Soluções de apoio à gestão das empresas:

ERP adaptado & WMS à medida

Mestrado em Engenharia Informática – Computação Móvel

Mariana de Jesus Elói

Leiria, outubro de 2021

Soluções de apoio à gestão das empresas:

ERP adaptado & WMS à medida

Mestrado em Engenharia Informática – Computação Móvel

Mariana de Jesus Elói

Estágio realizado sob a orientação da Professora Doutora Eugénia Moreira Bernardino e sob supervisão de Rui Pinto.

Leiria, outubro de 2021

Originalidade e Direitos de Autor

O presente relatório de estágio é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionada a Autora e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Mestrado em Engenharia Informática – Computação Móvel, no ano letivo 2020/2021 da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Agradecimentos

Gostaria de agradecer a toda a equipa Arentia, que me acolheu durante o estágio. Ao meu supervisor pela oportunidade dada e aos colegas com quem trabalhei diretamente nos vários projetos por toda a ajuda prestada. A aprendizagem foi imensa e todos foram importantes para esse processo.

Gostaria ainda de agradecer à minha orientadora, professora Eugénia Bernardino, pelo tempo e ajuda fornecida ao longo dos meses. Mesmo o meu estágio não tendo sido acompanhado desde o início, o apoio fornecido foi fundamental para a elaboração do relatório.

Por fim, gostaria ainda de agradecer aos meus pais e ao meu namorado por todo o apoio prestado durante estes últimos anos de estudo. Sem eles, teria sido mais difícil ultrapassar algumas das dificuldades.

Resumo

Este relatório descreve o trabalho realizado durante o Estágio curricular do Mestrado em Engenharia Informática – Computação Móvel, lecionado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

O objetivo deste relatório é descrever os projetos realizados durante os nove meses de estágio realizados na empresa Arentia. Esta é uma empresa que fornece *software* na área da gestão, assim como desenvolve aplicações que ajudam os clientes a simplificar os processos das suas empresas.

Neste relatório é realizado um enquadramento do estágio, é descrita a metodologia utilizada, são apresentadas as tecnologias, as arquiteturas, os dois projetos desenvolvidos, os testes realizados e por fim as conclusões.

Os principais objetivos do estágio foram acompanhar e concluir o desenvolvimento de um projeto e iniciar um projeto novo. O acompanhamento do primeiro projeto, relacionado com o *PHC Web*, recaiu sobre a implementação de pequenos pedaços de código que ajudassem a minimizar o erro humano e a simplificar processos na criação de documentos de faturação na plataforma em questão.

O segundo projeto consistiu na implementação de uma aplicação *web*, que permitisse a gestão de armazéns. Foi necessário estudar a aplicação já existente e adaptá-la de acordo com os novos requisitos, definidos em conjunto com o cliente. Houve ainda a necessidade de garantir que as regras definidas eram cumpridas, de forma semelhante, na aplicação *web* e no *PHC*, uma vez que estas plataformas têm algumas funcionalidades em comum (em diferentes fases do processo).

O estágio permitiu adquirir mais experiência e competências, quer profissionais, quer relacionais. O desenvolvimento de duas aplicações com bases tão distintas permitiu explorar novos conhecimentos e desenvolver o raciocínio lógico para a criação de aplicações coerentes. Um dos projetos foi colocado em produção durante o decorrer do estágio e o outro encontra-se em fase de testes no cliente.

Palavras-chave: *PHC*, aplicações *web*, *software* padronizado, *software* à medida

Abstract

This report describes the work realized during the Master's Degree curricular internship in Computing Engineering – Mobile Computing, taught in Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

The objective of this report is to describe the projects developed during the nine months of the internship in the company Arentia. This is a company that provides software in the management area as well as developing applications, which help clients to simplify the processes of their companies. In this report is contextualized the internship, is described the methodology used, presented the technologies, architectures and implementations from both developed projects, is described the tests performed and, finally, are presented the conclusions.

