidade de cada cliente;
- Implementação modular (ver figura 3);
- Integração com outros Sistemas de Informação e/ou equipamentos de medição para aquisição de dados.

O ACCEPT é constituído por 6 módulos distintos que estão representados na figura 3.

Figura 3 - Módulos do sistema ACCEPT. Imagem retirada do *website* da Sinmetro [25]

Todos os módulos do ACCEPT podem ser adquiridos individualmente conforme as necessidades do cliente. Na tabela 1 é feita uma descrição dos 6 módulos do ACCEPT descrevendo as suas vantagens e benefícios de acordo com a Sinmetro.

Módulo ACCEPT	**Objetivo**	**Vantagens/Benefícios**
analytics	Definição e monitorização de indicadores em tempo real	• Configurar os indicadores a monitorizar; • Visualizar numa única janela os indicadores definidos; • Agregar informação dos diferentes módulos do sistema ACCEPT; • Comparar indicadores, verificar se os objetivos estão a ser atingidos e detetar as áreas onde o desempenho está abaixo do esperado.
gml	Controlo estatístico da quantidade em pré-embalados	• Evitar enchimento de quantidade por excesso; • Garantir o cumprimento dos critérios legais dos produtos pré-embalados; • Tratamento estatístico, análise de dados e consequente análise de desvios; • Aquisição direta dos sistemas de medição.
sqc	Controlo estatístico da qualidade	• Registo de análises e validação do cumprimento das especificações de cada característica; • Garantia da rastreabilidade das análises efetuadas; • Tratamento estatístico, análise de dados e consequente análise de desvios; • Aquisição de dados diretamente dos equipamentos usados no controlo da qualidade; • Facilidade de utilização e de pesquisa, e impressão de relatórios à medida.

oee	Controlo da eficácia das linhas de produção.	• Cálculo em tempo real do indicador *Overall Equipment Effectiveness*[2] (OEE), para uma ou várias linhas de produção; • Registo de tempos de funcionamento e de paragem para cálculo do indicador da disponibilidade; • Registo das quantidades produzidas e rejeitadas para cálculo do indicador da qualidade; • Comparação da velocidade padrão com a velocidade de produção para cálculo do indicador de velocidade; • Análise estatística e gráfica dos OEE's e respetivos indicadores de disponibilidade, velocidade e qualidade.
micro	Rastreabilidade total das amostras laboratoriais	• Simplificar o processo laboratorial, reduzindo o tempo de recolha, identificação, preparação e registo das análises microbiológicas; • Garantir a rastreabilidade, de forma simples e clara, das amostras recolhidas na fábrica, ao longo de todas as fases da análise microbiológica; • Minimizar o número de erros na identificação de amostras; • Alertar o operador caso as leituras dos resultados sejam realizadas fora dos prazos definidos.
mobile	Registo de dados em mobilidade	• Facilitar a recolha de dados em ambientes onde o uso de um PC não é possível; • Alertar o operador para valores não conformes; • Evitar a redundância de dados, eliminando registos em papel; • Reduzir erros do operador; • Integração direta com o módulo ACCEPT SQC.

Tabela 2 - Descrição dos módulos do ACCEPT. Adaptado do *website* da Sinmetro [25]

[2] Métrica de eficácia de um equipamento em relação à sua capacidade projetada, durante o(s) período(s) esperado(s) [52]

3.2.3.1 – Arquitetura ACCEPT

A Figura 4 apresenta a arquitetura tradicional do ACCEPT em funcionamento numa fábrica.

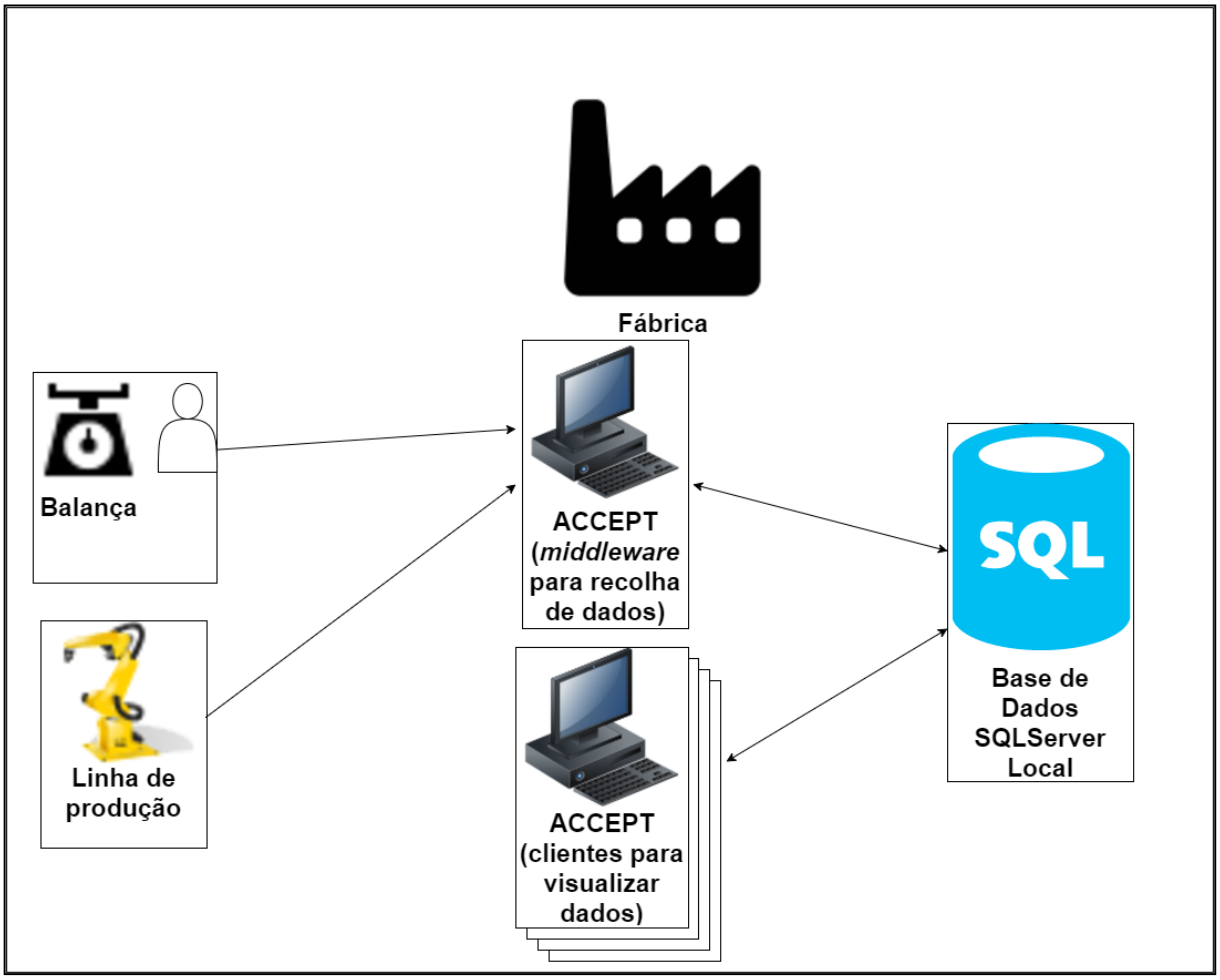


Figura 4- Arquitetura do ACCEPT em funcionamento numa fábrica

Tal como evidenciado na figura anterior, a recolha dos dados das pesagens na fábrica pode ser realizado de dois métodos distintos, através da pesagem manual utilizando uma balança ligada ao computador onde está a correr o ACCEPT, ou através de sensores colocados ao longo das linhas de produção.

A introdução de dados através de uma balança diretamente ligada ao computador onde está a correr o *software* ACCEPT utiliza a interface (ou portos) adequada a cada tipo de balança. Após cada pesagem, o ACCEPT atualiza a base de dados bem como as estatísticas e gráficos com base nessa informação. Deste modo o utilizador tem a perceção da evolução do processo de enchimento ao longo da amostragem.

O outro método de aquisição de dados passa pela receção das pesagens enviadas por sensores de peso presentes ao longo das linhas de produção. O número de sensores existentes em cada linha de produção depende da fábrica e dos artigos a ser produzidos. Em alguns casos são usados sensores para determinar o peso da embalagem vazia, e posteriormente, são usados outros sensores colocados mais à frente na linha de produção para determinar o peso do produto após o seu enchimento. Desta forma é automaticamente calculada a massa líquida do produto tendo em conta o seu peso antes e depois do enchimento. Noutros casos é utilizado um sensor no final da linha de produção para determinar a massa bruta do produto, fazendo uso de uma tara média previamente calculada para determinar a massa líquida. Para algumas empresas este método pode não ser praticável (por questões monetárias ou outras), utilizando assim o método manual.

Todos os dados referentes às pesagens e estatísticas são guardadas numa base de dados Microsoft SQL (*Structured Query Language*) *Server*. Estes dados são recebidos num computador onde está a correr o ACCEPT e guardados na base de dados, servindo o computador como *middleware* entre as balança e a base de dados. Para visualizar os dados estatísticos (gráficos, tabelas, etc.) poderão existir também outros computadores na fábrica dotados do módulo do ACCEPT específico para o efeito.

4 – Implementação

Nesta secção é descrita a implementação do ACCEPT Web. Em primeiro lugar são apresentados os objetivos e a metodologia de desenvolvimento usada. De seguida é apresentada a arquitetura do projeto, seguido das várias tecnologias utlizadas no desenvolvimento do ACCEPT Web.

4.1 Objetivo e Metodologia

Tal como descrito no capítulo 2.2, o objetivo do estágio é transportar as várias funcionalidades existentes na versão *desktop* do ACCEPT para um ambiente *Web*, para que as empresas tenham acesso aos dados das suas linhas de produção em qualquer lado e em qualquer altura.

Aquando do início do estágio estavam já definidas algumas condições para a realização do projeto. Na fase inicial, foram definidos alguns requisitos de alto nível aos quais a aplicação deveria responder:

- A aplicação deverá ser acessível através da *Web* e compatível com dispositivos móveis e computadores;
- Deverá ser usada uma metodologia de desenvolvimento adequada ao trabalho a realizar;
- Deverá ser utilizado um *template* anteriormente adquirido pela Sinmetro como base visual e funcional para o projeto. Quando necessário, poderão ser criados novos elementos ou vistas na aplicação tentando sempre manter a coerência visual com o resto da aplicação;
- Para o servidor PHP, deverá ser criado um projeto utilizando uma *framework* MVC escolhida pelos colaboradores que integram o projeto;

- Durante o desenvolvimento, o servidor deverá estar alojado num computador da Sinmetro, sendo este acedido através da rede local. O *software* onde o servidor estará alojado deverá ser o WAMP.
- A troca de dados entre o servidor e a base de dados deverá ser feita através duma API REST desenvolvida pela Sinmetro;
- Não existem restrições do uso de *plugins*, pacotes ou similares tanto para o *front-end* como para o *back-end*, desde que estes permitam a sua utilização para fins comerciais. A integração final destes componentes na aplicação está dependente da aprovação do responsável do projeto, tendo em conta o seu aspeto visual e funcional;
- A aplicação deverá ter suporte às linguagens portuguesa e inglesa;
- Para a gestão das funcionalidades do projeto, deverá ser usada a aplicação *online* Asana.

Ao longo do desenvolvimento projeto foram tidos em conta estes requisitos, de forma a manter uma estrutura coerente e escalável.

Em primeiro lugar existia já criado um *template* HTML com o visual que se pretendia manter na aplicação. Este *template* possuía também alguns exemplos de bibliotecas ou *plugins* de jQuery que foram aproveitadas para o produto final, como por exemplo: gráficos, seletor de datas, tabelas, entre outros, sendo que todos estes casos utilizavam dados estáticos apenas para testes. Foi também definido que apesar de ser a base para o desenvolvimento do projeto, o visual e os *plugins* utilizados poderiam sofrer alterações conforme as necessidades que fossem encontradas durante o desenvolvimento. Este *template* foi adquirido pela Sinmetro à empresa MediaWeb Creations Lda [35]. A figura 5 apresenta o *dashboard* base presente no *template* fornecido.

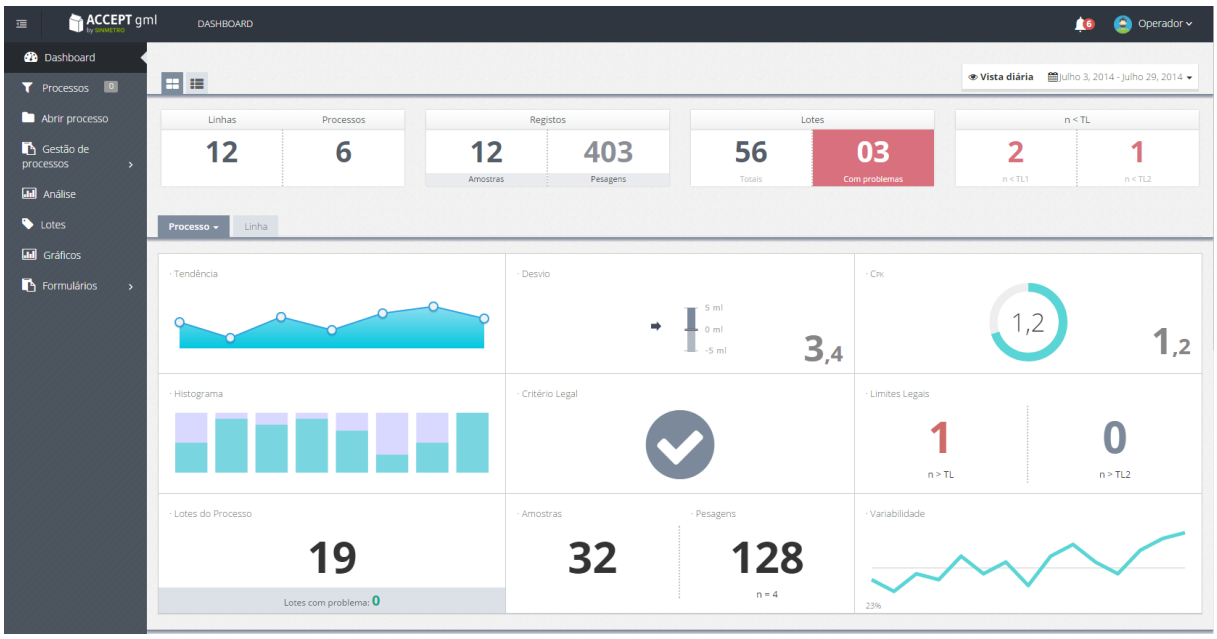


Figura 5 - Aspeto inicial do *dashboard* do *template* fornecido

Para auxiliar no desenvolvimento e estruturação do código da aplicação no servidor foi também definido nas condições iniciais do estágio que deveria ser utilizada uma *framework* MVC (*Model-View-Controller*), ficando esta escolha ao cargo dos colaboradores envolvidos no projeto. Tendo em conta as opções disponíveis, foi escolhida a *framework* Laravel uma vez que já existiam, entre os colaboradores do projeto, as competências necessárias para a utilização da mesma. No capítulo 4.3.1 está disponível uma introdução a esta *framework*.

Quanto à recolha de dados, em vez de usar uma base de dados relacional foi definido que deveria ser utilizada uma API REST já existente, desenvolvida pela Sinmetro. Numa fase inicial estavam apenas disponíveis alguns métodos, sendo os restantes adicionados ao longo do tempo conforme o trabalho fosse sendo desenvolvido.

Quanto às tecnologias de "front-end" também não foi imposta nenhuma restrição ao uso de *frameworks*. A escolha do uso de ferramentas adicionais dependeria das necessidades que fossem encontradas durante o desenvolvimento do projeto, podendo para isso ser utilizadas quaisquer *frameworks* que se considerassem úteis e vantajosas. As *frameworks* de *javascript* usadas estão descritas entre os capítulos 4.3.6 e 4.3.13.

Outra das funcionalidades definidas para o projeto foi o suporte multilingue. Inicialmente foi apenas identificada a necessidade do suporte à língua portuguesa e inglesa, podendo mais

tarde ser adicionadas outras linguagens, mas já fora do âmbito do estágio. As ferramentas utilizadas para o suporte multilingue foram as de suporte nativo do Laravel (para PHP) e o jQuery-localize e estão descritas nos capítulos 4.1.3.2 e 4.3.11, respetivamente.

Quanto à gestão das funcionalidades a implementar no projeto, foi utilizada a aplicação *online* Asana para gerir o estado do desenvolvimento das funcionalidades. Inicialmente o responsável pelo projeto criava uma tarefa, que era disponibilizada aos intervenientes do ACCEPT Web. Essa tarefa era depois analisada e dava-se início ao desenvolvimento dessa funcionalidade. A plataforma do Asana permite também trocar mensagens e imagens, sendo esta usada para tirar dúvidas quando necessário e registar os progressos feitos nas funcionalidades.

Nos primeiros 4 meses do projeto foram também documentadas algumas das atividades realizadas no dia-a-dia e as dificuldades sentidas, de forma a se ter uma ideia da evolução do trabalho realizado ao longo do tempo. Este documento pode ser consultado no apêndice A.

4.1.1 – Planeamento

Uma vez que no início do estágio não havia uma especificação detalhada para cada funcionalidade ou de todas as funcionalidades a implementar (apesar de haver uma ideia geral do produto final), ficou estabelecido que a implementação seria feita conforme fosse sendo estabelecida a ordem de prioridades das funcionalidades. O planeamento de tarefas para a duração do estágio foi realizado num panorama geral de *front-end*, *back-end*, testes, implementação, etc. Era também expectável que a implementação de *back-end* e *front-end* fosse feita consoante a necessidade detetada, principalmente para testar a validade e funcionalidade da apresentação de dados e dos métodos da API.

O planeamento para o cumprimento dos requisitos do estágio é apresentado na figura 5, considerando que o estágio decorreu entre 1 de setembro de 2015 e 31 de maio de 2016.

	Nome da tarefa	Duração	Início	Término
1	Escrita do relatório de estágio	292 dias	Ter 15/09/15	Dom 31/07/16
2	**Reuniões Semanais**	**198 dias**	**Ter 24/11/15**	**Ter 28/06/16**
35	**Projeto**	**262 dias**	**Ter 01/09/15**	**Ter 31/05/16**
36	Apresentação à equipa	1 dia	Ter 01/09/15	Ter 01/09/15
37	**Formação**	**38 dias**	**Ter 01/09/15**	**Qui 08/10/15**
38	Conceitos de qualidade/metrologia/estatistica	2 dias	Ter 01/09/15	Qua 02/09/15
39	Controlo metrológico de pré-embalados	1 dia	Qui 08/10/15	Qui 08/10/15
40	**Testes Iniciais**	**7 dias**	**Qui 03/09/15**	**Qua 09/09/15**
41	Estudo inicial de ferramentas/aplicações/frameworks a utilizar	2 dias	Qui 03/09/15	Sex 04/09/15
42	Criação de um novo projeto PHP com a framework escolhida	1 dia	Seg 07/09/15	Seg 07/09/15
43	Ligação à API REST da Sinmetro	2 dias	Ter 08/09/15	Qua 09/09/15
44	**Integração entre o template e o projeto PHP**	**7 dias**	**Qui 10/09/15**	**Qua 16/09/15**
45	Estudo das aplicações e plugins do template	2 dias	Qui 10/09/15	Sex 11/09/15
46	Integração das vistas do template no projeto PHP	3 dias	Seg 14/09/15	Qua 16/09/15
47	**Implementação de funcionalidades de back-end necessárias numa fase inicial**	**5 dias**	**Qui 17/09/15**	**Seg 21/09/15**
48	Autenticação/ outros	5 dias	Qui 17/09/15	Seg 21/09/15
49	**Implementação do front-end**	**135 dias**	**Qui 17/09/15**	**Sex 29/01/16**
50	Estudo de plugins e aplicações necessárias	135 dias	Qui 17/09/15	Sex 29/01/16
51	Implementação das funcionalidades	135 dias	Qui 17/09/15	Sex 29/01/16
52	Testes e correções	135 dias	Qui 17/09/15	Sex 29/01/16
53	**Implementação do back-end**	**107 dias**	**Seg 01/02/16**	**Sex 27/05/16**
54	Estudo de plugins e aplicações necessárias	107 dias	Seg 01/02/16	Sex 27/05/16
55	Implementação das funcionalidades	107 dias	Seg 01/02/16	Sex 27/05/16
56	Testes e correções	107 dias	Seg 01/02/16	Sex 27/05/16
57	**Testes em clientes Sinmetro**	**8 dias**	**Qua 30/03/16**	**Qua 06/04/16**
58	Deslocação a cliente(s) Sinmetro e testes	1 dia	Qua 30/03/16	Qua 30/03/16
59	Correções com base nos testes	8 dias	Qua 30/03/16	Qua 06/04/16
60	Apresentação interna do projeto à empresa	1 dia	Seg 30/05/16	Seg 30/05/16
61	Disponibilização do projeto na Web	2 dias	Seg 30/05/16	Ter 31/05/16

Figura 6 - Planeamento de tarefas para o estágio

4.1.2 – Metodologia de Desenvolvimento

A metodologia de desenvolvimento utilizada durante o desenvolvimento do projeto foi o *Software Prototyping*. Esta metodologia pode ser dividida em 4 subtipos, sendo que dentro destes foram seguidas as especificações do *Software Prototyping*. Esta secção apresenta um estudo sobre a metodologia usada e os seus subtipos.