The main goals of the internship were to follow and finish the development of a project and to start a new one. The follow-up of the first project, related to *PHC Web*, summed up the implementation of little pieces of codes that helped to minimize human mistakes and simplify procedures in the creation of billing documents.

The second project consists in the implementation of a web app, that would allow Warehouse Management. It was necessary to study the existing app and adapt it according to the new requisites, defined together with the client. There was still a necessity in ensuring that the rules previously defined were fulfilled, in a similar way, in the web and *PHC* app, since these platforms have a few functionalities in common (in different phases of the process).

The internship allowed to acquire more experience and competencies, whether professional or rational. The development of both apps with such distinct bases allowed to explore new knowledge and develop logical reasoning towards the creation of coherent applications. One of the projects was placed in production during the internship and the other one is going through a testing phase at the client.

Keywords: *PHC*, Web applications, Packaged software, Custom software

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
CRUD	*Create, Read, Update and Delete*
DOM	*Document Object Model*
EDM	*Entity Data Model*
ERP	*Enterprise Resource Planning*
ESTG	Escola Superior de Tecnologia e Gestão
HTML	*HyperText Markup Language*
IDE	*Integrated development environment*
IIS	*Internet Information Services*
MVC	*Model-View-Controller*
ORM	*Object-Relational Mapping*
SaaS	*Software as a Service*
SQL	*Structured Query Language*
TI	*Tecnologias Informáticas*
US	*User Stories*
VPN	*Virtual Private Network*

1. Introdução

O presente relatório tem como objetivo apresentar todo o trabalho realizado durante o estágio, realizado no âmbito da unidade curricular de Estágio, do curso de Mestrado em Engenharia Informática – Computação Móvel, lecionado na Escola Superior de Tecnologia e Gestão (ESTG). O relatório irá abordar as tarefas desenvolvidas no decorrer dos dois projetos desenvolvidos, assim como todos os processos relevantes durante o desenvolvimento dos mesmos.

Este capítulo apresenta uma breve introdução sobre o estágio. Primeiro, serão apresentados os motivos e objetivos do estágio (secção 1.1), seguindo-se um cronograma (secção 1.2) e a apresentação da estrutura do relatório (secção 1.3).

1.1. Motivação e objetivos

O motivo que originou o estágio curricular, no âmbito do mestrado, deveu-se à oportunidade oferecida pela empresa Arentia, uma vez que já tinha existido um contacto prévio, durante o ano letivo anterior. O que motivou a aceitar o convite, prende-se com o facto de saber quais as tecnologias que a empresa utiliza e sentir que seria uma área onde existe imenso por onde evoluir. Além da oportunidade de trabalhar com várias tecnologias, havia ainda a hipótese de aprender mais sobre temas não-relacionados com a programação, como é o caso da Gestão.

Os principais objetivos do estágio incluíam continuar o acompanhamento de um projeto iniciado anteriormente, assim como ser responsável por outro projeto. O estágio teria ainda como objetivos a aplicação e ampliação dos conhecimentos adquiridos ao longo da licenciatura, do mestrado e de experiências anteriores, assim como desenvolver novas práticas de trabalho.

O trabalho desenvolvido durante o estágio assenta em dois projetos muito diferentes. O primeiro projeto que será abordado, foi o projeto já iniciado aquando do início do estágio e que tem como base uma aplicação *web* já existente, para gestão da faturação de uma empresa. O segundo projeto foi um projeto que tinha sido iniciado por ex-colaboradores e que tem como objetivo permitir gerir um armazém de uma empresa. O envolvimento em dois projetos

tão diferentes permitiu obter a experiência de adaptar um *software* padronizado às necessidades do cliente, assim como a experiência de criação de um *software* à medida.

1.2. Cronograma

O estágio foi realizado durante o ano letivo 2020/2021, tendo a duração de nove meses (com data de início a 24-08-2020 e data de fim a 23-04-2021). Este foi realizado na empresa Arentia, S.A. (ver secção 2.1), sediada em Leiria, e que fornece *software* de faturação, assim como soluções desenvolvidas pela própria empresa.

Na Figura 1.1 é possível consultar o cronograma das tarefas desenvolvidas durante o estágio.

Figura 1.1 – Cronograma das tarefas desenvolvidas durante o estágio

As primeiras dez semanas foram dedicadas ao desenvolvimento do projeto já iniciado, sendo que destas, as últimas cinco já com o cliente a realizar testes e a fornecer *feedback*. As cinco semanas seguintes serviram para colocar o projeto em produção e dar apoio inicial, para resolução de situações específicas.

Entre a segunda semana de dezembro e a primeira semana de janeiro, o tempo foi dedicado ao levantamento de requisitos do novo projeto, assim como à análise do que já estava desenvolvido na aplicação. Em simultâneo e até à terceira semana de março foram implementadas as funcionalidades requeridas. Na segunda semana de março, foi colocada uma versão do projeto no cliente, para que este pudesse fazer testes e reportar eventuais alterações ou correções. Nas semanas seguintes e até ao final do estágio foram realizadas correções ou melhorias reportadas pelo cliente, assim como uma reunião para estabelecer o ponto de situação.

Na terceira semana foram realizados testes relacionados com o projeto *PHC Web*, uma vez que havia a necessidade de fazer *upgrade* à versão que o cliente tinha instalada.

Além destes dois projetos, existiram também tarefas relacionadas com o apoio a clientes de uma outra aplicação, que não são abordadas neste relatório.

1.3. Estrutura do relatório

Este documento encontra-se estruturado em sete capítulos, sendo o primeiro o capítulo atual. Neste foi realizada a introdução do documento, assim como apresentados os objetivos e o cronograma do estágio.

No segundo capítulo (Enquadramento do estágio) é apresentada a entidade onde foi realizado o estágio, assim como o enquadramento dos projetos. Neste capítulo é ainda feita uma comparação entre o *software* padronizado e o *software* desenvolvido à medida.

No terceiro capítulo (Metodologias de desenvolvimento) é realizada uma breve apresentação da metodologia utilizada pela entidade.

No quarto capítulo (Desenvolvimento – Projeto *PHC Web*) e no quinto capítulo (Desenvolvimento – Projeto *aWMS*) são apresentados os projetos realizados durante o estágio. Nestes capítulos são apresentadas as arquiteturas e tecnologias utilizadas nos projetos, assim como o modelo de negócio do cliente e a implementação das soluções.

No sexto capítulo (Testes) são apresentados os testes realizados e é descrita a forma como foram aplicados em cada projeto.

Por último, no sétimo capítulo (Conclusão) é realizada uma breve conclusão sobre o trabalho realizado durante o estágio.

2. Enquadramento do estágio

Neste capítulo é realizado um enquadramento do estágio, assim como são abordados alguns dos conceitos mais relevantes. Primeiramente é apresentada a entidade de acolhimento (secção 2.1), seguindo-se de um enquadramento aos projetos realizados (secção 2.2) e, por fim, uma análise sobre as diferenças entre *software* padronizado e *software* à medida (secção 2.3).

2.1. Entidade de acolhimento

A Arentia, S.A. é uma empresa sediada em Leiria, fundada em 2008, tendo como principais mercados as indústrias dos moldes, plásticos, metalomecânica, construção metálica, entre outros. É uma empresa que fornece *software* de gestão, assim como soluções de *hardware* [1].

O principal objetivo da Arentia é fornecer soluções simples, que vão de encontro às necessidades apresentadas pelo cliente e que o ajudem a melhorar os seus processos. Sendo parceira de dois *Enterprise Resource Planning* (ERP), oferece soluções para gestão, assim como plataformas desenvolvidas pela própria empresa para ajudar os seus clientes a aumentar a sua produtividade [2].

A empresa é constituída por cerca de 50 pessoas, havendo colaboradores nas áreas de gestão, programação, tecnologias informáticas (TI), comercial e administração. A maior fatia dos colaboradores está relacionada com as áreas tecnológicas (programação e tecnologias informáticas). A empresa está organizada por departamentos, estando cada um deles organizado por equipas, podendo estar membros de diferentes equipas alocados a um mesmo projeto.