O *Software Prototyping* é uma metodologia de desenvolvimento que tem por objetivo o rápido desenvolvimento de *software* para testar a validade dos requisitos [36]. Esta metodologia centra-se na criação de protótipos para testar as várias funcionalidades esperadas e segue os seguintes passos [37]:

1. Identificar os requisitos básicos em termos de funcionalidade e interface;
2. Desenvolver o protótipo inicial;

3. Examinar o protótipo e recolher informação sobre o seu estado por parte do cliente;
4. Com base no parecer do passo 3, o protótipo pode ser dado como terminado ou poderá voltar à fase de implementação até que o resultado final corresponda às expectativas do cliente.

Esta metodologia ajuda os clientes e os criadores do sistema a entender os requisitos do mesmo. Através do uso dos protótipos, o utilizador final do sistema consegue saber de que forma este funcionará, assim como detetar falhas ou erros que poderão não ter sido encontrados na fase de requisitos. Isto permite uma redução de riscos uma vez que uma grande parte das falhas pode ser encontrada numa fase antecipada do projeto.

Alguns dos benefícios do uso do *Software Prototyping* são [36]:

- Melhor usabilidade do sistema através da identificação de falhas numa fase inicial do desenvolvimento;
- Melhor correspondência em relação ao resultado final pretendido;
- Melhor interface através da identificação de elementos visuais pretendidos ou não pretendidos;
- Esforço de desenvolvimento do *software* reduzido.

Poderão no entanto ser também identificadas algumas desvantagens desta metodologia tais como:

- Demasiada dependência nos protótipos havendo riscos de incoerências e/ou erros caso o levantamento de requisitos seja defeituoso;
- A complexidade do sistema pode aumentar devido às mudanças dos requisitos;
- O esforço investido na criação dos protótipos poderá ser demasiado alto caso não haja monitorização;
- Os utilizadores finais poderão ficar confusos com o estado dos protótipos.

Os tipos de protótipos desenvolvidos podem geralmente seguir 2 dimensões [37]. A dimensão horizontal é mais orientada para a componente visual do *software*, procurando dar uma vista mais alargada de todo o aspeto visual que produto final irá tomar. Estes protótipos podem também ser usados, por exemplo, como demonstrações a nível empresarial de forma a encontrar potenciais clientes ou um eventual financiamento. Já a dimensão vertical serve para elaborar uma função num subsistema mais específico do produto. Com os protótipos verticais pretende-se obter detalhes exatos sobre uma funcionalidade de forma adquirir dados sobre os

seus requisitos, como por exemplo para a criação da base de dados, interceção entre sistemas, processamento de dados, *inputs* e *outputs*, entre outros.

O *Software Prototyping* pode ser divido em 4 tipos, *Throwaway Prototyping, Evolutionary Prototyping, Incremental Prototyping e Extreme Prototyping*, que serão analisados de seguida.

4.1.2.1 *Throwaway Prototyping*

Throwaway Prototyping [38] refere-se à criação de protótipos incompletos de forma rápida que eventualmente serão descartados em vez de fazerem parte do produto final. Depois de serem recolhidos os requisitos preliminares é construído um modelo funcional de forma a mostrar ao cliente a aparência final esperada.

O aspeto mais importante deste método de prototipagem é a rápida criação de modelos funcionais, podendo desde muito cedo validar os requisitos e a interface do sistema. Isto permite evitar custos e tempo em eventuais mudanças no sistema bem como tratar alguns dos problemas relacionados com a identificação dos requisitos.

4.1.2.2 *Evolutionary Prototyping*

O objetivo do *Evolutionary Prototyping* é criar um protótipo robusto que servirá como base para todos os outros protótipos desenvolvidos [39]. Este protótipo inicial comtempla apenas os requisitos iniciais mais importantes e que estão bem definidos, sendo este depois melhorado à medida que vão sendo definidos e compreendidos novos requisitos.

Esta metodologia permite ir adicionando e mudando funcionalidades que não são possíveis realizar numa fase inicial do desenvolvimento, seja por os requisitos não estarem bem definidos ou compreendidos ou por não estarem desenvolvidas outras funcionalidades das quais estas dependem. Assim, o protótipo inicial vai sofrendo alterações e evoluindo ao longo do desenvolvimento, evitando-se riscos causados pelo fraco conhecimento dos requisitos.

4.1.2.3 *Incremental Prototyping*

Nesta metodologia são criados protótipos de forma separada e independente para as várias funcionalidades do sistema [40]. Após o desenvolvimento dos protótipos, estes são, ou vão sendo integrados entre si de forma a criar o produto final.

Uma vantagem deste método é que o cliente e /ou utilizadores finais têm a oportunidade de testar os componentes desenvolvidos e sua funcionalidade, podendo também fornecer *feedback*, enquanto outros componentes ainda estão em desenvolvimento influenciando assim o produto final.

4.1.2.4 *Extreme Prototyping*

O *Extreme Prototyping* é uma metodologia utilizada para a criação de aplicações, especialmente orientada para aplicações *Web* [41]. Esta metodologia divide o desenvolvimento de uma aplicação *Web* em essencialmente 3 passos:

1 Criação de páginas estáticas apenas com elementos HTML;

2 Programação dos ecrãs e dos elementos HTML utilizando dados simulados;

3 Implementação dos serviços e funcionalidades e integração com a interface.

O objetivo desta metodologia é focar o segundo passo, desenvolvendo uma página completamente funcional antes de implementar os serviços. Após a página estar completamente funcional é possível obter *feedback* sobre o seu estado e realizar eventuais mudanças. Além disso, isto permite que as equipas de desenvolvimento se dediquem às suas tarefas separadamente, e tomar decisões antecipadamente para garantir uma integração com sucesso entre as páginas e os serviços.

Esta foi a metodologia usado durante o desenvolvimento do ACCEPT Web. O uso desta metodologia deveu-se ao facto de, dentro do *Sofware Prototyping*, ser o que mais de adequa à criação de aplicações *Web*. No caso do ACCEPT Web, as tarefas a realizar eram bastante orientadas à representação de dados, pelo que geralmente requeriam a criação de novas páginas ou a alteração de outras páginas previamente existentes (geralmente provenientes do *template*).

Em termos práticos, a criação das tarefas era feita pela ordem preferencial da Sinmetro. Aquando do início de cada tarefa eram discutidos quais as funcionalidades, *inputs/outputs* e aspeto visual da janela. Em termos de *plugins* para a representação de dados, tentava-se

sempre que possível utilizar os que já estavam incluídos no projeto base, uma vez que apresentavam o aspeto visual desejado inicialmente. Quando os *plugins* não existiam, a sua utilização não era possível, ou não apresentavam o resultado esperado (uma vez que a introdução de dados mudaria o seu aspeto), eram pesquisados e experimentados outros *plugins* utilizando dados estáticos. À medida que se iam experimentando, eram escolhidos os que apresentavam o resultado mais aproximado do esperado. Após a aprovação, as extensões eram então integradas no produto final utilizando dados dinâmicos.

Em suma, o processo de desenvolvimento está representado na figura 7 e seguia os seguintes passos:

1. A Sinmetro criava a tarefa e atribuía-a ao colaborador (estagiário);
2. Era feito um planeamento da tarefa e estudados os i*nputs* e *outputs* necessários;
3. Caso fossem necessários *plugins* e/ou extensões externas, eram estudadas e testadas várias possibilidades existentes, caso contrário eram utilizadas as alternativas já existentes no *template*;
4. Caso fosse necessário desenhar uma página nova ou redesenhada uma página já existente no *template*, eram feitas as alterações necessárias. Após as alterações era feita uma revisão por parte da Sinmetro. Caso o resultado fosse satisfatório avançava-se para a fase de desenvolvimento, caso contrário eram discutidas e feitas as mudanças necessárias até se chegar ao resultado pretendido;
5. Com a página criada, eram implementadas as funcionalidades utilizando os *plugins* escolhidos (caso fosse necessário) fazendo uso de dados estáticos para efeitos de teste;
6. Era novamente feita uma revisão por parte da Sinmetro e feitas as alterações necessárias até se chegar ao resultado esperado;
7. Era implementado o código necessário para carregar dados dinâmicos da API e/ou da base de dados;
8. Após a funcionalidade estar implementada, era feita uma análise final para identificar as alterações necessárias e proceder às respetivas correções.
9. Após eventuais correções a tarefa era dada como terminada, passando-se à tarefa seguinte.

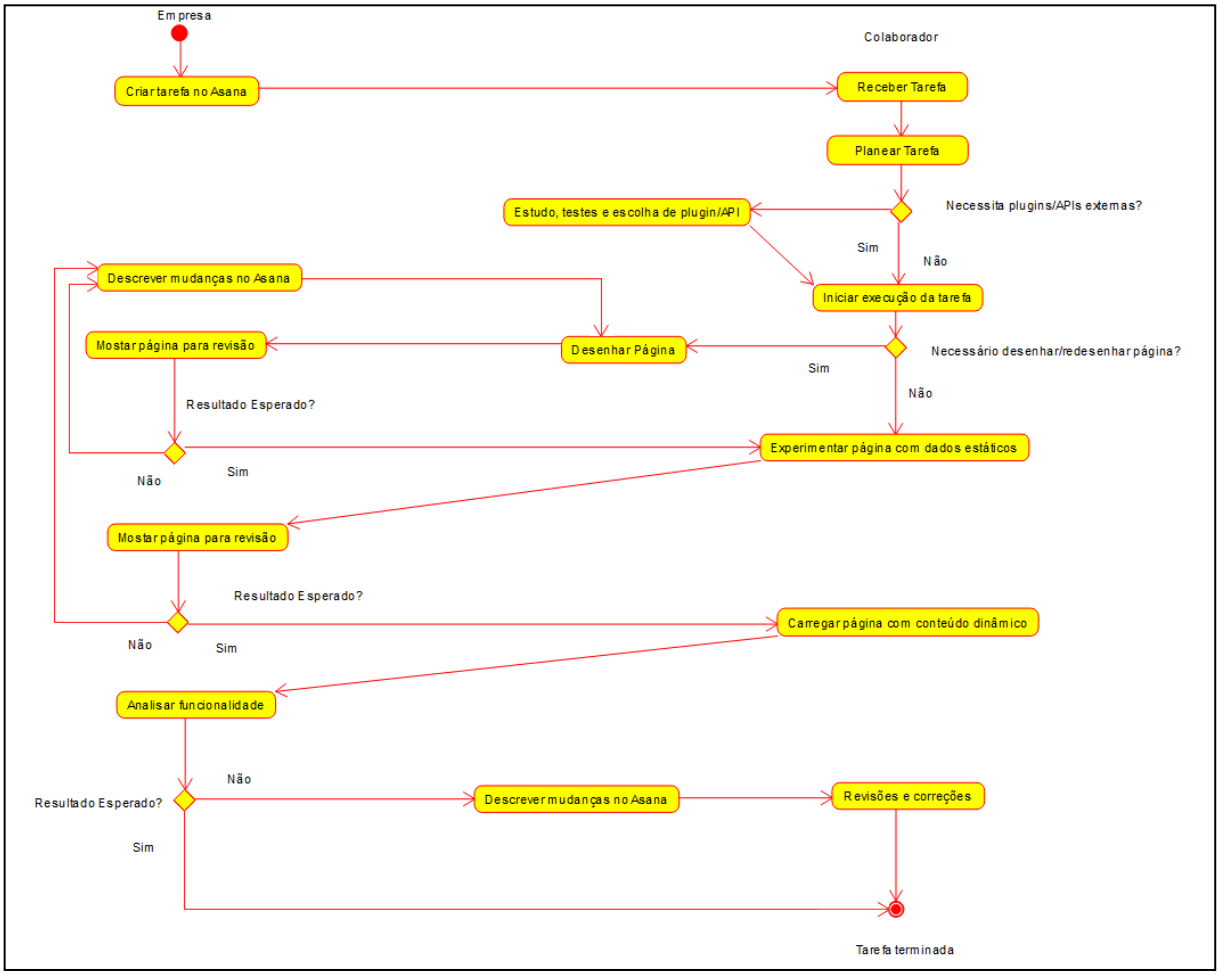


Figura 7 - Diagrama de atividades das tarefas a realizar

4.2 Arquitetura

Antes de começar a implementação da aplicação foi necessário pensar na arquitetura que o ACCEPT Web iria seguir de forma a perceber quais os componentes que faziam ou iriam fazer parte do projeto, bem como quais as suas dependências e como iriam comunicar entre si.

Assim, o sistema ACCEPT Web é composto pelos seguintes componentes:

- Clientes (*browsers*);
- Um servidor PHP desenvolvido através da *framework* Laravel;
- Uma API REST que serve de *middleware* entre o servidor PHP e a base de dados e que contém os métodos usados pelo servidor;
- Uma base de dados Microsoft *SQLServer* onde estão guardados os dados;
- Um cliente ACCEPT existente numa fábrica para preencher a base de dados.

A figura 8 apresenta a arquitetura geral do ACCEPT Web.

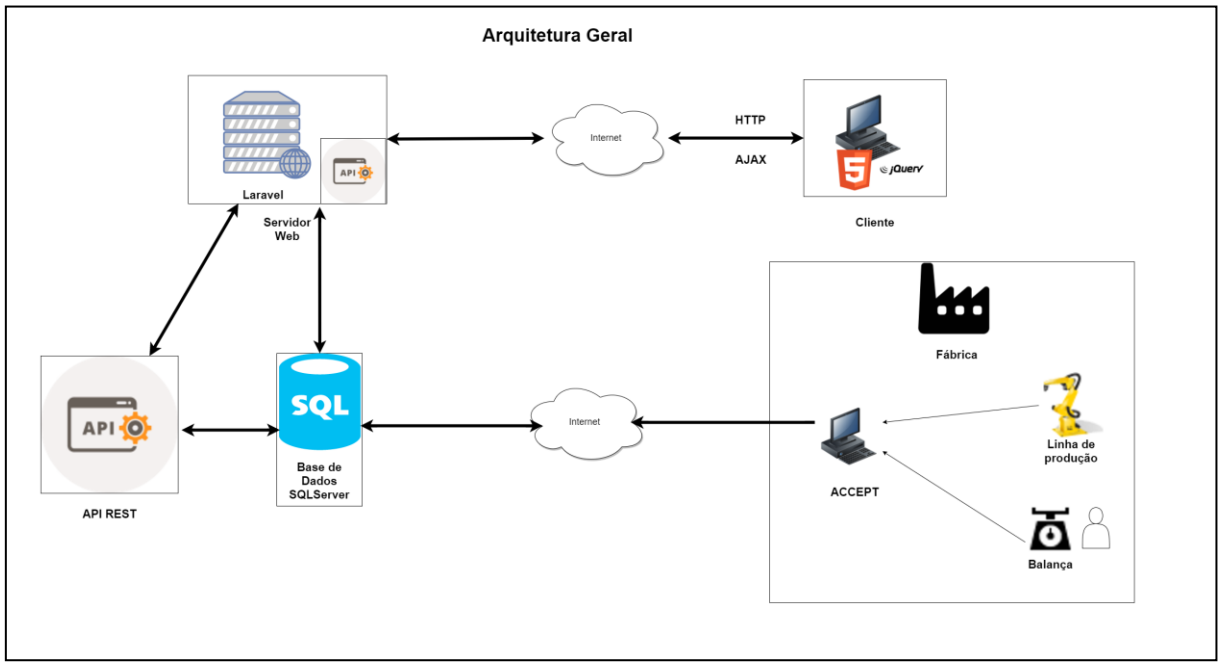


Figura 8 - Arquitetura geral do ACCEPT Web

Esta arquitetura engloba um conjunto de componentes responsáveis pelo processo decorrido desde o preenchimento da base de dados até à visualização desses dados por parte do cliente. Quando o cliente (*browser*) realiza os pedidos ao servidor PHP, o servidor usa uma conexão com a API REST para adquirir os dados da base de dados e devolve-los ao cliente. Existem também situações em que a conexão à base de dados é feita diretamente pelo servidor sem recorrer à API.

O servidor PHP inclui vários métodos para comunicação com o cliente e a API REST ou a base de dados. Após a receção de um pedido HTTP (*Hypertext Transfer Protocol*) por parte do cliente, o servidor utiliza um mecanismo de rotas para converter o URL num caminho para um método interno. A execução destes métodos realiza sempre um acesso à base de dados, exceto para a apresentação da página de login.

A figura 9 apresenta a arquitetura detalhada do servidor Laravel, onde está representado o processo de comunicação entre o servidor e o cliente após um pedido HTTP.