Dada a situação de saúde pública existente durante a realização do estágio[1], não era possível a convivência em massa por parte de todos os colaboradores, sendo o teletrabalho a forma preferencial de trabalho. Mesmo à distância, havia um sentido de interajuda entre colaboradores do mesmo departamento e até de diferentes departamentos, quando necessário e possível.

[1] Estágio realizado durante a pandemia da Covid-19, onde era aconselhado a prática de teletrabalho quando possível.

2.2. Enquadramento aos projetos

Tendo em consideração o que já foi referido no subcapítulo 1.1, este relatório contempla o trabalho realizado para os dois diferentes projetos desenvolvidos durante o estágio.

O primeiro projeto que será descrito será o Projeto *PHC Web*, que assenta na adaptação de um *software* padronizado, neste caso um ERP, às necessidades do cliente. De forma geral, este projeto consistiu na customização do ERP adquirido, de forma a ir de encontro aos requisitos apresentados pelo cliente. Houve também algumas customizações feitas para auxiliar o utilizador com o seu manuseamento e diminuir a probabilidade de erro ao preencher documentos.

Os ERP são "sistema integrados que possibilitam o fluxo de informação entre os diversos sectores do negócio" [3], permitindo ter acesso à informação em tempo real. Este tipo de *software* permite ter toda a informação centralizada num único local, permitindo aceder à informação sobre diversas áreas (gestão, contabilidade, recursos humanos) através de um único *software* [3].

Os ERP são implementados de acordo com os processos já definidos pela empresa, podendo ser utilizado o *software* padrão, em que é a empresa que tem de se adaptar ao *software*, ou customizado, em que o *software* é adaptado à empresa, através de desenvolvimentos realizados por técnicos certificados do *software* [3] [4].

Os ERP são constituídos por módulos, permitindo às organizações crescer com o mesmo ERP e a mesma base de dados. Uma organização pode começar por adquirir apenas os módulos das áreas mais relevantes (por exemplo, o módulo de gestão) e depois adquirir os módulos para outras áreas (por exemplo, recursos humanos ou gestão de relacionamento com o cliente) [3].

A implementação de um ERP numa empresa apresenta alguns benefícios, como padronização, redução de tarefas repetitivas, melhoramento dos processos, fiabilidade de dados, entre outras; assim como tem desvantagens, como a possibilidade de existir um erro que afeta todo o processo, custos de manutenção e atualização, resistências dos colaboradores, dependência do fornecedor, entre outros [4].

De acordo com um pequeno estudo realizado em 2012, com uma amostra de 32 empresas, em média, as empresas tinham um ERP há cerca de 18 anos, implementando, em média, dois

ERP durante este período [5]. O sistema mais utilizado na amostra é o *Navision,* da *Microsoft* (46,9%), seguindo-se o *PHC* (12,5%) [5].

O segundo projeto, o Projeto *aWMS,* é uma aplicação desenvolvida à medida do cliente. De modo geral, este projeto consistiu na elaboração de uma aplicação que permitisse a gestão de armazéns, realizada pelo departamento de Qualidade. Além da gestão de armazém, tinha também de ser possível o controlo de qualidade do material produzido (aprovação, rejeição, verificação, entre outros).

No caso deste projeto, o mesmo foi começado partindo de um desenvolvimento base (iniciado por outros ex-colaboradores da empresa, como já anteriormente referido), e foi alterado para ir de encontro às necessidades do cliente, sendo por isso um *software* à medida. Havendo já processos bem definidos no cliente, esta aplicação teria de ir de encontro aos processos já existentes, podendo apenas serem sugeridas alterações que facilitassem processos ou o manuseamento por parte dos utilizadores.