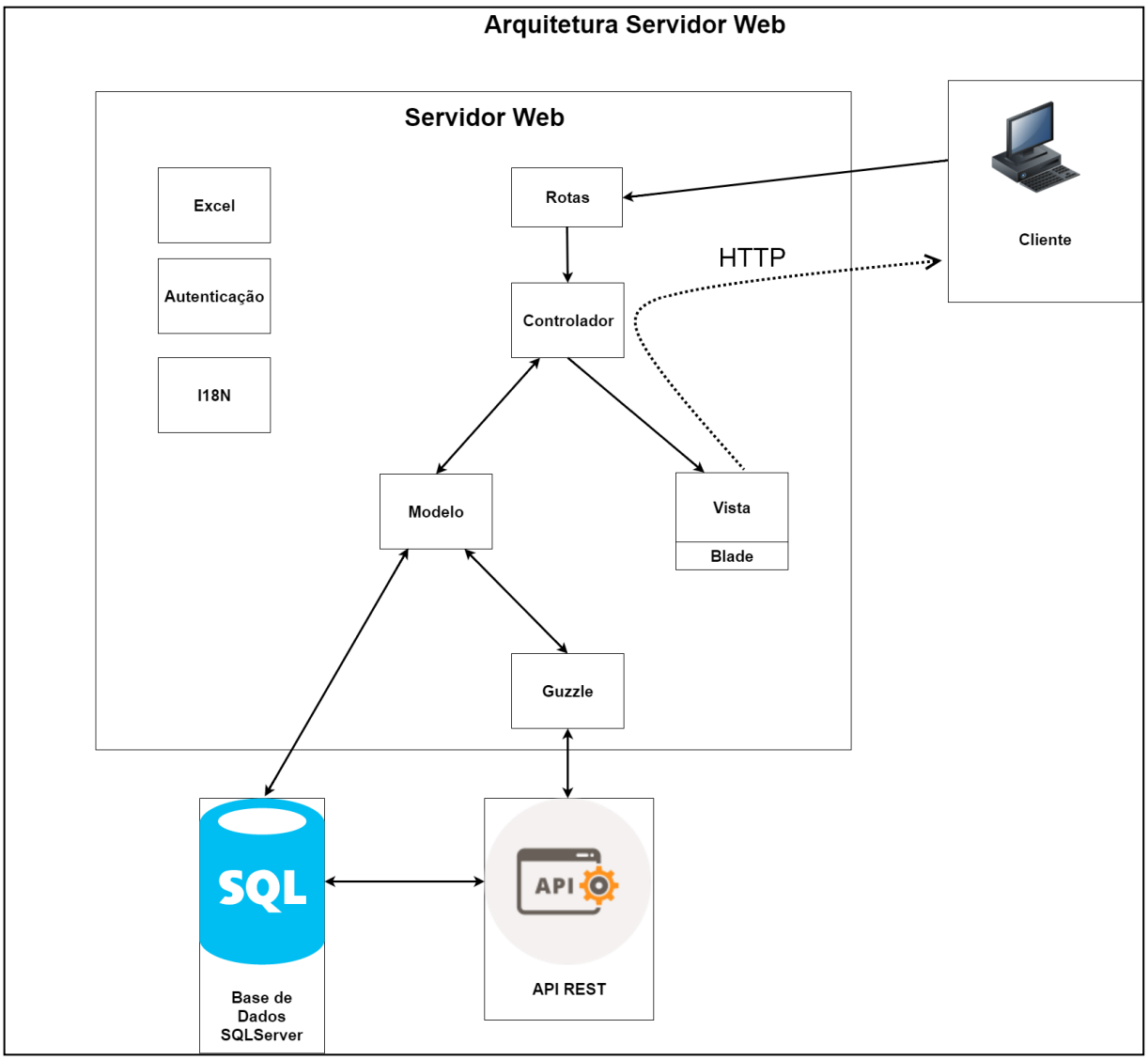


Figura 9 - Arquitetura do servidor Laravel

Por exemplo, para consumir métodos da API, o controlador chamado pelo sistema de rotas irá invocar os métodos necessários para realizar uma conexão à API através do Guzzle (ver capítulo 4.3.2). Após receber a resposta do modelo, o controlador utiliza os dados, se for necessário, para preencher uma vista HTML com recurso ao Blade (ver capítulo 4.3.1.2) e envia a página resultante para o cliente.

Sendo uma ferramenta maioritariamente de análise estatística, o ACCEPT Web recorre frequentemente a instrumentos visuais como grelhas de dados e gráficos. Apesar de existirem várias páginas no ACCEPT Web, todas elas são incorporadas numa única página *web*, de acordo com o padrão arquitetural SPA *(Single Page Application*) uma vez que os dados (gráficos, grelhas, etc.) são sempre carregados através de AJAX (*Asynchronous Javascript*

and XML) sem ser necessário recarregar a página. A figura 10 apresenta um esboço do processo de criação de uma página HTML e da sua atualização parcial através de AJAX.

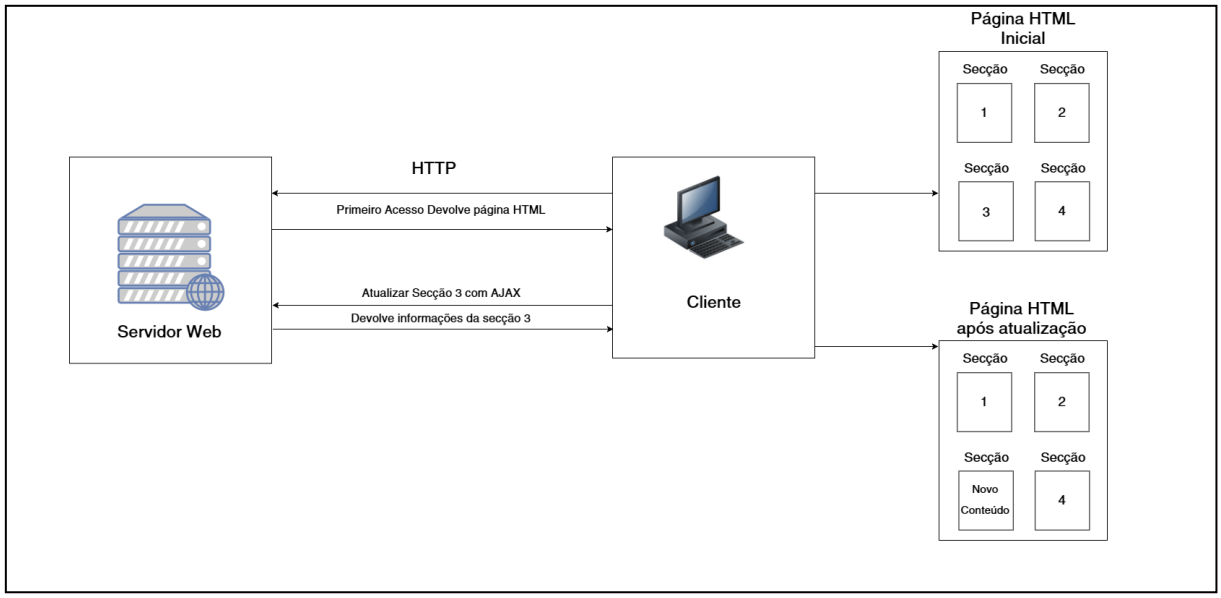


Figura 10 - Esboço da atualização parcial de uma página através de AJAX

A atualização das páginas de forma parcial ocorre por pedido do cliente, geralmente após a especificação pelo utilizador do intervalo de tempo dos dados a visualizar. Ao fazer o pedido é realizada uma chamada AJAX assíncrona e é recebido um objeto JSON (*JavaScript Object Notation*) que é usado para atualizar uma parte da página HTML. O excerto de código fonte 1 apresenta um exemplo de uma chamada AJAX que é executada quando o utilizador muda o intervalo de tempo dos dados e realiza a pesquisa ao clicar num botão.

```
function getData(){

      //Data de início para a pesquisa. Ex:2010-01-01
      var startDate = $('startDate').val();
      //Data de fim para a pesquisa. Ex: 2015-12-31
      var endDate = $('#endDate').val();

      $.ajax({
            type: 'GET',//Método AJAX (GET ou POST)
            url: 'getProcessos',//Nome da rota Laravel
            data:{
                  //Variaveis a enviar para o servidor
                  chaveProcesso:'',
                  startDate: startDate,
                  endDate, endDate
            },
            success: function(data){
                  //Mudar os dados pretendidos
                  //Ex: redesenhar tabelas, gráficos, texto, etc.
                  changeInfo(data);
            },
            error: function(){
```

```
                    //Apresentar mensagem de erro
                }
        });
}
```

Excerto de Código Fonte 1 - Exemplo de chamada AJAX com 3 variáveis

No exemplo de código fonte anterior, ao chamar o método "getData" é feito um pedido AJAX para ir buscar todos os processos entre duas datas (por exemplo, entre 1 de janeiro de 2010 e 31 de dezembro de 2015). Utilizando o mecanismo de rotas do Laravel este pedido é depois redirecionado para o método "getProcessos" do controlador "ProcessoController".

```
//routes.php
Route::post('getProcessos', array('before' => 'isLoggedIn',
'as' => 'getProcessos',
'uses' => 'ProcessosController@getProcessos'));
```

```
//ProcessosController.php
public function getProcessos(){

        $startDate = \Input::get('startDate'); //variáveis passadas no AJAX
        $endDate = \Input::get('endDate');
        $chaveProcesso = \Input::get('chaveProcesso');

        //chamar o método "getProcessos" na API
        $results = $this->modelInstance-
>getProcessos($startDate,$endDate,$chaveProcesso);
        $results = json_decode($results);
        return $results;
}
```

Excerto de Código Fonte 2 - Exemplo do uso do mecanismo de rotas e de um acesso à API

Como evidenciado no excerto de código fonte 2, o método "getProcessos" do controlador faz um pedido à API, passando-lhe os dados recebidos pelo método GET. Após receber resposta devolve os dados recebidos para o cliente em formato JSON, que os utiliza para mudar o seu estado sem ter de recarregar a página por completo.

Para a recolha e envio de dados para o *front-office* foram utilizados os métodos disponibilizados na API desenvolvida pela Sinmetro. Já para a gestão do *back-office* a equipa de desenvolvimento optou por uma conexão direta à base de dados uma vez que a API não suportava os métodos necessários e não era expectável que estes fossem concluídos em tempo útil. Assim, no *back-office*, a comunicação com a base de dados foi feita diretamente através de comandos SQL.

4.3 Tecnologias de Desenvolvimento

Nesta secção são apresentadas as tecnologias (APIs, *frameworks*, extensões, bibliotecas, etc…) utilizadas durante o desenvolvimento do ACCEPT Web. Do capítulo 4.3.1 ao capítulo 4.3.5 são apresentadas tecnologias do lado do servidor, sendo estas o Laravel, Guzzle, Laravel Excel, laravel-crud-generator e CSV. São depois apresentadas tecnologias do lado do cliente relacionadas com *javascript* e CSS (*Cascading Style Sheets*), nomeadamente o jQuery, Date Range Picker e Moment.js, Datatables, Flot Charts, Circliful, jQuery-localize, Sortable, Javascript Cookie e Bootstrap.

4.3.1 Laravel

Para desenvolver o servidor PHP foi utilizada a *framework* Laravel [42]. Esta é uma *framework* PHP *open-source* [4] para o desenvolvimento de aplicações *Web* criada por Taylor Otwell concebida para a criação de sistemas que assentam sobre o padrão arquitetural MVC.

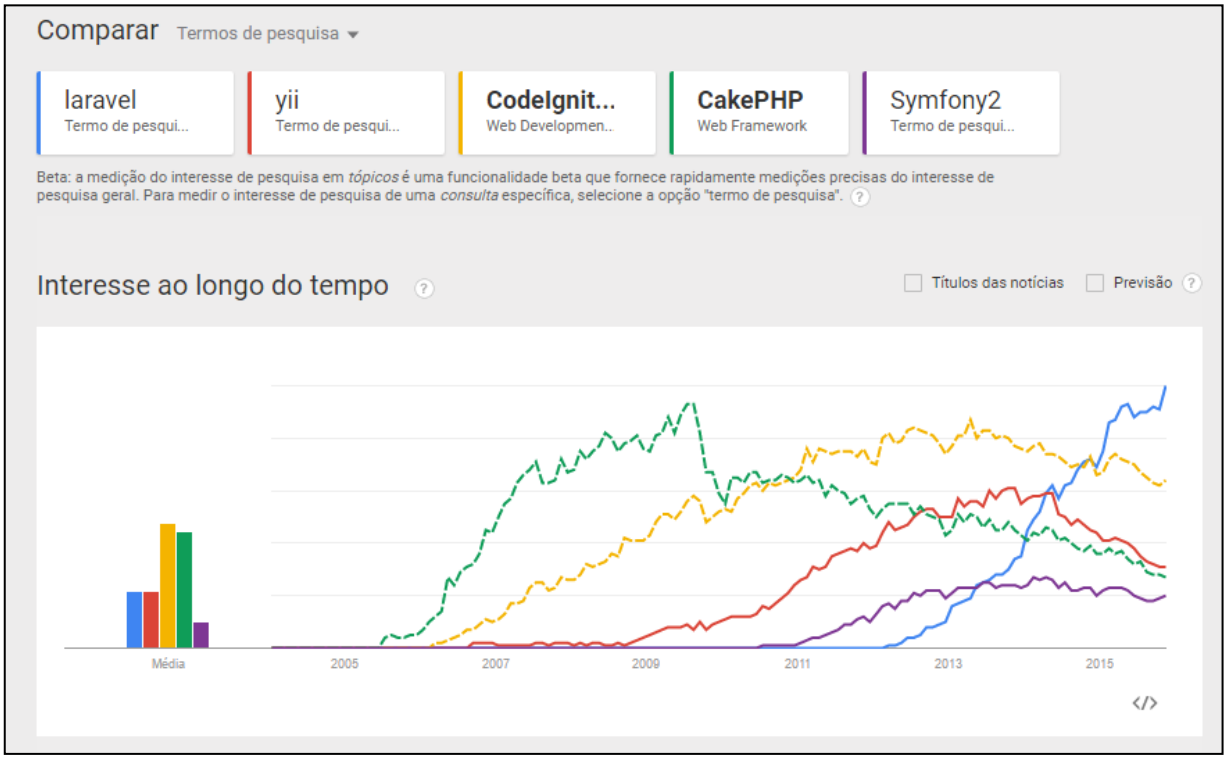


Figura 11 - Tendência de pesquisa de 5 *frameworks* PHP no motor de busca Google. Imagem gerada através do site Google Trends [6]

Apesar de existirem outras *frameworks* de PHP, Laravel apresenta um grande crescimento de popularidade desde o seu lançamento em 2011, tornando-se a *framework* com melhor pontuação atribuída pelos utilizadores do *site GitHub* durante a sua quarta versão em 2014 [5]. A figura 11 apresenta a tendência de pesquisas no motor de busca *Google* de algumas *frameworks* PHP desde outubro de 2005 até novembro de 2015.

Uma vez que a *framework* Laravel nasceu em 2011, é possível perceber pela imagem anterior que o crescimento do interesse dos programadores/engenheiros de *software* foi bastante rápido quando comparado às outras *frameworks*. Segundo um questionário feito pelo *site* SitePoint [21] que teve a duração de um mês e participação de cerca de 7800 pessoas, Laravel é a *framework* PHP mais usada tanto a nível de projetos pessoais como projetos profissionais, como é possível ver na figura 12.

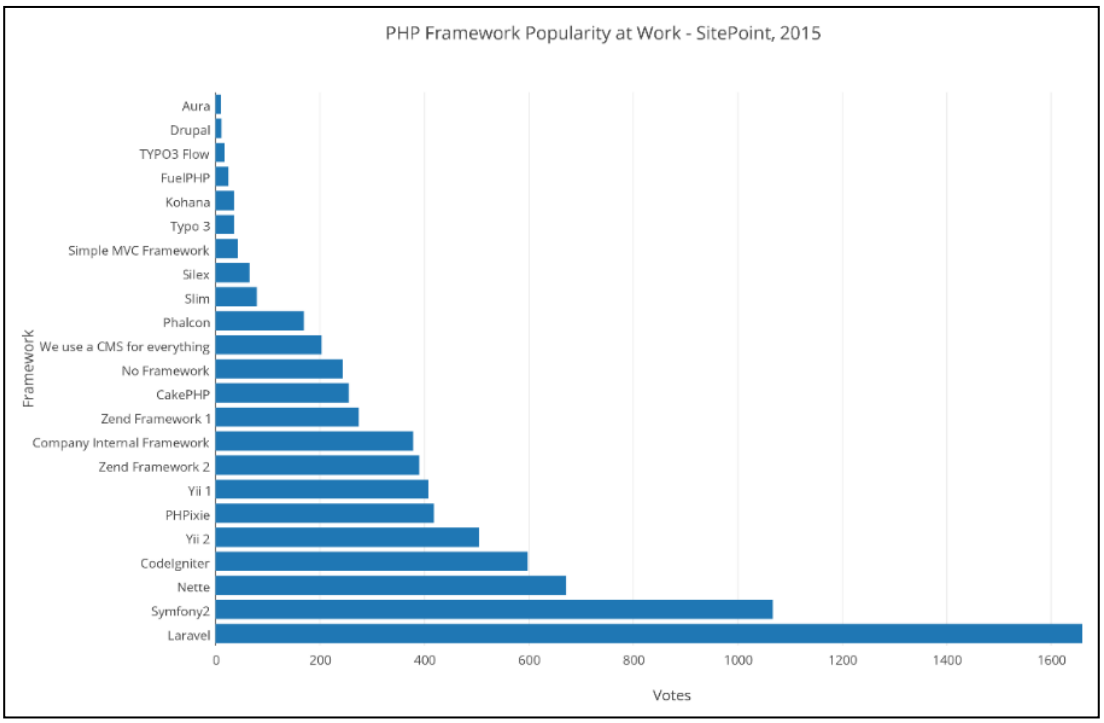


Figura 12 - *Frameworks* PHP mais usadas em projetos profissionais segundo o questionário do site SitePoint

4.3.1.1 – Funcionamento

O processo base de comunicação entre cliente e servidor do Laravel está representado na figura seguinte e apresenta a seguinte sequência:

1. O utilizador realiza um pedido HTTP ao servidor;
2. O pedido é tratado por um mecanismo de rotas (*routing*) que redireciona para o devido controlador (*Controller*);
3. O controlador interage com o modelo (*Model*) para obter informação;
4. O modelo acede à base de dados e responde ao controlador com a devida informação;
5. O controlador cria a vista (*View*) passando-lhe os dados anteriormente recebidos;
6. A vista é processada e gera HTML que é enviado para o cliente.

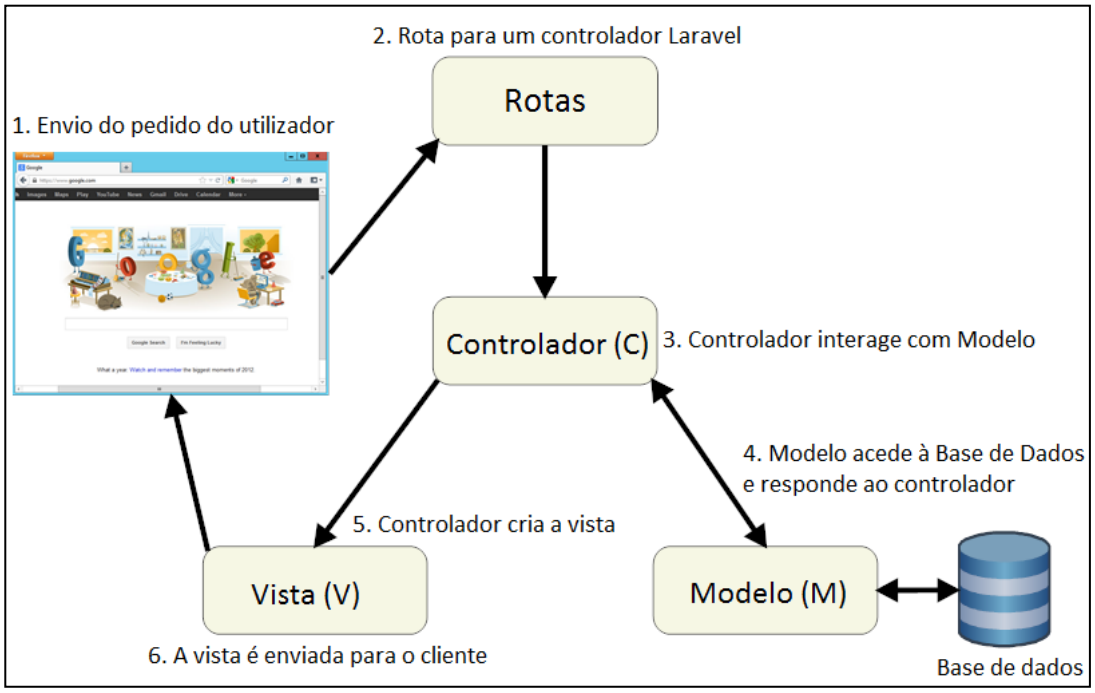


Figura 13 - Padrão MVC utilizador pelo Laravel. Imagem adaptada do site laravelbook [24]

Laravel possui uma funcionalidade de filtragem de pedidos HTTP chamado *Middleware*. O *Middleware* permite, para cada pedido, verificar se determinadas condições ocorrem e atuar conforme o resultado. Durante o desenvolvimento do ACCEPT Web esta funcionalidade foi utilizada para garantir que a cada pedido HTTP a linguagem da aplicação era sempre a mesma. Isto foi necessário porque a linguagem era mudada para a de padrão da aplicação a cada pedido recebido, tal como identificado por vários outros programadores [20]. A utilização do *Middleware* está detalhada no capítulo seguinte.