2.3. *Software* padronizado vs. *Software* à medida

Tendo como base os projetos desenvolvidos, tornou-se relevante procurar perceber o que se conhece sobre estes dois tipos de *software*: padronizado e desenvolvido à medida. Assim, foi realizada uma revisão para perceber o que existe na literatura sobre este tema.

Um *software* padronizado é um "*software product that is released for and traded in a specific market*" [6], como é o caso dos *softwares* de gestão utilizados nas empresas; estes são *softwares* já desenvolvidos e que são disponibilizados ao seu público-alvo [6]. Por sua vez, um *software* desenvolvido à medida é "*highly specific software which is made to fit workflow of each user*" [7], ou seja, é um *software* desenvolvido de acordo com o que o cliente precisa, sendo necessárias alterações para ser aplicado noutro cliente [7].

O *software* padronizado passou a ser uma grande parte do mercado de *software* desde os finais dos anos 80, alterando os processos de desenvolvimento de *software* conhecidos até ao momento, uma vez que estes se focavam na criação de aplicações para um cliente [8]. Este crescimento deve-se, maioritariamente, às aplicações modulares, como acontece com os ERP (já abordados no subcapítulo 2.2) [9]. Carmel em [8] atribui cinco características ao *software* padronizado, sendo elas:

1. *Adaptar ao "cliente desconhecido"*: uma vez que não se conhece o utilizador, não se consegue obter *feedback* durante todo o ciclo de desenvolvimento até o produto ser disponibilizado. Após a sua publicação, é que a empresa responsável pelo seu desenvolvimento pode disponibilizar canais de comunicação para obter *feedback* dos utilizadores, seja para sugestões ou reportar erros.
2. *Dificuldades na formalização de requisitos*: não existindo um cliente conhecido, o levantamento de requisitos torna-se mais complicado, sendo necessário recorrer a outros métodos para tentar obter o máximo de informação possível. No entanto, existe uma maior proporcionalidade em especificações incorretas ou completas, havendo necessidade de serem lançadas novas funcionalidades ou melhorias nas versões seguintes.
3. *Suportar múltiplas plataformas e múltiplas versões*: as empresas que produzem este tipo de *software* têm a necessidade de garantir que o seu produto suporta várias plataformas/sistemas operativos, tornando os seus produtos moduláveis. Também precisam de manter disponíveis múltiplas versões, uma vez que o utilizador pode não querer migrar para uma nova versão.
4. *Facilitar a velocidade de desenvolvimento, ajustar-se à janela de oportunidade e atender à pressão do mercado*: as equipas de desenvolvimento deste tipo de produto sofrem uma certa pressão para cumprir prazos, uma vez que o mercado é competitivo e as janelas de oportunidade são curtas. Como as tecnologias estão em constante evolução, existe uma necessidade de acompanhar tendências e atender as necessidades dos utilizadores, antes de estes mudarem de fornecedor.
5. *Incorporar minimização de riscos*: o risco de falha neste tipo de *software* é superior ao *software* à medida, o que pode resultar em perdas significativas para a empresa responsável pela criação do *software*. O principal objetivo é reduzir o capital aplicado através de avaliações de mercado, como ajuda na hora de decidir se o produto deve ou não ser lançado naquela fase.

Assim, podem ser já definidos alguns pontos onde o *software* padronizado e o *software* à medida divergem, como o facto de o *software* padronizado ser mais orientado ao produto e mais personalizável do que um *software* à medida. Existe ainda o papel do utilizador no processo de desenvolvimento: no primeiro caso, o utilizador não é parte integral do processo, enquanto que no *software* à medida este é um elemento relevante para uma implementação

de sucesso. Outro ponto diferencial entre ambos são as técnicas, métodos e ferramentas utilizadas durante o processo de desenvolvimento [9].