4.3.1.2 - Funcionalidades do Laravel utilizadas

Esta secção apresenta algumas das funcionalidades nativas e extensões do Laravel que foram utilizadas durante o desenvolvimento do servidor *Web*.

- **Blade**

Durante o desenvolvimento do ACCEPT Web foi usado o sistema de *templates* Blade. Um sistema de *templates* combina um ou mais *templates* e vistas com um conjunto de dados de forma a criar páginas *Web*.

```
//import com PHP
require_once('header.php');
//import com Blade
@include('header.php')

(...)

//Abrir formulário com rota para "login"
{!! Form::open(array('route' => 'login', 'method' => 'POST')) !!}
//Criar campos de "input" (username e password)
{!! Form::text('username', array('placeholder' => 'Insert Username')) !!}
{!! Form::password('password', array('placeholder' => 'Insert Password'))
!!}
//Botão para "submit" do formulário
{!! Form::submit('Login', array('class' => 'btn btn-login')) !!}
{!! Form::close() !!}
```

Excerto de Código Fonte 3 - Exemplo do uso do *Blade* para importar ficheiros e para criar um formulário

O Blade possui uma sintaxe própria que simplifica as operações de controlo, (como condições e *loops*), bem como um sistema de *cache* das vistas do lado do servidor para melhorar a performance. Na fase inicial do projeto, uma vez que as vistas apenas continham código HTML foi necessário adaptar as vistas para a sintaxe do Blade como por exemplo os *imports* de outros ficheiros. O excerto de código fonte 3 apresenta um exemplo do uso da sintaxe do Blade.

- **Artisan**

Artisan é uma interface de linha de comandos que permite realizar várias ações num projeto Laravel como ver as rotas existentes ou criar ficheiros de forma automática, como por exemplo controladores ou modelos. No desenvolvimento do ACCEPT Web, a utilização deste aplicativo serviu maioritariamente para a criação dos ficheiros para a configuração da base de dados através do uso da *package laravel-crud-generator* (ver capitulo 4.3.4).

- **Autenticação e Autorização da Aplicação *Web***

Uma vez que o acesso à aplicação necessitava de autenticação prévia por parte do utilizador, foi utilizado o sistema de autenticação e autorização do Laravel. Este sistema usa filtros que permitem verificar se uma certa condição se verifica antes de chamar uma rota. Desta forma, sempre que uma rota é chamada, é verificado se existe um utilizador autenticado antes de redirecionar o processamento para um controlador. Caso não haja nenhum utilizador autenticado, o *browser* é redirecionado para a janela de login. Um exemplo da utilização do mecanismo de filtros é apresentado no excerto de código fonte 4.

```
//Filtro
Route::filter('isLoggedIn', function(){
	if(\Session::get('loggedUser') == null){
		//Redireciona para a janela de login
		//se não houver utilizador autenticado
		return \Redirect::to('login');
	}
});
```

```
//Rota
Route::get('exampleRoute',array(
	//Filtro chamado antes de redirecionar para o controlador
	'before' => 'isLoggedIn',
	'as' => 'exampleRoute',
	'uses' => 'exampleRoute@exampleController'));
```

Excerto de Código Fonte 4 - Exemplo do uso de um filtro numa rota

Quando o utilizador tenta fazer a autenticação, as credenciais inseridas no formulário de login são enviadas para a API. Caso a API retorne um código de sucesso, as informações do utilizador são guardadas na sessão numa variável chamada "loggedUser". Caso contrário é apresentado um erro ao utilizador a dizer que as credenciais estão erradas ou que não foi possível fazer o login, dependendo do código de erro. O excerto de código fonte 5 apresenta o método de autenticação do utilizador.

```
public function performLogin(){

	//Adquirir  as credenciais
	$username = \Input::get('username');
	$encryptedPassword = \Crypt::encrypt(\Input::get('password'));

	//Fazer autenticação na API
	$results = $this->modelInstance->login($username,$encryptedPassword);
	$results = json_decode($results,true);

	if($results == null){//Erro de login
		return \Redirect::to('login')->with('loginError',
trans('login.loginError'));
	}else if($results['loginResult'] == 2){ //Erro de combinação
username/password
		return \Redirect::to('login')->with('loginError',
trans('login.usernamePasswordError'));
	}else if($results['loginResult'] == 1){ //Sucesso
		//Colocar informações do utilizador na sessão
		$loggedUser = $results['userInfo'];
		\Session::put($loggedUser);
		//Redirecionar utilizador para a página principal
		return \Redirect::to('index');
	}else{//Erro de login
		return \Redirect::to('login')->with('loginError',
trans('login.loginError'));
	}
}
```

Excerto de Código Fonte 5 - Método de autenticação

- **Suporte Multilingue**

Para o suporte multilingue foi utilizado o mecanismo de *Localization* do Laravel que permite fazer a tradução de *strings* pré-definidas. Estas *strings* estão localizadas em ficheiros existentes na diretoria "resources/lang" do Laravel, sendo que dentro desta diretoria é criada uma pasta para cada linguagem a suportar (neste caso português e inglês). O excerto de código fonte 6 apresenta um exemplo da criação destas *strings* (com o nome "exampleString" e com os respetivos ficheiros referidos) e a sua utilização numa vista.

```
//resources/lang/en/exampleFile.php
return [
      'exampleString' => 'This is an example string in english.';
];
```

```
//resources/lang/pt/exampleFile.php
return [
      'exampleString' => 'Isto é um exemplo de uma string em português.';
];
```

```
//resources/lang/pt/exampleFile.php
<span>{{trans('exampleFile.exampleString')}}</span>
```

Excerto de Código Fonte 6 - Exemplo do uso do mecanismo Localize do Laravel

O método "trans()" apresentado no excerto de código fonte anterior procede à pesquisa da *string* com o nome especificado (ex: *exampleString*) existente no ficheiro especificado (ex: *exampleFile.php*). O nome das *strings* é o que identifica a *string* a usar e é o mesmo para todas as linguagens sendo que o que muda é apenas o texto. O ficheiro usado está dependente da linguagem definida no servidor. Esta linguagem pode ser mudada em *runtime*, no entanto sempre que é feito um pedido HTTP ao servidor a linguagem é automaticamente mudada para a linguagem padrão definida nos ficheiros de configuração do Laravel. Como tal, foi necessário implementar uma solução no ficheiro usando como base algumas das implementações sugeridas num artigo do *website* "Laracast" [20]. O excerto de código fonte 7 apresenta o método criado nos ficheiros de *Middleware* do Laravel. Este método é executado a cada pedido feito ao servidor e procura a linguagem definida na sessão. Essa linguagem é depois atribuída à aplicação para garantir que não é alterada cada vez que o utilizador realiza um pedido ao servidor.

```
<?php
use Closure;
use Session;
use App;
use Config;
class Locale {
      /**
      * Handles an incoming request
```

```
        * @param \Illuminate\Http\Request $request
        * @param \Closure $next
        * @param mixed
        */
        public function handle($request, Closure $next){
                //Caso não haja linguagem na sessão é usada a linguagem padrão
                $language = \Session::get('language',
Config::get('app.locale'));
                //Mudar a linguagem
                \App::setLocale($language);
                //Passar ao próximo request
                return $next($request);
        }
}
```

Excerto de Código Fonte 7 - Solução de *Middleware* implementado no ACCEPT Web

Para poder ser feita a mudança da linguagem da aplicação pelo utilizador, foi colocada uma lista na página de login. Ao escolher a linguagem desejada, a linguagem da aplicação era mudada e permanecia a mesma para todos os *requests*. Para definir as linguagens suportadas pela aplicação, foi criado o ficheiro "languages.php" na pasta de configurações do Laravel, sendo este depois acedido através do Blade.

4.3.2 Guzzle

Devido ao facto das informações consumidas pelo servidor serem originárias de uma API externa e não de uma base de dados relacional, foi necessário encontrar uma ferramenta que possibilitasse a comunicação entre o servidor e a API REST. A escolha do *Guzzle* [7] deveu-se ao facto de ter sido sugerida por um dos colaboradores da Sinmetro. Após uma análise e alguns testes, percebeu-se que a sua instalação e utilização eram relativamente simples e que oferecia as características necessárias para o desenvolvimento do ACCEPT Web. Por forma a não quebrar o padrão arquitetural MVC, todo o código referente à conexão à API foi colocado na pasta de modelos do Laravel.

```
<?php

//APIModel.php

class APIModel
{
        private $user;
        private $password;
        private $client;
        private $APIBaseURL = 'http://X.X.X.X:YYY/Caminho/da/API/';

        public function __construct()
        {
                $this->user = 'user';
                $this->password = 'password';
                $this->client = new \GuzzleHttp\Client(['auth' => [$this->user,
$this->password]]);
```

```
        }

        public function connectAPI($method)
        {

                try {
                        $res = $this->client->get($this->APIBaseURL . $method);
                } catch (GuzzleException $e) {
                        $message = trans('database.erroDatabase');
                        return $message;
                }

                return $res->getBody();
        }
}
```

Excerto de Código Fonte 8 - Exemplo do início do serviço Guzzle

O excerto de código fonte 8 apresenta a forma usada para realizar a conexão à API, em que "X.X.X.X" representa o IP da API e "YYYY" representa o porto. A variável "*$client*" contém a instância do cliente *Guzzle*, as variáveis "*$user*" e "*$password*" servem para fazer a autenticação e a variável "*$res*" representa a resposta recebida da API, geralmente sobre o formato JSON. Por fim a variável "*$method*" recebida por parâmetro é o que determina qual o método da API a ser chamado, juntamente com as variáveis necessárias.

Para realizar consultas à API, os restantes modelos estendem do modelo apresentado no excerto de código fonte 8 de forma a poderem aceder ao método "connectAPI". No excerto de código fonte 9 é apresentado um exemplo de um modelo que utiliza o método de conexão à API.

```
<?php

//Modelo.php

class Modelo extends APIModel
{
        //Variável da instância do modelo de ligação à API
        private $modelInstance;

        public function __construct(){
                //Inicialização da instância do modelo de ligação à API
                $this->modelInstance = new APIModel();
        }

        public function metodo($var1,$var2,$var3){
                //Exemplo da chamada do método
                //http://X.X.X.X:YYYY/nomeDoMetodo/$var1/$var2/$var3

                //Conexão à API
                $result = $this->modelInstance->connectAPI('nomeDoMetodo/' .
$var1 . '/' . $var2 . '/' . $var3);
                return $result;
```

```
        }
}
```

Excerto de Código Fonte 9 - Exemplo de um modelo que usa o modelo de conexão à API

Finalmente, do lado do controlador, é criada a instância do modelo a utilizar, sendo esta instância usada para realizar as consultas à API. O excerto de código fonte 10 representa um exemplo da utilização de um método com três variáveis em que o resultado obtido é posteriormente descodificado (uma vez que estava no formato JSON) e pronto a ser manipulado para apresentar ao cliente.

```php
<?php

//ExemploController.php

class ExemploController extends Controller
{
      //Variável da instância do modelo
      private $modelInstance;

      public function __construct(){
            //Inicialização da instância do modelo
            $this->modelInstance = new Modelo();
      }

      public function doSomething(){

            //Obter as variáveis
            $var1 = (...);
            $var2 = (...);
            $var3 = (...);

            //Ligação ao modelo
            $result = $this->modelInstance->metodo($var1,$var2,$var3);
            //Descodificação do resultado
            $result = json_decode($results,true);

            //Fazer algo com o resultado (...)
            return (...);
      }
}
```

Excerto de Código Fonte 10 - Exemplo de um controlador que usa um modelo para fazer consultas à API

4.3.3 Laravel Excel

Muitas das grelhas de dados existentes na aplicação continham informação que deveria ser exportada para o formato "Excel". Para tal foi usada a API Laravel Excel [17] que permite descarregar ficheiros Excel personalizados a partir de vistas HTML. Esta API utiliza várias ferramentas do Laravel, incluindo compatibilidade com o motor de *templates Blade*. Apesar

de ser necessária alguma configuração adicional, esta ferramenta permite também fazer a exportação de ficheiros PDF, o que se tornou uma mais-valia visto que existia também a necessidade de descarregar relatório neste formato.

Para criar os ficheiros Excel esta API necessita de receber um ficheiro com uma tabela para proceder à sua exportação. Para tal, para cada grelha que se pretendia extrair foi criado um novo ficheiro na pasta de vistas com a extensão ".blade.php". Este ficheiro continha apenas uma cópia exata do código HTML da tabela existente na aplicação *Web*, sendo que o método para a criação do documento Excel recebia o nome do ficheiro a ser usado bem como a variável para o preenchimento da grelha. Este método permitia também realizar alguns ajustes estéticos ao ficheiro (como o tamanho da células) de forma a que o ficheiro final apresentasse o formato desejado. No excerto de código fonte 11 é apresentado um exemplo da utilização desta API.

```
public function exportExcel(){

      try {
        //Obter dados da sessão
            $excelInfo = \Session::get('infoToExport');
        //Nome do Excel. Ex: Lotes
            $excelName = $excelInfo['excelName'];
        //Data de inicio. Ex: 01 Jan 2015
            $startDate = $excelInfo['startDate'];
        //Data de fim. Ex: 31 Dez 2015
            $endDate = $excelInfo['endDate'];
        //Construir o nome para o ficheiro Excel
            $filename = $excelName . ' ' . $startDate . ' ' . $endDate;

            \Excel::create($filename, function($excel){
            //Criar uma aba no Excel
                  $excel->sheet('Details', function($sheet){

                        //Obter dados da sessão
                        $infoToExport = \Session::pull('infoToExport');
                        //Obter a informação a passar para a tabela
                        $info = $infoToExport[...];
                        //Nome da vista na tabela para determinar o
ficheiro a usar
                        $pageName = $infoToExport["pageName"];

                        //Carregar a tabela com as informações recebidas
                        $sheet->loadView($pageName, compact('info'));
                  });
            })->download('xlsx');

      } catch (\Exception $e) {
            $errorDownloadExcel = trans('excel.erroDownload');
            return \Redirect::back()-
>with('errorDownloadExcel',$errorDownloadExcel);
```

```
        }
}
```

Excerto de Código Fonte 11 - Exemplo do uso da API Laravel Excel

O código apresentado no excerto de código fonte anterior (tendo em conta os exemplos apresentados no mesma) irá dar origem a um ficheiro Excel chamado "Lotes 01 Jan 2015-31 Dez 2015.xlsx". Dentro desse ficheiro será criada um separador com o nome "Details" e as informações nela contidas são originárias do preenchimento da tabela existente na pasta de vistas do Laravel com o nome passado pela variável "$pageName". A figura 14 apresenta o resultado final da exportação do ficheiro em formato Excel.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	**Identificação**	**Estatística**																	**Indicadores**	
2	**Processo**	**Amostras**	**Tamanho Amostra**	**Pesagens**	**Qtd Nominal**	**Média**	**D. Nominal**	**Min**	**Max**	**Amplitude**	**D. Padrao**	**lie**	**lse**	**lic**	**lsc**	**CPK**	**CL**	**D**	**N<TL1**	**N<TL2**
3	Linha 0 - 1500 mL	1	4	4	1500	1503,22	3,22	1501,72	1504,08	2,36	1,07	1494	1506	1494	1506	CPK		*0*	0	0
4	Linha 0 - 1500 mL	1	4	4	1500	1504,88	4,88	1497,84	1508,93	11,09	4,99	1494	1506	1494	1506	CPK		*0*	0	0
5	Linha 0 - 375 mL	2	4	8	375	375,81	0,81	371,47	381,28	4,22	1,97	369	381	369	381	CPK		*0*	0	0
6	Linha 0 - 375 mL	1	4	4	375	376,05	1,05	372,69	378,4	5,71	2,53	369	381	369	381	CPK		*0*	0	0
7	Linha 0 - 375 mL	1	4	4	375	376,04	1,04	375,13	377,28	2,15	1,08	369	381	369	381	CPK		*0*	0	0
8	Linha 0 - 375 mL	1	4	4	375	377,04	2,04	375,51	377,97	2,46	1,07	369	381	369	381	CPK		*0*	0	0
9	Linha 0 - 375 mL	1	4	4	375	375,66	0,66	373,37	377,07	3,7	1,78	369	381	369	381	CPK		*0*	0	0
10	Linha 0 - 375 mL	2	4	8	375	378,42	3,42	374,83	382,22	4,23	1,91	369	381	369	381	CPK		*0*	0	0
11	Linha 0 - 375 mL	2	4	8	375	374,19	-0,81	372,73	376,17	2,7	1,23	369	381	369	381	CPK		*-1*	0	0
12	Linha 0 - 500 mL	2	4	8	500	497,4	-2,6	494,06	503,44	6,72	2,89	494	506	494	506	CPK		*-1*	0	0
13	Linha 0 - 750 mL	1	4	4	750	750,78	0,78	749,34	752,57	3,23	1,51	744	756	744	756	CPK		*0*	0	0
14	Linha 0 - 750 mL	1	4	4	750	749,29	-0,71	747,97	750,71	2,74	1,48	744	756	744	756	CPK		*-1*	0	0
15	Linha 0 - 750 mL	1	4	4	750	747,1	-2,9	743,63	750,48	6,85	3,12	744	756	744	756	CPK		*-1*	0	0
16	Linha 0 - 750 mL	1	4	4	750	750,41	0,41	748,18	752,83	4,65	1,92	744	756	744	756	CPK		*0*	0	0

Figura 14 - Exemplo de um ficheiro Excel exportado

Numa fase mais avançada do projeto, foi também necessário importar ficheiros Excel (através dum formulário preenchido pelo utilizador). Para facilitar a leitura dos ficheiros, foram utilizados alguns dos métodos desta API para converter os ficheiros para o formato CSV (*Comma Separated Values*) (ver capítulo 4.3.5).

Uma das dificuldades no uso desta API prendeu-se no facto de as variáveis recebidas pelo método do controlador não estarem disponíveis dentro do método de *callback* da API. A resolução para este problema consistiu em colocar as informações necessárias na sessão aquando da criação da página. Desta forma era possível aceder às variáveis da sessão necessárias dentro do método do Laravel Excel sendo assim possível criar o ficheiro para exportação.

4.3.4 laravel-crud-generator

Para fazer a gestão do *backoffice* foi utilizada a *package* "laravel-crud-generator" [43]. Esta *package* está disponível para a versão 5 do Laravel e permite uma criação rápida dos ficheiros necessários para a gestão das tabelas da base de dados (vistas, modelos, controladores, rotas, etc…) através de comandos *artisan* [44].

Ao contrário do que foi feito para o *front-end*, no *backoffice* a comunicação com a base de dados foi efetuada diretamente através de tabelas ou de vistas SQL em vez do uso da API desenvolvida pela Sinmetro. Apesar de se entender que o método adotado não era o mais correto uma vez que ia contra a arquitetura anteriormente definida, a tomada desta decisão deveu-se ao facto de na altura em que se iniciou o desenvolvimento desta funcionalidade (inicio de março de 2016) a API REST não conter as funcionalidades necessárias para a implementação, e o departamento de produção da mesma não prever que essas funcionalidades ficassem prontas em tempo útil.

Por forma a realizar a conexão entre o servidor PHP e a base de dados SQL *Server* foi necessária a instalação e configuração de *drivers* externos específicos uma vez que estes não estavam incluídos na versão instalada do *software* WAMP. Assim, foram descarregados os *drivers* disponibilizados pela Microsoft [45] e seguidas as instruções para configurar o servidor corretamente. Inicialmente houve algumas dúvidas em relação à versão dos *drivers* a utilizar, bem como alguns erros durante a configuração. No entanto, após um processo de *debug* dos erros encontrados foi possível instalar corretamente os *drivers* no servidor e realizar as *queries* à base de dados.

Uma vez que o *laravel-crud-generator* apenas está preparada para gerar comandos *mySQL*, foi necessário adaptar os controladores para utilizarem *queries SQL Server*. Foi também necessário mudar os *imports* de ficheiros externos das vistas uma vez que estas eram criadas através de um *template*, assim como algumas configurações gerais como as chaves primárias e os nomes das rotas, entre outras, uma vez que o nome destes campos na base de dados era diferente dos nomes esperados pelo *plugin*. Inicialmente os ficheiros foram mudados manualmente, no entanto, após se perceber que seria vantajoso, foram ajustados os *templates* que o *laravel-crud-generator* usa para criar os ficheiros, sendo depois apenas necessário realizar pequenas mudanças específicas a cada tabela.

4.3.5 CSV

Por forma a preencher os dados da base de dados rapidamente, foi identificada a necessidade de importar ficheiros. Uma vez que os ficheiros esperados teriam a extensão ".csv" foi utilizada a biblioteca CSV [46] para facilitar a leitura e *parsing* dos ficheiros.

O uso desta biblioteca começa pela importação e armazenamento do ficheiro enviado por um formulário preenchido pelo utilizador no servidor. Após o armazenamento é feita a leitura do ficheiro com auxílio da API e é processado o seu conteúdo para criar os comandos SQL que irão preencher a base de dados. No formulário preenchido pelo utilizador é também escolhido o caracter que irá servir de delimitador durante a leitura do ficheiro.

No excerto de código fonte 12 é apresentado um exemplo da leitura e *parsing* do ficheiro enviado pelo utilizador. Este método cria uma cópia do ficheiro enviado pelo utilizador que depois é lida com auxílio da bliblioteca CSV. De seguida são lidos os *headers* do ficheiro para determinar quais as colunas a importar e construir a *query* para importar os valores. Finalmente todas as linhas do ficheiro são iteradas individualmente sendo os valores dessa linha utilizados para preencher a base de dados.

```php
<?php

use League\Csv\Reader;

public function importFromFile(){

	//Receber dados do formulário
	$file = \Input::file('file');
	//Obter o nome do ficheiro
	$fileName = $file->getClientOriginalName();
	//Definir caminho da pasta temporária onde o ficheiro será guardado
	$destinationPath = public_path() . '/DBTempFiles';
	//Mover o ficheiro para a pasta de destino
	$file->move($destinationPath,$fileName);

	//Usar a API para ler o ficheiro para obter os headers
	$csv = Reader::createFromPath($destinationPath . '/' . $fileName);
	//Obter caracter delimitador para ler o ficheiro (vírgula ou ponto e
vírgula)
	$csv->setDelimiter(\Input::get('delimiter'));

	//Iterar a primeira linha do ficheiro para obter os headers e criar a
query de introdução de dados
	$query = (...);

	//Começar a iterar a partir da segunda linha do ficheiro
	$csv->each(function($row) use (&$query){
		//Iterar a linha para retirar os valores
		$valores = (...);
		//Executar a query para introduzir os valores na base de dados
		return DB::insert($query,$valores);
	});
}
```

Excerto de Código Fonte 12 - Exemplo do uso da API CSV para ler um ficheiro e popular a base de dados

No formulário o utilizador deverá também escolher qual a tabela da base de dados a que o ficheiro corresponde. No ficheiro importado, é esperado que na primeira linha estejam presentes os campos da base de dados, e nas linhas seguintes os dados pela ordem definida na

primeira linha. O nome e a ordem pela qual as colunas estão presentes no ficheiro não seguem qualquer regra, uma vez que após importar o ficheiro, o utilizador poderá fazer um mapeamento entre as tabelas da base de dados e das colunas do ficheiro. A figura 15 apresenta a janela de mapeamento após a importação do ficheiro.

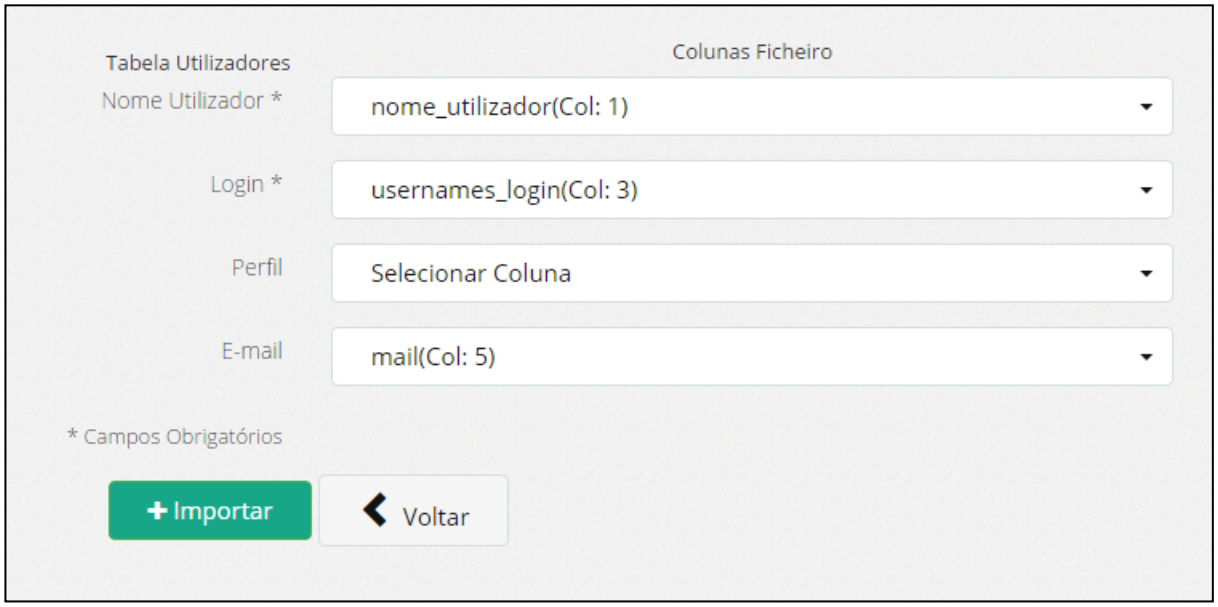


Figura 15 - Formulário de mapeamento entre uma tabela da base de dados e um ficheiro

Por forma a tornar o processo mais intuitivo e conveniente para o utilizador, são iteradas todas as linhas do ficheiro e importados todos os valores corretos. Mesmo que haja erros em alguma das linhas a importação continua até ser lido todo o ficheiro. Na prática, sempre que um valor é introduzido com sucesso na base de dados, é incrementado um valor a uma variável que conta o número de linhas inseridas com sucesso e passa-se à próxima linha do ficheiro. Caso haja um erro ao introduzir um valor na base de dados, é guardado o número da linha (através do contador da iteração) para mais tarde apresentar ao utilizador e passa-se à próxima iteração.

Caso haja pelo menos um valor introduzido com sucesso, o utilizador é redirecionado para a página correspondente à tabela escolhida, sendo informado de quantos registos foram adicionados e quais as linhas do ficheiro que contêm erros (caso tenham havido erros). Caso não haja pelo menos um valor introduzido com sucesso na base de dados (uma vez que o ficheiro pode ser inválido ou o mapeamento tenha sido feito de forma incorreta), o utilizador é redirecionado para a janela de importação do ficheiro com uma mensagem de erro.

4.3.6 jQuery

jQuery [11] é uma biblioteca *open-source* de *javascript* compatível com várias plataformas concebida para simplificar a escrita de código *javascript*, focando-se primariamente nas tarefas de manipulação de DOM (*Document Object Model*), manipulação de eventos (por exemplo, cliques), animações, CSS e pedidos AJAX.

Optou-se por utilizar esta biblioteca, pois não só é a mais popular [18][47] como também tem praticamente todas as características necessárias para o desenvolvimento do ACCEPT Web. Para além disso, muitos dos *plugins* utilizados durante o estágio baseiam-se em *jQuery*, incluindo muitos dos que foram disponibilizados no *template* fornecido no início do estágio.

4.3.7 Date Range Picker e Moment.js

Por norma, todas as pesquisas à API necessitam de duas datas (início e fim), introduzidas pelo utilizador. Assim, tornou-se necessária a utilização de uma solução que permitisse fazê-lo de forma intuitiva. Para tal foi utilizado o *plugin* jQuery *Date Range Picker* [8]. Este componente utiliza outras 3 *frameworks*/*plugins*: *Bootstrap* para o visual, *jQuery* para a funcionalidade e Moment.js [9] para a manipulação de datas e horas. A figura 16 apresenta um exemplo de um *Date Range Picker* apresentado na documentação do *plugin*.

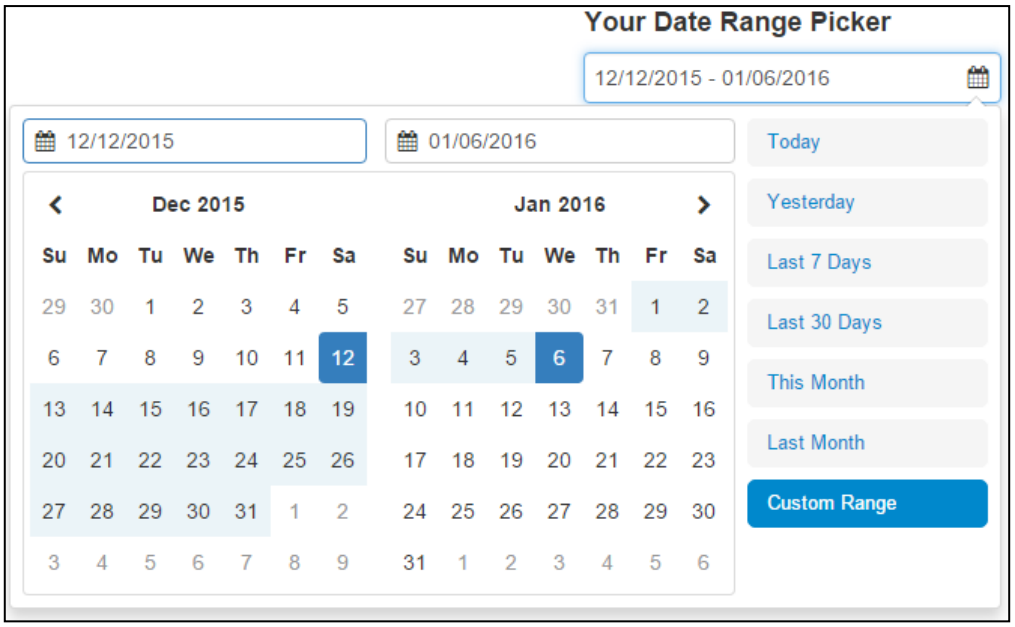


Figura 16 - Exemplo de um Date Range Picker. Imagem retirada da documentação do *plugin Date Range Picker* [8]

A maior dificuldade na implementação desta funcionalidade foi o facto de, por omissão, a data estar configurada para o formato Mês/Dia/Ano, sendo que o formato esperado seria Dia/Mês/Ano.

Na fase inicial de desenvolvimento houve também um problema com o estado das datas uma vez que sempre que se recarregava a página as datas mostradas eram mudadas para uma data definida estaticamente. A solução passou por manter o estado colocando a data escolhida num campo escondido. Ao fazer um pedido ao servidor, o valor dessa data era reenviado, permitindo que o estado (valor da data) fosse mantido entre dois pedidos. Depois o servidor enviava de volta essa data na resposta e era colocada novamente nos campos escondidos de forma a atualizar a data no *Date Range Picker*. Numa fase posterior de desenvolvimento este problema deixou de ser relevante uma vez que os pedidos passaram a ser feitos por AJAX, não sendo necessário recarregar a página.

4.3.8 Datatables

DataTables [10] é um *plugin* para *jQuery* que permite criar tabelas com dados em vistas HTML e adicionar-lhes funcionalidades de forma simples. A escolha deste *plugin* deveu-se ao facto de o *template* fornecido no início do estágio já ter esta solução implementada com alguns exemplos, o que ajudou bastante na sua utilização.

Não só tem suporte a AJAX, permitindo recarregar as tabelas sem recarregar a página, como também permite a paginação da tabela de forma bastante simples. Uma das funcionalidades que foi particularmente útil foi a funcionalidade de aplicar condições a cada uma das colunas, como por exemplo, pintar os elementos de uma certa coluna. O excerto de código fonte 13 apresenta um exemplo da inicialização do *plugin* DataTables. Neste exemplo as colunas da tabela são pintadas consoante o valor existente no *array* "semaforosLimites", usando o *índex* das linhas percorridas para aceder à posição no *array*.

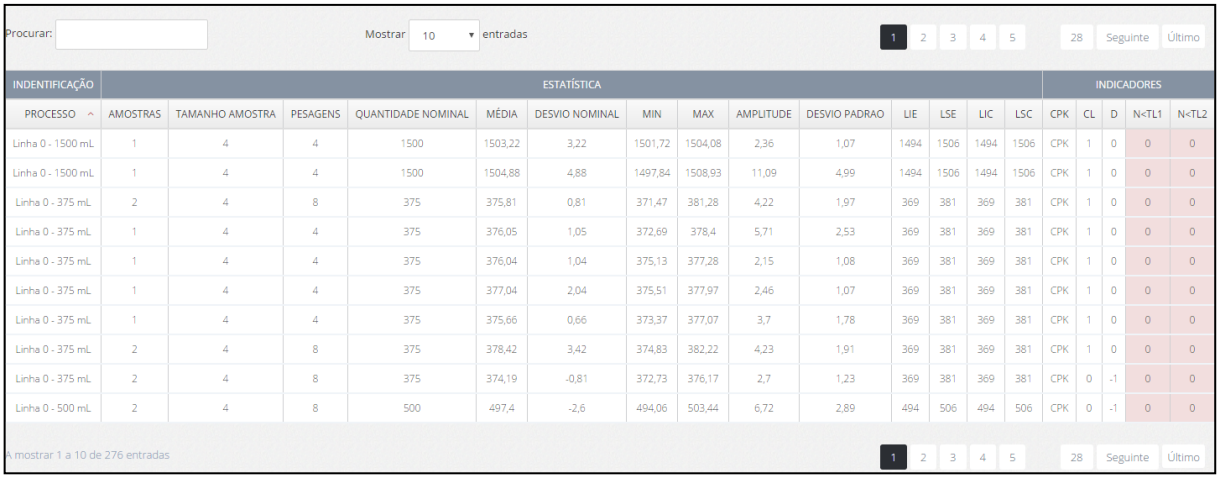
```
function createDatatableAnalise(data, semaforosLimites){

      $('#tableID').Datatable({
            //Array de dados para preencher a tabela
            data: data,
            'columnDefs': [{
                  //Terceira coluna a contar da direita
                  'targets': -3,
                  'createdCell': function (td, cellData, rowData, row,
col){
```

```
                        if(semaforosLimites[row][0] == 1){
                            //Pintar célula de vermelho se o código for 1
                            $(td).css('background-color','red');
                        }else if(semaforosLimites[row][0] == 2){
                            //Pintar célula de amarelo se o código for 2
                            $(td).css('background-color','yellow');
                        }
                    }
                }]
        });
}
```

Excerto de Código Fonte 13 - Exemplo de um método para criação de uma tabela com o Datatables

Na figura 17 está representado o resultado produzido pelo método apresentado na figura anterior (com a penúltima e antepenúltima colunas pintadas).



Procurar: Mostrar 10 entradas — 1 2 3 4 5 … 28 Seguinte Último

INDENTIFICAÇÃO	ESTATÍSTICA														INDICADORES				
PROCESSO	AMOSTRAS	TAMANHO AMOSTRA	PESAGENS	QUANTIDADE NOMINAL	MÉDIA	DESVIO NOMINAL	MIN	MAX	AMPLITUDE	DESVIO PADRAO	LIE	LSE	LIC	LSC	CPK	CL	D	N<TL1	N<TL2
Linha 0 - 1500 mL	1	4	4	1500	1503,22	3,22	1501,72	1504,08	2,36	1,07	1494	1506	1494	1506	CPK	1	0	0	0
Linha 0 - 1500 mL	1	4	4	1500	1504,88	4,88	1497,84	1508,93	11,09	4,99	1494	1506	1494	1506	CPK	1	0	0	0
Linha 0 - 375 mL	2	4	8	375	375,81	0,81	371,47	381,28	4,22	1,97	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	1	4	4	375	376,05	1,05	372,69	378,4	5,71	2,53	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	1	4	4	375	376,04	1,04	375,13	377,28	2,15	1,08	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	1	4	4	375	377,04	2,04	375,51	377,97	2,46	1,07	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	1	4	4	375	375,66	0,66	373,37	377,07	3,7	1,78	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	2	4	8	375	378,42	3,42	374,83	382,22	4,23	1,91	369	381	369	381	CPK	1	0	0	0
Linha 0 - 375 mL	2	4	8	375	374,19	-0,81	372,73	376,17	2,7	1,23	369	381	369	381	CPK	0	-1	0	0
Linha 0 - 500 mL	2	4	8	500	497,4	-2,6	494,06	503,44	6,72	2,89	494	506	494	506	CPK	0	-1	0	0

A mostrar 1 a 10 de 276 entradas — 1 2 3 4 5 … 28 Seguinte Último

Figura 17 - Tabela produzida através do Datatables

Um dos maiores desafios encontrados na utilização desta extensão foi o posicionamento dos elementos no DOM (campo de pesquisa, botões de paginação, texto de filtro,…) apresentados na figura anterior. Apesar da documentação existente, o processo de desenvolvimento dos primeiros métodos que usam este *plugin* passou por tentativa e erro uma vez que com a ajuda da documentação não se obteve o resultado esperado. Para resolver o problema foi criado um elemento *div* para cada elemento e posteriormente os elementos foram colocados nos lugares pretendidos através do uso de CSS.

Outro dos problemas encontrados passou pela construção do objeto JSON para preenchimento da tabela, sendo que foi necessário manipular o objeto recebido da API uma vez que o seu formato era diferente do esperado pelo *plugin*.

4.3.9 Flot Charts

Flot Charts [12] é uma biblioteca de jQuery concebida para a criação de gráficos. Esta biblioteca suporta vários tipos de gráficos e permite integração com eventos como mouseover e *click*. Apesar de trazer já integradas várias funcionalidades, esta biblioteca tem algumas limitações mais básicas que são no entanto complementadas pela importação de *plugins* externos, muitas vezes desenvolvidos pela comunidade de programadores [13].

Este *plugin* foi um dos mais utilizados durante o desenvolvimento do ACCEPT Web uma vez que a análise dos dados depende extensivamente da representação visual dos mesmos. O facto de a sua documentação ser bastante bem produzida ajudou bastante na sua utilização e na resolução de problemas. Esta documentação peca, no entanto, pela falta de exemplos de código fonte, sendo necessário recorrer a fóruns de forma a obter resultados que não seriam tão simples de obter pela simples leitura da documentação.

A figura 18 apresenta um gráfico criado através deste *plugin* no qual é possível ver os detalhes de cada ponto ao passar o rato por cima do mesmo.