Carmel em [10] afirma que o modelo de desenvolvimento em cascata não é indicado para o desenvolvimento do *software* padronizado, uma vez que é um modelo que necessita dos requisitos totalmente elaborados e fechados para avançar para a próxima fase, aspeto que não é possível realizar nos *softwares* padronizados, visto que existe uma percentagem dos requisitos que apenas é possível definir depois de obter *feedback* dos utilizadores. O autor sugere que um modelo indicado seria o modelo em espiral [8], uma vez que este tipo de modelo permite regressar a fases anteriores no fim de um ciclo completado, tendo sempre uma versão intermédia para apoio na análise e *design* da nova etapa [10]. Contudo, o desenvolvimento contínuo também se adapta a este tipo de produto, com algumas diferenças em relação ao *software* à medida.

No que se refere ao *software* à medida é preferível a utilização de modelos incrementais ou evolucionários (e.g. modelo em espiral) em vez de um modelo em cascata, sendo que há a necessidade de a lista de requisitos ser revista e ajustada conforme o projeto avança, adaptando-se à complexidade e a questões que não tinham sido previstas [11]. Um modelo incremental é uma adaptação de um modelo "em cascata", constituído por várias iterações. Em cada interação é possível entregar uma versão e obter *feedback* que pode ser incorporado em iterações futuras, o que se traduz num maior envolvimento do cliente e identificação de riscos atempadamente [10], [12]. O princípio dos modelos evolucionários é aplicar "*different methods for stages and life-cycle models at different development rounds*" [10], para que em cada ronda de desenvolvimento seja utilizado o método que melhor se aplica naquela fase. No entanto, um dos problemas que este princípio pode levantar é que a consistência entre os diferentes modelos e métodos não é garantida.

Usando em ambos os tipos de *software* o desenvolvimento contínuo, a principal diferença será o início da fase de manutenção, isto porque esta é claramente identificável no *software* à medida (existe um momento em que o produto está terminado), enquanto o *software* padronizado parece nunca sair desta fase (assim que sai uma versão, irá aparecer outra, dando a ideia que a fase de desenvolvimento nunca termina) [6].

O levantamento de requisitos é outro ponto onde ambos os *softwares* diferem: é mais complexo elaborar a lista de requisitos para uma aplicação padronizada do que para uma aplicação à medida. Enquanto numa aplicação à medida o levantamento é realizado com o

cliente ou obtendo algum tipo de *feedback* do cliente, isto não acontece nas aplicações padronizadas, uma vez que não existe um cliente em concreto. Apenas após a fase de lançamento do produto e da disponibilização de canais de *feedback*, é que a equipa de programação terá acesso ao *feedback* dos utilizadores. Existe ainda a hipótese de as empresas realizarem estudos de mercados, avaliarem a concorrência e estarem atentos a novas leis [6].

Será difícil para o utilizador comparar o custo de produção de ambos, uma vez que apenas é possível comparar produtos padronizados concorrentes, visto que o produto à medida nunca será igual noutro cliente, criando vários pontos de diferença. O mesmo utilizador pode obter intervalos de preços distintos para o mesmo produto à medida, uma vez que não existe um valor base definido e que possa ser usado pelas várias empresas [7].

No caso dos produtos padronizados é possível comparar preços e funcionalidades disponíveis, uma vez que estes estão disponíveis e o preço de mercado dos produtos concorrentes são conhecidos. Num estudo realizado em 2012, foi concluído, com base em 31 *softwares* produzidos em organizações japonesas, que quando os *function points* e o preço unitário destes são conhecidos, a estimativa de erro no preço será de cerca de 87%; no entanto, quando o esforço e o preço unitário do esforço é conhecido, a estimativa de erro será de 20% [7].

Na Figura 2.1, e de acordo com um relatório elaborado em 2010, pode ver-se que se estima que 75% dos projetos de modernização ficam abaixo do orçamentado em 50% ou menos. Em contraste, 42% dos projetos com *software* padronizado ficam acima do orçamento em 50% ou mais (destes, 13% referem-se a projetos que ultrapassam em 100% o orçamento inicial) e 36% dos projetos relacionados com o desenvolvimento de novas aplicações veem os seus custos acima do orçamento em 50% ou mais (com 26% acima dos 100%) [13].