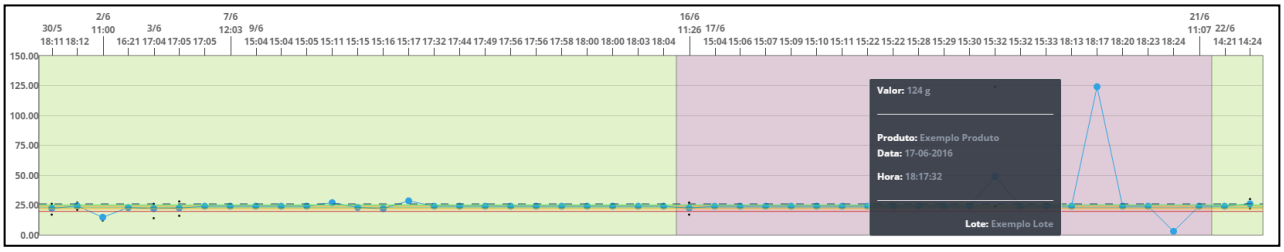


Figura 18 - Gráfico criado através do *plugin* Flot Charts

4.3.9.1 *Plugins* Adicionados

Tal como foi referido anteriormente, a biblioteca Flot Charts recorre à importação de *plugins* para realizar funções que não estão disponíveis utilizando apenas a versão base. Para permitir adicionar animações aos gráficos foi utilizado o *plugin* Flot Animator [14] que já estava incluído nos ficheiros do *template* fornecido. A utilização deste *plugin* é relativamente simples, como é percetível pelo excerto de código fonte 14.

```
//Exemplo para a criação de um gráfico simples
var plot = $.plot($("#myChart"), [{ data: [...] }]);

//Exemplo para a criação de um grafico com animações
var plot = $.plotAnimator($("#myChart"), [{ data: [...],
      animator: {
```

```
            start: 100, //inicio da animação após a criação do gráfico (em
ms)
            steps: 99, //numero de passos a executar durante a animação
            duration: 1000, //duração da animação (em ms)
            direction: "left", //direção da animação (esquerda, direita ou
centro)
      }
}]);
```

Excerto de Código Fonte 14 - Exemplo da utilização do *plugin* Flot Animator

Outra das funcionalidades que foi necessária implementar foi o desenho de linhas tracejadas para representar o valor da média. Uma vez que a versão base da biblioteca não possuía esta funcionalidade de forma nativa, foi necessária a inclusão do *plugin* externo "flot.dashes" [53] que foi importado e integrado no projeto.

4.3.10 Cicliful

Circliful [15] é um pequeno *plugin* para jQuery que permite criar círculos de progresso animados. A utilização deste *plugin* segue dois passos relativamente simples: a criação da tag div com os dados necessários e a sua inicialização através de jQuery. No excerto de código fonte 15 é apresentado um exemplo da utilização deste *plugin* no ACCEPT Web.

```
//Obter percentagem da API
var percentagem = (...);

//Limpar a div com id "pie_chart"
$("#pie_chart").empty().removeData();
//Atributo para representar a percentagem em texto
$("#pie_chart").attr("data-text",percentagem+"%");
//Atributo para desenhar a percentagem no gráfico
$("#pie_chart").attr("data-percent",percentagem);
//Outros atributos (cor, tamanho do texto, etc...)
$("#pie_chart").attr(...);

//Método para desenhar o gráfico
$("#pie_chart").circliful();
```

Excerto de Código Fonte 15 - Exemplo da utilização do *plugin* Circliful

4.3.11 jQuery-localize

Tendo em conta a necessidade do suporte multilingue, foi necessária a implementação de uma solução de tradução do lado do cliente uma vez que as páginas podiam ser atualizadas através de AJAX, sendo necessário traduzir alguns dados do lado do cliente. Após algumas comparações entre várias bibliotecas e *plugins*, foi escolhido o "jQuery localize" [22]. A sua fácil implementação e funcionamento foram os fatores que mais contribuíram para esta escolha.

O método de funcionamento deste *plugin* passa pela atribuição do atributo "data-localize" a um elemento HTML que irá conter texto. Dentro deste atributo deverá ser colocado o caminho para a *string* pretendida. Todas as *strings* são definidas num ficheiro JSON.

Após a criação dos elementos HTML com as *strings* pretendidas, é necessário chamar o método apresentado no excerto de código fonte 16 para localizar as *strings* e substituir o conteúdo dos elementos HTML com o devido texto. É também possível atribuir um texto padrão ao elemento que será apresentado caso a *string* não exista ou a tradução falhe.

```
//HTML
<span data-localize='caminho.string'>Nome Padrão</span>
//Javascript
function translate(){
      $ ('[data-localize]').localize ('localizePath/languages/fiIeName',{
            language:language
      });
}
```

Excerto de Código Fonte 16 - Método para tradução de strings

O método para a tradução deverá receber o nome do ficheiro JSON que contém as *strings* traduzidas. O segundo parâmetro deste método é a linguagem da tradução, sendo que os caracteres usados deverão ser iguais aos usados no nome do ficheiro JSON. Assim, considerando que a variável "language" tem o valor "pt", ao chamar o método do excerto de código fonte anterior, a biblioteca irá tentar localizar um ficheiro chamado "fileName-pt.json".

De forma a determinar qual a linguagem que está a ser utilizada atualmente pelo utilizador, foi criado uma *tag* "meta" que contém a linguagem atual (recebida pelo servidor). Sempre que o método de tradução é chamado, é utilizada a string existente neste elemento HTML de forma a saber qual a linguagem que o utilizador está a usar, tal como mostrado no excerto de código fonte 17.

```
//HTML
<meta name='language' content='<?php echo App::getLocale() ?>'/>
//Javascript
var currentLanguage = $('meta[name="language"]').attr('content');
```

Excerto de Código Fonte 17 - Uso da tag "meta" para saber qual a linguagem atual

4.3.12 Sortable

Durante o processo de desenvolvimento do ACCEPT Web foi identificada a necessidade de dar um comportamento dinâmico ao *dashboard* já que este apresenta os principais conteúdos visuais do *front-end*. As funcionalidades necessárias eram:

- Permitir mostrar ou esconder cartões (elementos da vista que contêm secções específicas de informação);
- Trocar a posição entre os cartões;
- Expandir ou diminuir o tamanho dos cartões na vertical e horizontal.

Por forma a tratar da troca de posição entre os cartões, foi utilizada a biblioteca Sortable [48]. Esta biblioteca de Javacript foi concebida para permitir a reordenação de elementos HTML (listas, *divs*, etc.) através de *drag and drop*.

Para trocar a posição entre os cartões, foi utilizada a opção de *handle*. Para usar esta opção, foi definida uma classe para um elemento *span* dentro de cada cartão que iria atuar como botão de arraste. Para definir os elementos a arrastar foi necessário atribuir um *id* ao *container* dos cartões. Depois de iniciada a biblioteca, a opção de arraste estaria disponível para todos os elementos diretamente dentro do *container*. O excerto de código fonte 18 apresenta um exemplo do uso da biblioteca Sortable.

```
//HTML
<div id="container">
      //Elemento a trocar de posição
      <div id="cartao1">
            //Elemento que permite fazer drag/drop do elemento pai
            <span class="glyphicon glyphicon-move"></span>
      </div>
      (...)
</div>
```

```
//Javascript

//Definição do elemento pai
Sortable.create(container, {
      //Definição da classe dos elementos que permitem fazer drag/drop
      handle: '.glyphicon-move ' ,
      //Tempo de animação do drag/drop (em ms)
      animation: 50
});
```

Excerto de Código Fonte 18 - Exemplo do uso da biblioteca Sortable

4.3.13 Javascript Cookie

Uma vez que existia a necessidade de personalizar o aspeto dos cartões do *dashboard* foi necessário recorrer a *cookies* para guardar localmente uma variável com a ordem e tamanho

dos cartões. Assim, sempre que o utilizador voltava à página principal do ACCEPT Web o aspeto seria igual ao do último acesso. Apesar de se considerar limitada (uma vez que deixa de fazer sentido, por exemplo, ao limpar as *cookies* ou trocar de computador), esta foi a opção tomada pois a criação de um sistema de perfis de utilizador para a gravação de preferências gráficas não foi considerada prioritária pelo gestor de projeto, ficando esta funcionalidade para implementações futuras.

Para tal foi utilizada a API "Javascript Cookie" [49] que facilita a criação e acesso a *cookies*. A utilização desta API deveu-se à sua simplicidade de utilização, tal como é possível ver no excerto de código fonte 19, onde é apresentado um exemplo da criação e aquisição de um objeto guardado das *cookies* do *browser*.

```
//Objeto JSON com a ordem e tamanho dos cartões do dashboard
var dashboardArrangement = { ... };
//Guardar o objeto nas cookies
Cookies.set('dashboardArrangement', dashboardArrangement);
```

```
//Carregar a cookie no acesso à página
var dashboardArrangement = Cookies.getJSON('dashboardArrangement');
if(dashboardArrangement != null){
      //Organizar os cartões do dashboard conforme o objeto guardado nas
cookies
      organizeDashboard(dashboardArrangement);
}
```

Excerto de Código Fonte 19 - Exemplo da criação e aquisição de um objeto guardado em *cookies* através da API "Javascript Cookie"

4.3.14 Bootstrap

Bootstrap [23] é a mais popular *framework* de *front-end* para a construção de conteúdos *Web* responsivos [16][51]. O objetivo desta *framework* é fornecer formatações e *layouts* pré-programados através de classes de CSS de forma a que o programador necessite apenas de atribuir classes aos elementos, sabendo à partida qual o aspeto que estes elementos vão ter.

Durante o desenvolvimento do ACCEPT Web foram usadas várias funcionalidades e componentes disponibilizados pelo Bootstrap como por exemplo, botões, caixas de texto, tabelas, etc. Uma das funcionalidades mais usadas e ao mesmo tempo uma das que ofereceu mais dificuldades foi o sistema de *grid* [23]. Estas dúvidas foram no entanto ultrapassadas após uma reunião com um dos colaboradores da empresa MediaWeb Creations (empresa que forneceu o *template*). Após ultrapassada esta dificuldade, o sistema de *grid* foi adaptado em quase todas as páginas do projeto.

Este sistema permite dividir um *container* até 12 colunas com espaço igual entre elas, sendo que ao usar a divisão máxima cada coluna ocupará cerca de 8,33% da largura do elemento pai. O sistema de *grid* oferece classes específicas destinadas a cada tamanho de ecrã. O nome destas classes começa sempre por "col-" e dependendo do tamanho pretendido poderá ser sucedido por "xs-" para ecrãs muito pequenos (menos de 768px), "sm-" para ecrãs pequenos (mais de 768px), "md-" para ecrãs médios (mais de 992px) e "lg-" para ecrãs grandes (mais de 1200px). Finalmente, deverá ser usado um número entre 1 e 12 para determinar o tamanho que o elemento irá ocupar. A figura 19 apresenta um exemplo do uso deste sistema, utilizando vários tipos de tamanhos para os elementos.

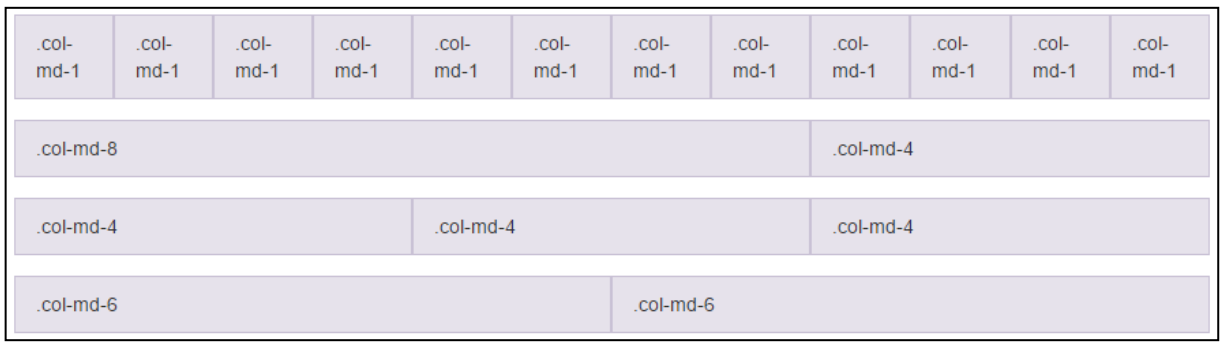


Figura 19 - Exemplo do sistema de *grid* da *framework* Bootstrap. Imagem retirada do site da *framework* Bootstrap [25]

Caso as classes dos elementos ultrapasse as 12 colunas (ex: col-md-6 e col-md-7), será criada uma nova linha e os elementos que não tiverem espaço na linha de cima passam para a linha de baixo.

4.4 Problemas e Desafios

Nesta secção são mencionados alguns dos problemas e desafios que foram detetados durante o decorrer do estágio. Para esses problemas são também mencionadas as soluções que foram tomadas para os resolver.

- **Breadcrumbs**

Numa fase inicial foi decidido que deveria ser criado um sistema de *breadcrumbs* de forma a facilitar a navegação do utilizador pelo *dahsboard* do *website*. Este sistema consiste na criação de *links* com os nomes das páginas visitadas para se manter uma ordem e histórico da navegação. Após a implementação de vários tipos de *breadcrumbs*, chegou-se à conclusão de que o número de "níveis de profundidade" que era possível entrar no *website* não justificava a

implementação deste sistema. Além disso, uma vez que uma grande parte das páginas do ACCEPT Web funciona como SPA, concluiu-se que o sistema de *breadcrumbs* não era o mais adequado para a navegação.

Para substituir este sistema, foram criados *links* nos botões previamente existentes. Estes links estão sempre presentes durante a navegação no *dashboard*, permitindo ao utilizador trocar entre os vários cartões existentes com um clique. Uma vez que cada *link* representa um cartão novo (sem dependência dos outros), não se justificava a apresentação do caminho percorrido até chegar a esse cartão, sendo que tornou-se apenas visível o cartão em que o utilizador se encontra atualmente.

- **Gráficos**

Sendo o ACCEPT Web uma aplicação com uma vertente visual muito forte, ao longo de todo o desenvolvimento foram utilizados vários tipos de gráficos e várias APIs e extensões para se determinar qual o aspeto final desejado.

Uma vez que o *template* inicial já continha vários exemplos de gráficos, procurou-se fazer uso desses gráficos uma vez que apresentavam o aspeto visual idealizado numa fase inicial. No *dashboard*, por exemplo, foram constantemente experimentados vários tipos diferentes de gráficos de forma a perceber quais eram melhores para representar cada tipo de dados.

- **Modelo MVC**

Numa fase inicial do projeto toda a informação era trocada com a base de dados por intermédio de uma API REST. Para fazer esta troca de informação foi utilizado o Guzzle (cap. 3.3.3) e toda a lógica de conexão à API era feita nos controladores do Laravel, não havendo assim uma organização eficiente do código

Uma vez que o Laravel é uma *framework* MVC, fazia sentido reorganizar o código para seguir este modelo e tornar o código da aplicação mais organizado. Assim alguns dos ficheiros e métodos foram refeitos para se obter a arquitetura MVC. Foi criado um modelo que serve como "ponte" entre a API REST e os restantes modelos. Desta forma o código ficou melhor estruturado e foi removida a lógica de ligação direta entre a base de dados e os controladores.

- ***Single Page Application***

Inicialmente todas as consultas e carregamentos de dados obrigavam a que as páginas tivessem de ser recarregadas por completo. Após uma análise conclui-se que este comportamento não era desejável uma vez que se tornava um processo mais lento, principalmente quando se queriam fazer várias consultas.

Deste modo optou-se pela implementação de uma arquitetura SPA através de AJAX para que a navegação no *website* fosse mais fluida, tanto para o carregamento de tabelas como gráficos ou outros dados.

A implementação desta arquitetura permite ao utilizador navegar entre vistas de forma mais simples e rápida uma vez que para cada acesso apenas são carregadas porções de uma página, não obrigando a que toda a página tenha de ser recarregada.

Esta página foi intencionalmente deixada em branco

5 – Resultados e Testes

Neste capítulo é feita uma apresentação do estado da aplicação findos os 9 meses projetados para o desenvolvimento do projeto assim como os testes realizados.

5.1 Resultado Final

Após os 9 meses de estágio, o que resultou do desenvolvimento do ACCEPT Web foi uma aplicação que permite às empresas de produtos pré-embalados saber o estado e controlar as suas linhas de produção. Esta informação é imprescindível para as empresas que queiram não só maximizar o rendimento da produção e minimizar os seus custos, mas também garantir que estão a ser cumpridas todas as leis aplicáveis. Esta aplicação "transporta" as funcionalidades da versão *desktop* já existente do ACCEPT, tendo objetivo de proporcionar uma plataforma moderna, intuitiva e prática de forma a que se possa tomar o maior partido da mesma.

Apesar de não terem sido implementadas todas as funcionalidades existentes na versão *desktop* do ACCEPT ou idealizadas pela Sinmetro (principalmente devido ao facto da sua implementação não ser praticável no espaço de tempo do estágio), realizaram-se as funcionalidades que se consideravam prioritárias e de mais importância para serem utilizadas em casos reais.

No final do estágio, o ACCEPT Web possuía as seguintes funcionalidades principais:

- *Dashboard* principal onde o utilizador tem acesso rápido às principais informações das linhas de produção num espaço de tempo escolhido. Na parte inicial desta vista, o utilizador pode ver o número de linhas de produção que estiveram a produzir, os processos de produção, o número de amostras e pesagens efetuadas, o número de lotes produzidos e o número de lotes com que não obedeceram os critérios legais, bem como algumas estatísticas relevantes. É também possível ver em detalhe a progressão

de cada processo de produção ao longo do tempo, assim como todas as amostras e pesagens efetuadas. É nesta vista que estão presentes os principais gráficos da aplicação de forma a dar ao utilizador uma visão rápida e simples de interpretar. Finalmente, é também possível através desta vista descarregar relatórios dos lotes de produção que tenham sido previamente criados pelos utilizadores;

- Janela de gestão de processos, onde o utilizador pode criar e gerir pastas para organizar os seus processos;
- Janela de criação de processos, onde o utilizador pode criar novos processos ou editar processos já existentes;
- Janela onde o utilizador pode ver com mais detalhe a evolução de um processo, tendo acesso a uma carta de controlo detalhada, um histograma e as informações detalhadas de cada amostra;
- Janela de análise geral, onde está presente uma grelha que apresenta as estatísticas e indicadores de qualidade (indicadores legais) de todos os lotes dos processos que decorreram no intervalo de tempo especificado;
- Janela de visualização de lotes, onde é apresentada uma grelha das informações dos lotes produzidos no intervalo de tempo definido. Tanto nesta janela como na janela de análise geral é possível descarregar um ficheiro em formato Excel (.xlsx) com as informações presentes na grelha;
- Janela de configuração de lotes, onde estão presentes as várias linhas de produção da empresa. Para estas linhas é possível configurar o lote que irá começar a produzir, bem como ver todas as linhas de produção que já estão a produzir;
- Menu de configuração, onde o utilizador pode gerir os aspetos inerentes aos dados presentes na base de dados, como por exemplo os utilizadores, os produtos, as embalagens, os fornecedores e as linhas de produção. Este menu está também dotado de uma funcionalidade que permite ao utilizador carregar um ficheiro em formato Excel ou CSV para importar informações em massa para a base de dados.

De forma a tornar a navegação na aplicação o mais fluida e intuitiva possível, foi seguido o padrão arquitetural SPA. Apesar de existir mais de uma página na aplicação, tentou-se, sempre que possível, que os dados das páginas fossem carregados através de AJAX, permitindo que o seu conteúdo fosse mudado em pequenos blocos, evitando assim que a páginas tivessem de ser carregadas por inteiro.

Foi também implementada uma parte do ACCEPT Web que ficou em produção na empresa NovaDelta (ver capítulo 5.3). Esta implementação permite o tratamento da gestão de lotes, utilizando a versão *dektop* do ACCEPT para recolha de dados.

Na figura 20 é apresentado o aspeto da janela de *dashboard* na fase final do estágio, preenchida com dados reais de um processo.

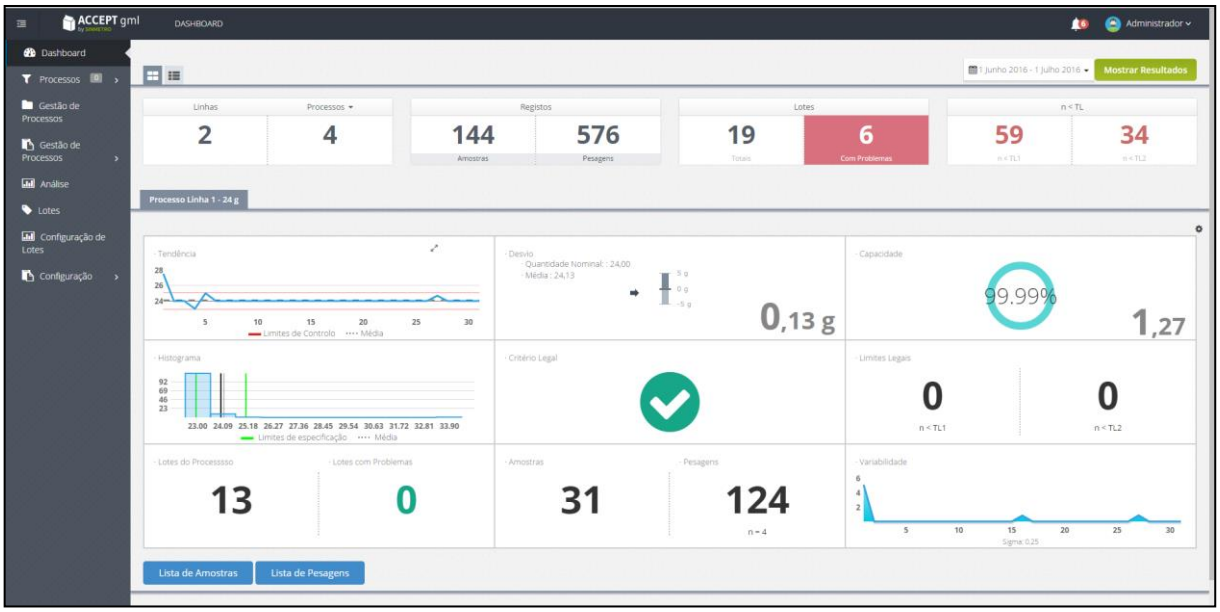


Figura 20 - Aspeto da janela de *dashboard* na fase final do estágio

5.2 Testes de usabilidade

Por forma a validar o funcionamento e comportamento das funcionalidades implementadas, foram realizados testes ao longo do desenvolvimento do projeto. Estes testes foram realizados de forma manual e foram feitos por diversos colaboradores das duas empresas a laborar no local do estágio, tendo como finalidade descobrir eventuais comportamentos não desejáveis da aplicação. Para cada teste, foram dadas ao utilizador as informações que se consideravam relevantes para a realização do mesmo, mas de forma que essa informação não influenciasse o desfecho do teste para se obterem os resultados mais favoráveis e o mais aproximado possível do utilizador final.

Geralmente os testes eram feitos por 1 a 3 pessoas, tentando-se sempre que possível atingir uma maior diversidade no grau de conhecimento dos utilizadores em relação à aplicação a ser testada e a aplicação *desktop* já existente. Desta forma foi possível ter uma ideia do grau de dificuldade de cada utilizador em realizar a tarefa que lhe foi proposta, sabendo à partida se o

utilizador já tinha ou não conhecimento prévio do método de funcionamento da aplicação *desktop*. Estes testes decorreram num ambiente informal, não tendo sido registadas informações detalhadas em relação à prestação do utilizador. Uma vez que eram feitos testes regularmente a cada funcionalidade, era discutido durante a realização do teste quais as melhorias que deviam ser realizadas tendo em conta o progresso e os comentários dos utilizadores. A tabela 3 apresenta alguns exemplos simples das tarefas que foram dadas aos utilizadores. A primeira coluna representa a ação que foi dada ao utilizador e a segunda coluna representa os passos que se esperavam que o utilizador seguisse, sendo que por vezes eram também dadas instruções específicas.

Ação	**Passos a executar**
Fazer login na aplicação	1. Aceder à página de login da aplicação; 2. Escrever o nome de utilizador e *password* fornecidos e realizar login.
Descarregar um ficheiro Excel com as informações dos lotes produzidos no último ano	1. Ir à janela de lotes; 2. Procurar os lotes do último ano utilizando o calendário existente na página; 3. Filtrar os lotes com um nome específico; 4. Descarregar o Excel através de um botão existente na página.
Descarregar um ficheiro Excel com as informações gerais do último ano	1. Ir à janela de análise geral; 2. Procurar as informações do último ano utilizando o calendário existente na página; 3. Filtrar os resultados usando os filtros à escolha do utilizador; 4. Descarregar o Excel através de um botão existente na página.
Ver a informação estatística e gráfica da produção e descarregar o relatório de um lote	1. Ir à página de *dashboard*; 2. Filtrar os dados por um intervalo de tempo escolha do utilizador; 3. Escolher um processo da lista de linhas de produção; 4. Visualizar os gráficos, a carta de controlo e as grelhas de

	amostras e pesagens; 5. Ver os lotes com problemas do processo a ser visualizado; 6. Descarregar o relatório do lote através de uma opção existente na página.
Configurar e parar um lote	1. Ir à página de configuração de lotes; 2. Configurar um lote numa linha de produção à escolha; 3. Ver a estatística do lote; 4. Parar o lote.
Gerir as funcionalidades de *backoffice*	1. Ir a uma página de configuração à escolha do utilizador; 2. Adicionar, visualizar, editar e apagar valores à escolha do utilizador.
Importar um ficheiro CSV fornecido	1. Ir à página de importação de dados em massa; 2. Importar o ficheiro e escolher as opções correspondentes; 3. Fazer o mapeamento entre as informações do ficheiro e os campos da base de dados.

Tabela 3 - Exemplos de testes de alto nível realizados à aplicação

Tendo em conta os resultados obtidos nos vários testes, eram realizadas, se necessário, as mudanças e correções que se considerassem relevantes tendo em conta o desempenho do utilizador, nomeadamente na dificuldade e tempo da realização das tarefas. Uma vez que a maior parte das funcionalidades do ACCEPT Web existe na aplicação *desktop* já existente, tentou-se sempre que possível manter a interface intuitiva e similar de forma que a transição entre as aplicações fosse relativamente fácil.

5.3 Testes em clientes Sinmetro

Tal como estava planeado desde o início do estágio, foi realizada uma deslocação a um cliente da SINMETRO para testar o funcionamento do ACCEPT Web. Como tal, foi feita uma deslocação à fábrica da empresa NovaDelta [50] em Campo Maior no dia 30 de março de 2016. Esta deslocação teve como finalidade testar uma parte das funcionalidades do ACCEPT Web, que se consideravam relevantes no ambiente real testado.

Durante o dia foram feitas algumas correções à aplicação, bem como a implementação de algumas funcionalidades novas identificadas na altura. Apesar de se terem atingidos os objetivos inicialmente propostos, esta visita ficou no entanto marcada por alguns problemas encontrados na receção de dados das balanças, uma vez que os valores por elas enviados não eram os esperados, resultando na apresentação de valores irreais no ACCEPT Web. Ficou no entanto implementada na empresa uma versão do ACCEPT Web, sendo que no decorrer das semanas seguintes foram realizadas várias correções e mudanças que se consideraram necessárias para o correto funcionamento da aplicação.

6 – Conclusão

Fazendo um balanço geral do trabalho desenvolvido durante os 9 meses de estágio, considero que o projeto se encontra praticamente no estado idealizado no início do seu desenvolvimento.

Tendo em conta o planeamento do projeto realizado na fase inicial, considero cumpridos os objetivos propostos, excetuando o de "Disponibilização do projeto na *Web*". A razão pela qual este objetivo não foi cumprido, deve-se ao facto não terem sido descritas explicitamente as funcionalidades a implementar. Uma vez que o planeamento apenas contemplava tarefas genéricas (como implementação de *back-end* e *front-end*), foi deixado em aberto o leque de possibilidades para as várias funcionalidades a implementar. Isto levou a que tenham sido desenvolvidas as funcionalidades sem que se tenha definido em que ponto do projeto se deveria proceder à sua disponibilização da *Web*. Considero no entanto que, tendo em conta o estado do projeto no final do estágio, este estaria pronto para ser disponibilizado na *Web* com as funcionalidades implementadas até à altura. Uma das causas para o atraso na disponibilização na *Web* foi relativa à falta de testes em clientes reais, sendo este um fator alheio aos intervenientes do projeto, já que estavam condicionados à disponibilidade dos clientes.

Em relação às dificuldades encontradas no desenvolvimento do projeto, considero que os maiores desafios se deveram à falta de conhecimentos específicos, principalmente a nível de linguagens *front-end* bem como ao nível de conceitos de metrologia. Houve também algumas dificuldades no planeamento inicial do projeto, uma vez que não foi possível realizar um levantamento de requisitos tão específico quanto o idealizado (visto que as tarefas existentes no inicio do estágio eram bastante genéricas e iam sendo especificadas conforme a necessidade da SINMETRO).

Já os planos para o futuro do projeto consistem em continuar a desenvolver novas funcionalidades e a melhorar as já existentes, tendo em conta os requisitos da Sinmetro e dos seus clientes, bem como a sua disponibilização na *Web* quando se considerar adequado.

Bibliografia

[1] R. Bourdon, “WampServer,” 12 09 2015. [Online]. Available: http://www.wampserver.com/en/.

[2] L. Aferymed - Aferição e Medidas. [Online]. Available: http://www.aferymed.pt/. [Acedido em 13 01 2016].

[3] “Sublime Text,” [Online]. Available: http://www.sublimetext.com/2. [Acedido em 10 12 2015].

[4] T. Otwell, “Laravel PHP Framework Github,” [Online]. Available: https://github.com/laravel/laravel. [Acedido em 10 12 2015].

[5] M. Bean, em *Laravel 5 Essentials*, Packt, 2015, p. v.

[6] “Google Trends,” Google, [Online]. Available: https://www.google.pt/trends/. [Acedido em 10 12 2015].

[7] “Guzzle, PHP HTTP client,” [Online]. Available: https://github.com/guzzle/guzzle . [Acedido em 09 02 2015].

[8] “Date Range Picker,” [Online]. Available: http://www.daterangepicker.com/ . [Acedido em 12 12 2015].

[9] “Moment.js,” [Online]. Available: http://momentjs.com/. [Acedido em 12 12 2015].

[10] SpryMedia Ltd, “DataTables Table plug-in for jQuery,” [Online]. Available: https://www.datatables.net. [Acedido em 14 12 2015].

[11] The jQuery Foundation, “jQuery,” [Online]. Available: https://jquery.com. [Acedido em 10 12 2015].

[12] “Flot: Attractive JavaScript plotting for jQuery,” [Online]. Available: http://www.flotcharts.org/. [Acedido em 30 12 2015].

[13] “Flot,” [Online]. Available: https://github.com/flot/flot . [Acedido em 30 12 2015].

[14] Codicode, “jQuery Flot Animator,” [Online]. Available: http://www.codicode.com/art/jquery_flot_animator.aspx . [Acedido em 30 12 2015].

[15] “jquery-plugin-circliful,” [Online]. Available: https://github.com/pguso/jquery-plugin-circliful. [Acedido em 30 12 2015].

[16] “Bootstrap Get Started,” [Online]. Available: http://www.w3schools.com/bootstrap/bootstrap_get_started.asp. [Acedido em 10 01 2016].

[17] “Laravel Excel Documentation,” [Online]. Available: http://www.maatwebsite.nl/laravel-excel/docs. [Acedido em 15 02 2016].

[18] “Usage of JavaScript libraries for websites,” [Online]. Available: http://w3techs.com/technologies/overview/javascript_library/all. [Acedido em 15 01 2016].

[19] “Move Work Forward - Asana,” [Online]. Available: https://asana.com. [Acedido em 26 01 2016].

[20] “Where to setLocale in laravel 5 on multilingual - multidomain app,” [Online]. Available: https://laracasts.com/discuss/channels/general-discussion/where-to-setlocale-in-laravel-5-on-multilingual-multidomain-app. [Acedido em 01 02 2016].

[21] “The Best PHP Framework for 2015: SitePoint Survey Results,” [Online]. Available: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/. [Acedido em 10 12 2015].

[22] “jquery.localize.js,” [Online]. Available: https://github.com/coderifous/jquery-localize. [Acedido em 08 01 2016].

[23] “Bootstrap · The world's most popular mobile-first and responsive front-end framework,” [Online]. Available: http://getbootstrap.com/ . [Acedido em 10 01 2016].

[24] “Architecture of Laravel Applications,” [Online]. Available: http://laravelbook.com/laravel-architecture/. [Acedido em 10 12 2015].

[25] L. SINMETRO-SISTEMAS DE INOVAÇÃO EM QUALIDADE E METROLOGIA, “Sinmetro,” 2016. [Online]. Available: http://www.sinmetro.pt/. [Acedido em 01 10 2015].

[26] L. SINMETRO-SISTEMAS DE INOVAÇÃO EM QUALIDADE E METROLOGIA, “Sinmetro,” [Online]. Available: http://www.sinmetro.pt/?l=pt&p=16&sm=32&area=2. [Acedido em 01 10 2015].

[27] Instituto Português da Qualidade, Vocabulário Internacional de Metrologia (VIM), Caparica, 2012.

[28] INTERNATIONAL ORGANIZATION OF LEGAL METROLOGY, Quantity of product in prepackages, 2004.

[29] Aferymed, “Pré-embalados,” [Online]. Available: http://www.aferymed.pt/index.php?option=com_content&view=category&id=82&Itemid=439. [Acedido em 02 03 2016].

[30] “Diário da República, 1.ª série — N.º 195 — 8 de Outubro de 2008,” *Diário da República,* pp. 7133-7136, 2008.

[31] Instituto Português da Qualidade, “Microsoft PowerPoint - pre-embalados_madeira1,” 2010. [Online]. Available: http://www.qualidademadeira.com.pt/ficheiros/documentos/Seminarios/filipe_pinto_machado.pdf. [Acedido em 02 03 2016].

[32] “P1198.pdf,” 18 12 1991. [Online]. Available: http://www.aferymed.pt/images/legislacao/P1198.pdf. [Acedido em 02 03 2016].

[33] SP Technical Research Institute of Sweden, “e-marke.gif (475×429),” [Online]. Available: http://www.sp.se/sv/units/measurement/PublishingImages/mass/e-marke.gif. [Acedido em 23 03 2016].

[34] KERN & SOHN, “FPVO-Waage FKTF - KERN & SOHN GmbH,” [Online]. Available: http://www.kern-sohn.com/shop/pt/balancas-industriais/balancas-para-o-controlo-de-prod-pre-embalados/FKTF/ . [Acedido em 11 03 2016].

[35] Mediaweb, “Mediaweb | Homepage,” [Online]. Available: http://mediaweb.pt/. [Acedido em 15 09 2015].

[36] “Software Prototyping,” 23 07 2008. [Online]. Available: http://pt.slideshare.net/drjms/software-prototyping. [Acedido em 19 05 2016].

[37] “SDLC - Software Prototype Model,” [Online]. Available: http://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm. [Acedido em 19 05 2016].

[38] “Teach-ICT A2 Level ICT OCR exam board - The System Life Cycle, prototyping, rapid application development, RAD,” [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/331_systems_cycle/prototyping_RAD/miniweb/pg4.htm. [Acedido em 29 05 2016].

[39] “Teach-ICT A2 Level ICT OCR exam board - The System Life Cycle, prototyping, rapid application development, RAD,” [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/331_systems_cycle/prototyping_RAD/miniweb/pg3.htm. [Acedido em 29 05 2916].

[40] I. Media, “Incremental Prototyping,” 2007. [Online]. Available: http://www.sqa.org.uk/e-learning/IMAuthoring01CD/page_09.htm. [Acedido em 29 05 2016].

[41] S. Komatineni, “Reshaping IT Project Delivery Through Extreme Prototyping,” 11 15 2006.

[Online]. Available: http://archive.oreilly.com/pub/a/onjava/2006/11/15/reshaping-it-project-delivery-through-extreme-prototyping.html . [Acedido em 29 05 2016].

[42] T. Otwell, “http://laravel.com/,” [Online]. Available: http://laravel.com/. [Acedido em 10 12 2015].

[43] “GitHub - kEpEx/laravel-crud-generator: php artisan command to generate fully working crud with grid paginated server side only by having database tables,” [Online]. Available: https://github.com/kEpEx/laravel-crud-generator. [Acedido em 31 03 2016].

[44] T. Otwell, “Artisan Console - Laravel - The PHP Framework For Web Artisans,” [Online]. Available: https://laravel.com/docs/5.1/artisan. [Acedido em 31 03 2016].

[45] Microsoft, “Download Microsoft Drivers for PHP for SQL Server from Official Microsoft Download Center,” [Online]. Available: https://www.microsoft.com/en-us/download/confirmation.aspx?id=20098. [Acedido em 21 06 2016].

[46] T. L. o. E. Packages, “GitHub - thephpleague/csv: CSV data manipulation made easy in PHP,” [Online]. Available: https://github.com/thephpleague/csv. [Acedido em 08 04 2016].

[47] R. Sayar, “Top JavaScript Frameworks, Libraries and Tools and When to Use Them,” 23 11 2015. [Online]. Available: https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/. [Acedido em 30 07 2016].

[48] L. Konstantin, “GitHub - RubaXa/Sortable: Sortable — is a JavaScript library for reorderable drag-and-drop lists on modern browsers and touch devices. No jQuery. Supports Meteor, AngularJS, React, Polymer, Knockout and any CSS library, e.g. Bootstrap.,” [Online]. Available: https://github.com/RubaXa/Sortable. [Acedido em 31 01 2016].

[49] K. Hartl, F. Brack e e. al., “GitHub - js-cookie/js-cookie: A simple, lightweight JavaScript API for handling browser cookies,” [Online]. Available: https://github.com/js-cookie/js-cookie. [Acedido em 22 02 2016].

[50] NOVADELTA – COMÉRCIO E INDÚSTRIA DE CAFÉS, S.A., “Grupo Nabeiro,” [Online]. Available: http://www.grupo-nabeiro.pt/index.php?id=50&page=3. [Acedido em 26 04 2016].

[51] C. Wodehouse, “What is Bootstrap 3? - Front-End Framework - Responsive Mobile First Sites,” [Online]. Available: https://www.upwork.com/hiring/development/bootstrap-3-front-end-framework-responsive-mobile-first-sites/. [Acedido em 30 07 2016].

[52] “Overall equipment effectiveness,” 07 09 2016. [Online]. Available: https://en.wikipedia.org/wiki/Overall_equipment_effectiveness. [Acedido em 12 02 2016].

[53] “GitHub - cquartier/flot.dashes: Dashed line plugin for flot (flotcharts.org), courtesy of michael.hixson@gmail.com,” [Online]. Available: https://github.com/cquartier/flot.dashes. [Acedido em 18 02 2016].

Apêndices

Apêndice A – Diário de atividades da fase inicial do estágio

Titulo: ACCEPT Web - Aplicação web com modelo de negócio SaaS para controlo de enchimento de pré-embalados

1 Setembro - Na primeira metade da manhã estive a conhecer o software já existente da empresa e a ver no que consistia.
Na segunda parte da manhã, até à hora de almoço estive a ler os manuais de utilização do software.
À tarde comecei a instalar o laravel na máquina de testes da empresa e depois foi-me dada a tarefa de começar a fazer as funções que fazem os pedidos e interpretam as respostas da API. No inicio foi-me bastante difícil perceber como fazer a tarefa uma vez que estava habituado a usar a base de dados do laravel e não uma API externa. Após conhecer a ferramenta "Guzzle" foi relativamente fácil fazer a ligação à API.