COST OVERRUN COMPARISON

RESOLUTION	CASE 1: APPLICATION DEVELOPMENT	CASE 2: PACKAGE APPLICATION	CASE 3: MODERNIZATION
Below 20%	43%	22%	46%
20% to 50%	21%	36%	29%
51% to 100%	10%	29%	14%
Over 100%	26%	13%	11%

Figura 2.1 – Comparação do custo de três tipos de *software* [13]

A Figura 2.2 tem representada a estimativa de funcionalidades utilizadas para uma aplicação à medida. De acordo com o mesmo relatório de 2010, é estimado que 20% das funcionalidades são utilizadas com frequência, sendo a maior percentagem respetiva às funcionalidades pouco ou nunca utilizadas. No que se refere às aplicações padrão, a percentagem de funcionalidades utilizadas com frequência desce para os 5%, sendo reverenciada uma estimativa de mais de 10 000 decisões para a implementação de um ERP numa média empresa [13].

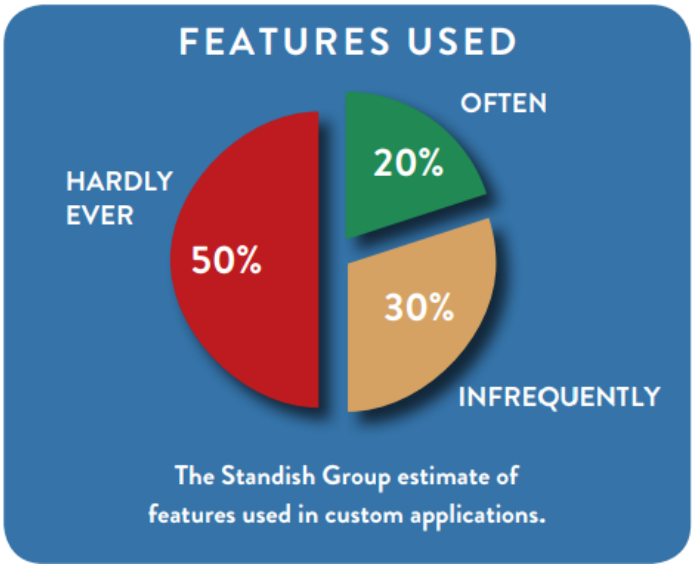

Figura 2.2 – Estimativa das funcionalidades utilizadas [13]

Podem ainda ser analisadas outras diferenças entre a implementação de um *software* à medida e um *software* padronizado. Enquanto no primeiro é possível desenvolvê-lo para que se encaixe exatamente como a empresa pretende, um *software* padronizado pode ser personalizado, mas apenas até um certo ponto, uma vez que é um sistema pronto a usar. Outra diferença é o tempo que decorre até ser colocado à disposição do utilizador: o *software* padronizado está pronto a usar assim que é adquirido, sendo apenas necessário instalar e dar formação aos utilizadores; já o *software* à medida pode levar bastante tempo a ser desenvolvido na sua totalidade, além do tempo que será despendido a dar formação aos utilizadores [14].

Na Tabela 2.1 podem ser vistas algumas das características do *software* à medida (subdividido em contractual e *in-house*) e *software* padronizado (este subdividido em *business-to-business* e *business-to-consumer*) [15].

Tabela 2.1 – Características do *software* à medida e do *software* padronizado [traduzido] [15]

Software	***Características típicas***
Software à medida	
Software à medida contractual	*Software* desenvolvido para um comprador específico
	Orçamento e calendário fixado
	Penalização por entregas fora do prazo
Software à medida interno	Utilizado para melhorar a eficiência/eficácia da organização interna
	Número limitado de utilizadores finais
	Possível conflito de interesses entre o departamento de TI e os utilizadores finais
Software padronizado	
Software padronizado business-to-business	*Software* vendido a outras empresas
	Muitos compradores diferentes
	Crítico para a empresa do comprador
Software padronizado business-to-consumer	*Software* vendido a compradores individuais
	Volume elevado de compradores
	Janelas de mercado e temporadas de compra
	Falhas podem ter consequências fatais