2 Setembro - Uma vez que já tinha preparado a ligação à API passei o resto do dia a fazer a função dos "lotes". A maior dificuldade foi identificar a maneira de manipular a resposta da API uma vez que não estava habituado a trabalhar com várias camadas de arrays. No final da tarde comecei ainda a fazer a função da "análise geral"

3 Setembro - Durante a manhã concluí a função da "análise geral" que tinha começado no dia anterior. À tarde comecei a trabalhar na vista dos lotes no laravel. Inicialmente não percebi como fazia a "mistura" do HTML com a linguagem Blade dentro do ficheiro PHP, mas os elementos já existentes em HTML foram relativamente fáceis de passar para um formulário Blade. Outro dos problemas que encontrei foi o dos "imports" dos ficheiros JavaScript. Este problema foi ainda mais agravado uma vez que não estava habituado à estrutura de pastas do Laravel 5. No entanto foi fácil resolver o problema após uma pesquisa na internet.
Durante a tarde estive a tentar preencher a tabela da página de lotes a dinamicamente usando a resposta da API. O maior problema nesta fase foi conseguir extrair as informações do pacote pois havia várias camadas de arrays. No entanto apos algumas tentativas foi relativamente fácil perceber como resolver a situação.

4 Setembro - No início da manhã estive a concluir o preenchimento dinâmico da tabela. Depois comecei a tratar da pesquisa por data através dum "datepicker" já existente no template HTML. Durante a tarde consegui por o datepicker a funcionar com a ajuda de um colega, no entanto houve alguns pormenores de otimização/melhoramento estético necessários que não foram implementados uma vez que este "datepicker" era todo baseado um JavaScript, linguagem com a qual não me sinto muito à vontade a trabalhar.

7 Setembro - Durante a manhã continuei a tratar dos problemas do "datepicker" que ficaram da semana anterior. Durante este período não houve progresso uma vez que não consegui perceber como fazer o "datepicker" funcionar de forma desejada devidos aos meus conhecimentos muito limitados de JavaScript.
De tarde comecei a trabalhar a vista da "análise geral" focando-me na funcionalidade dos "agrupamentos".

8-11 Setembro - durante o resto da semana concluí o preenchimento e alguns detalhes nas vistas dos lotes e da análise geral.

14 Setembro - Passei o dia todo a tentar fazer a funcionalidade de fazer download do relatório de lote. Tive bastantes dificuldades uma vez que não sabia como transformar uma stream (devolvida pela API) de volta num ficheiro PDF. Após seguir vários tutoriais e usar várias ferramentas continuei sem conseguir completar a tarefa. No final do dia, um dos colegas ajudou-me e após algumas tentativas consegui-mos finalmente concluir a funcionalidade.

15 Setembro - Durante este dia fiz a funcionalidade de Login. Foi uma funcionalidade relativamente simples de implementar, no entanto para manter as informações do utilizador autenticado decidi colocar as informações do utilizador numa variável de sessão, apesar de não saber se esta seria uma solução recomendável e/ou segura de usar. Houve também outro problema uma vez que a API espera receber a password em texto bruto e não em hash. Uma vez que o laravel não me permite extrair a password sem a encriptar, optou-se por uma solução temporária em que se uma um campo de texto em vez de um campo de password. Sugeri que as passwords fossem encriptadas na API uma vez que apenas posso mandar as passwords encriptadas, no entanto pediram-me que tentasse arranjar outra solução mais tarde. De qualquer forma não me parece muito correto as passwords estarem guardadas em bruto na API, apesar de o responsável não querer mudar.
Sobrou-me também alguem tempo à tarde que usei para começar a implementar a funcionalidade de descarregar as tabelas das vistas "lotes" e "análise geral" para uma ficheiro excel.

16 Setembro - Ao continuar a tarefa de descarregar para excel deparei-me com alguns problemas na parte da exportação. Após seguir alguns tutoriais consegui resolver o problema e exportar os excel corretamente. Reparei depois que o código que tinha feito estava muito mal optimizador uma vez que estava a fazer um segundo acesso à API desnecessário. Resolvi então guardar as informações do primeiro acesso à API numa variável de sessão de forma a poder usa-las na exportação do excel. Após algumas horas a resolver problemas com algumas variáveis consegui finalmente exportar os ficheiros com sucesso completando assim a tarefa. Eventualmente mais tarde serão discutidas possíveis mudanças de forma a formatar o excel para que seja mais fácil de ler e mais visualmente agradável.

17 Setembro - De manhã estive a tratar da sidebar. Estive a mudar a aba "activa" consoante a página em que o utilizador se encontra. Houve alguns problemas nesta simples tarefa uma vez que não conseguia usar código blade na vista. Após tentativa/erro consegui chegar ao resultado pretendido, apesar de haver uma das abas que não ficou a funcionar de forma pretendida.
De tarde comecei por resolver o problema da password do login através da encriptação e desencriptação da password introduzida pelo utilizador. Esta solução, apesar de funcional, deixou-me algumas dúvidas em termos de segurança, no entanto está a funcionar da maneira que foi pedido, uma vez que o responsável do projeto foi abordado em relação a este assunto e decidiu que seria para manter assim. De seguida passei o resto da tarde a olhar para alguns do pormenores deixados por resolver nas tarefas anteriores.

18 Setembro - Durante todo o dia tentei pôr a tabela da vista "analise geral" a carregar através de AJAX e Javascript para corrigir um erro que existia ao carregar os dados na tabela, pois o javascript demorava bastante tempo a carregar e criava um efeito visual estranho para o utilizador (a tabela era toda carregada e apenas no fim era feita a paginação, o que resultava numa expansão da tabela e no fim uma compressão). O objetivo dessa mudança é apresentar uma imagem de "loading" enquanto a tabela é carregada e só depois apresentá-la ao utilizador

21 Setembro - Apesar da tarefa de carregar a tabela da página "analise geral" não estar acabada, foi dada prioridade à tarefa de preenchimento do dashboard.
Durante este dia comecei a fazer o preenchimento do "Cpk" no dashboard. A parte de preenchimento dos valores foi relativamente fácil. O problema que encontrei foi o facto de o "chart" não recarregar após mudar os seus valores com javascript pois os ficheiros de Javascript importados já tinham sido carregados. Apesar de ter tentado mudar as variáveis no ficheiros de configuração dos "charts" e tentado recarregar os ficheiros de Javacript para as mudanças fossem aplicadas, não obtive sucesso. Decidi então copiar o código de construção do "chart" para o ficheiro que trata da chamada AJAX. A partir daí decidi que a melhor solução seria eliminar o conteúdo da " " onde estava inserido o chart e reinseri-lo com as variáveis necessárias. Depois bastaria colocar código co chart com as mudanças necessárias para que o chart fosse novamente criado.

22-24 Setembro - Durante estes 3 dias tratei dos restantes campos do dashboard. Uma vez que já tinha tratado do "Cpk" os restantes campos não foram difíceis de tratar pois o funcionamento dos "charts" era bastante semelhante, pelo que adotei a mesma solução de replicar o código dos "charts". Um dos problemas que encontrei (apesar de ser bastante fácil de resolver) foi o facto de novos campos terem sido acrescentados na API. Uma vez que estes campos foram introduzidos no meio dos já existentes e não no fim, foi necessário mudar algumas linhas de código pois os parâmetros usados estavam agora em posições diferentes do array recebido pela API. No final do dia comecei a trabalhar no último campo, o do histograma, no entanto percebe-mos que os campos devolvidos pela API não estavam corretos.

25 Setembro - Mais uma vez foram introduzidos novos campos na API, no entanto desta vez muitos dos campos mudaram de sítio, pelo que tive mais uma vez de efetuar algumas alterações no código. Depois comecei a trabalhar novamente no histograma do dashboard, no entanto não consegui chegar ao resultado pretendido uma vez que não conseguia apresentar linhas no gráfico de barras.

28 Setembro - Durante a manhã e ao início da tarde estive a pesquisar e experimentar vários tipos de gráficos para representar o histograma no dashboard. Apesar de não ter tido sucesso com nenhum dos gráficos que experimentei, a meio da tarde consegui chegar ao resultado pretendido usando o gráfico inicial com ajuda de um comentário de um fórum de programação. O histograma foi provavelmente a tarefa mais difícil do dashboard uma vez que não conseguia encontrar uma solução que permitisse desenhar linhas verticais num gráfico de barras. O que tornou esta tarefa ainda mais difícil foi o facto de a solução que encontrei não estar documentada na página da API do gráfico que usei.

29 Setembro - Durante a manhã continuei a trabalhar no histograma, tratando de alguns pormenores de representação do gráfico. Trabalhei também noutros pormenores do dashboard que estavam por fazer. No final da tarde comecei a trabalhar numa nova subvista do dashboard na qual é suposto mostrar a mesma informação que já tinha estado a trabalhar mas compilada num cartão mais pequeno, de forma a mostrar um cartão pouco detalhado para cada processo existente.

30 Setembro - Continuei a trabalhar nos cartões genéricos que comecei no dia anterior. No início o maior problema que encontrei foi descobrir como trabalhar com o sistema de cartões em HTML, no entanto foi possível implementar a funcionalidade através de tentativa/erro o que permitiu perceber como funcionava o sistema.

1 Outubro - Uma vez que o sistema de cartões já estava implementado, o passo seguinte foi preencher dinamicamente o cartão da "Vista Legal" com informações "legais" de cada processo. A maior dificuldade encontrada neste passo foi a criação de vários elementos HTML encadeados (nomeadamente "divs"). Após a criação dos cartões de informação, foi necessário acrescentar mais informação aos cartões, o que se tornou um problema uma vez que a informação era demasiada para apresentar num espaço tão pequeno, fazendo com que uma parte do texto ficasse de fora do cartão.

2 Outubro - Este dia foi dedicado, em primeiro lugar, ao preenchimento de dos "lotes com problemas" na vista legal que havia sido começada no dia anterior. Sendo esta uma tarefa relativamente simples, o resto do dia foi dedicado à programação dos botões da "vista legal", de forma ao que clicar no nome do processo o utilizador seja redirecionado diretamente para as informações detalhadas desse processo. Apesar de o inicio ter sido complicado, a tarefa foi concluída através da manipulação dos "ids" dos botões.

5 Outubro - Toda a manhã e metade da tarde deste dia foram usados para a criação de uma nova aba de "vista legal de lotes" onde é apresentada uma lista de cartões com informações legais de todos os lotes existentes. A dificuldade deste passo deveu-se à criação de um botão que deveria fazer o "download" de um ficheiro através de um "request AJAX". Uma vez que não se consegui fazer o "download" do ficheiro, foi decidido que o ficheiro deveria ser aberto numa nova aba, na qual o utilizador poderia então fazer o "download" do ficheiro se pretendesse. Na segunda metade da tarde foram corrigidos alguns problemas com o tamanho dos textos, tornando os elementos HTML adaptáveis ao "zoom" do ecrã.

6 Outubro - Após alguns testes à funcionalidade desenvolvida no dia anterior, houve sucesso em fazer o download do ficheiro através de "AJAX". Ficou então decidido que deveria haver tanto a opção de abrir o PDF numa nova aba como a opção de "download". O resto da manhã e uma grande parte da tarde foi dedicado a corrigir alguns erros e fazer algumas melhorias nas vistas do site. Das 16.15h às 17h houve uma pequena reunião com um dos responsáveis pela criação do "template" usado no site que teve o intuito de tirar algumas dúvidas sobre a utilização de alguns elementos do "template", dando mais enfase nas classes de "bootstrap" que constroem as vistas. Após a reunião já era possível avançar nalguns passos que antes não se conseguia pois já existia um conhecimento mais aprofundado do funcionamento do "template".

7 Outubro -

8 Outubro - Este dia foi dedicado à participação numa formação intitulada "Controlo Metrológico de Pré-embalados?" e foi dirigida pelos oradores Cristina Barros (Sinmetro) e Rui Silva (Aferymed). Esta formação teve uma duração de 7 horas e teve lugar na Nerlei em Leiria. O objetivo desta formação era aprender mais sobre os pré-embalados (nomeadamente questões legais) e como devem ser tratadas num "chão de fábrica".

9 Outubro - Neste dia foi desenvolvida a vista estatística dos processos. O objetivo desta vista é mostrar para cada processo algumas informações estatísticas, bem como os gráficos de tendência e o histograma. Uma vez que

foi necessário "montar" a estrutura desta vista através de classes "bootstrap", apenas foi possível fazer a parte estatística.

12 Outubro - Durante a parte da manhã foi concluída a vista estatística de processos, terminando o gráfico de tendência e o histograma. Após a conclusão desta vista, foi feita uma pequena reunião com os responsáveis pelo projeto de forma a decidir quais os passos seguintes a realizar, bem como algumas mudanças ao que já estava feito. Na parte final da tarde foram feitas algumas mudanças aos tamanhos de letra das várias vistas de forma a coincidir com o tamanho/resolução do ecrã. Apesar de algumas tentativas, não se consegui fazer esta mudança com sucesso uma vez que o tamanho de letra não ficava como desejado ao mudar a resolução do ecrã. No decorrer deste dia foi também descoberto um problema com a funcionalidade de "download" do pdf dos relatórios de lote. Este erro parecia estar diretamente relacionado com o browser "google chrome". Após várias tentativas houve alguns sucessos, sendo que aparentemente algumas vezes a funcionalidade funcionava corretamente e outras vezes não. Este erro parece estar relacionado com os "headers" do pacote enviado, sendo que o "chrome" não interpreta o pacote da mesma maneira que os outros "browsers".

13 Outubro - Em primeiro lugar foi feita a funcionalidade de mostar as informações para cada uma das linhas. Uma vez que esta funcionalidade era praticamente igual à dos processos, decidiu-se usar o mesmo cartão que os processos, redesenhando o cartão sempre que se quer ver uma linha nova. Foi também feita a funcionalidade de mostrar apenas os lotes com problemas, sendo que se optou também pela reutilização do cartão já existente de todos os lotes. Esta funcionalidade levantou algumas dúvidas quanto à disposição dos botões para troca de cartões, uma vez que não havia distinção entre os "todos os lotes" e "lotes de um processo". Foi decidido que deveria ser criado um botão para cada funcionalidade durante a fase de desenvolvimento, sendo que mais tarde seria decidida a disposição dos botões. Finalmente, foi começada a funcionalidade de mostrar os lotes com problemas referentes a um só processo. Apenas foi possível criar o botão, sendo que a funcionalidade em si ficou para o dia seguinte.

14 Outubro - Por volta das 9h da manhã houve a primeira reunião entre estagiário e coordenador na ESTG. Em primeiro lugar foi apenas feita uma pequena apresentação sobre o trabalho que estava a ser desenvolvido. Foi também sugerido que fossem experimentadas algumas "frameworks" que poderiam ajudar bastante em algumas funcionalidades do projeto. Por fim foram dadas algumas sugestões para o início da escrita do relatório de estágio. Esta reunião teve uma duração de cerca de 1 hora e 10 minutos. No resto do dia foi desenvolvida a funcionalidade de mostrar os lotes com problemas de um processo que havia sido começada no dia anterior.

15 Outubro - Neste dia foi inicialmente feita uma pesquisa sobre algumas frameworks de Javascript (eg. AngularJS, React, Ember, Jquery). Chegou-se à conclusão que que o uso destas frameworks não iria trazer grandes vantagens considerando o modelo atual no website, no entanto é algo a considerar para o futuro. No resto do tempo foi acabada a funcionalidade dos lotes com problemas e foram corrigidos/melhorados alguns pormenores visuais existentes.

16 Outubro - Foi feita a funcionalidade de vista legal + estatística dos processos. Uma vez que as vistas estatísticas e legais já estavam implementadas, foi possível reutilizar muito do código já feito. Houve num entanto um problema de dimensões ao desenhar os gráficos. Até ao fim do dia não foi possível corrigir este erro.

19 Outubro - Após algumas horas foi finalmente corrigido o erro das dimensões encontrado no dia 16 de Outubro. Descobriu-se que este erro era causado pelo facto de não estarem a ser dadas dimensões ao "container" dos

gráficos, sendo assim necessário forçar o seu tamanho. Na parte final da tarde foi feita uma reunião com duração de cerca de uma hora onde foram discutidas algumas das ideias para a realização da vista de "Amostras e Pesagens".

20 Outubro - Inicialmente foram feitas mudanças nas vistas de forma a que todas apresentassem os elementos da mesma forma. Foram também feitas algumas tentativas para corrigir um erro de visualização dos gráficos (deslocação do gráfico em relação ao "canvas") mas sem sucesso. No resto do dia foi começada a funcionalidade de mostrar as "Amostras e Pesagens" de cada processo.

21 Outubro - Durante este dia foi continuada a funcionalidade de "Amostras e Pesagens". Durante a tarde houve uma pequena discussão com duração de cerca de 30 minutos para decidir como os dados desta vista, bem como da vista detalhada de "Amostras e Pesagens" deviam ser apresentados. De seguidas foram feitas algumas alterações ao que já estava feito para coincidir com o que foi decidido.

22 Outubro - Foi começada a funcionalidade de mostrar as pesagens de cada amostra. Uma vez que a forma de mostrar a informação já tinha sido discutida, o único problema encontrado foi a maneira de conseguir representar uma tabela dentro de outra tabela e como fazer uma acão através do clique numa linha da tabela. Após uma rápida pesquisa foi relativamente simples resolver o problema de forma a apresentar a tabela de forma correta dentro de um elemento de outra tabela.

23 Outubro - Ao longo deste dia foi continuada e terminada a funcionalidade de mostrar as pesagens de cada amostra.

26 Outubro - De manhã foram foi acabada a vista de "Amostras e Pesagens" simples. Da parte da tarde foram foi começada a vista de "Amostras e Pesagens" detalhada.

27 a 30 de Outubro - Foi acabada a vista de "Amostras e Pesagens", bem como refeitos alguns detalhes na vista "Amostras e Pesagens" simples. Foi também feita uma alteração da ordem de representação das tabelas, de forma a que as pesagem aparecessem primeiro por serem mais importantes. Ao longo destes dias foi também feito um esboço do dashboard de linhas.

2 Novembro - Foi começada a funcionalidade de filtrar as tabelas de "Amostras e Pesagens", escondendo ou mostrando as colunas conforme a preferência do utilizador através do uso de checkboxes.

3 de Novembro - Durante a manhã foi acabada a funcionalidade de filtrar as tabelas de "Amostras e Pesagens". Esta tarefa apresentou mais dificuldades do que o inicialmente esperado, principalmente por causa dos headers da tabela, uma vez que ao esconder a primeira coluna de cada header, o header era também escondido. Durante a tarde foi recomeçada a funcionalidade do dashboard de linhas que tinha sido começada na semana anterior.

4 a 11 de Novembro - Foi terminado o dashboard de linhas.

12 a 30 de Novembro - Durante estes dias foram feitas várias melhorias a algumas funcionalidades já implementadas. Foram também implementados os filtros da vista de "Análise Geral" e dos "Lotes". A maior dificuldade neste passo foi a escolha da API a utilizar para fazer os filtros com autocomplete, uma vez que foram testados vários e o resultado não foi satisfatório. No final, a solução implementada, apesar de funcional, apresentava alguns problemas com a funcionalidade de "autocomplete" uma vez que nem sempre preenchia bem a "dropdown" para a escolha do item desejado.

1 de Dezembro - Da parte da manhã foi feita alguma pesquisa sobre a funcionalidade multilingue. Das 12h10 às 13h20 houve a segunda reunião com o coordenador de estágio, que serviu maioritariamente para discutir a estrutura do relatório de estágio. Da parte da tarde foi começada a funcionalidade multilingue entre português e inglês.

2 e 3 de Dezembro - Foi feita a tradução em português e inglês das paginas de login e do dashboard. Ficaram a faltar as restantes páginas bem como os ficheiros de javascript. Até este dia não havia informação sobre quais as linguagens a implementar, sendo numa fase inicial apenas importante o português e o inglês. Durante estes 2 dias foram também feitas várias mudanças à tabela de "Análise Geral" e consequentemente a funcionalidade de exportar a grelha.

Período de 4 de Dezembro a 25 de Janeiro - Durante este período foram feitas várias mudanças e melhoramentos a muitas das funcionalidades já existentes, principalmente o suporte multilingue e reestruturação do mecanismo do Laravel para seguir o padrão arquitetural MVC. Durante este tempo foi também implementada a funcionalidade da carta de controlo, sendo que nesta foram encontradas algumas dificuldades na sobreposição de labels nos eixos do gráfico.