No entanto, o *software* padronizado tem sofrido uma evolução nos últimos anos para a venda como *Software as a Servise* (SaaS) – *software* como serviço – através da criação de plataformas em *cloud*, que permite às empresas adquirir produtos sem a preocupação de ter de manter uma infraestrutura. Ou seja, o utilizador aluga o *software* que necessita, sem ter de comprar o *software* ou a infraestrutura necessária para a sua instalação [16]. Este tipo de *software* apresenta vários benefícios [16]:

- Custos reduzidos, uma vez que apenas é pago o serviço (que a maioria das vezes já inclui o custo de manutenção);
- Fácil acesso, visto que os serviços são disponibilizados através do acesso via *browser*, basta o utilizador ter acesso à Internet;
- Escalabilidade, permitindo uma maior facilidade em adquirir funcionalidades.

No entanto, também apresenta algumas desvantagens como [17]:

- Personalização, a maior parte deste tipo de sistemas não permite uma customização do *software*, permitindo apenas pequenas personalizações;

- Segurança, uma vez que o fornecedor do sistema/*cloud* tem acesso a toda a informação da organização, pode levantar questões relacionadas com a segurança e privacidade da informação da empresa.

Um exemplo disto é a aquisição de um ERP em *cloud*. Este é um tipo de *software* padronizado adquirido pelas empresas, sendo personalizado de acordo com os processos delas. No entanto, as micro, pequenas e médias empresas têm interesse em adquirir serviços em *cloud*, uma vez que reduz os custos, mesmo não sendo tão personalizáveis [18].

3. Metodologias de desenvolvimento

Este capítulo apresenta a metodologia utilizada durante o desenvolvimento de ambos os projetos. Esta é a metodologia utilizada pela empresa, não tendo sido necessário realizar uma análise para perceber qual a melhor metodologia para cada projeto.

A metodologia escolhida pela empresa é uma metodologia ágil, com a aplicação de alguns conceitos de *Scrum*. As metodologias ágeis permitem uma maior flexibilidade durante o desenvolvimento dos projetos.

A metodologia ágil consiste em dividir um projeto em pequenos subprojectos, sendo estes desenvolvidos em curtos espaços de tempo e sujeitos a alterações [12]. Em 2001, foi criado o manifesto que define os valores e os princípios desta metodologia, havendo uma valorização [19], [20]:

- Dos indivíduos (seja da equipa ou com os clientes) e das suas interações;
- Do funcionamento do *software*;
- Da colaboração com o cliente;
- E da capacidade de resposta à mudança.

A lista dos princípios vai também de encontro aos valores definidos pela equipa que criou o manifesto [20].

O princípio da metodologia ágil é considerar que o desenvolvimento de *software* deve ser iterativo e incremental, permitindo que o cliente possa adicionar novas funcionalidades ou pedir alterações nas existentes. O facto de o projeto ser dividido em subprojectos facilita o tempo de entrega, o planeamento dos desenvolvimentos e uma resposta de acordo com as necessidades do cliente [19].

O *Scrum* consiste numa *framework* da metodologia ágil, que permite controlar e gerir os processos do *software* a desenvolver. Esta *framework* foi pensada para aumentar a velocidade do desenvolvimento das aplicações, manter um alinhamento entre os membros do projeto, melhorar a comunicação em todos os níveis, entre outros objetivos. Uma das vantagens do *Scrum* é a flexibilidade deste, uma vez que os requisitos a serem desenvolvimentos são escolhidos em cada ciclo, sem necessidade de seguir procedimentos específicos [21].

Na Figura 3.1 é possível visualizar a sequência dos artefactos e eventos que constituem o *Scrum*. De forma simples, a cada iteração dá-se o nome de *Sprint*, que tem uma duração definida antes de iniciar o processo, podendo durar um mês ou menos, mas todas as iterações têm de ter a mesma duração para manter a consistência [22]. No início de cada *Sprint*, acontece o *Sprint Planning*, que é o evento onde é planeado todo o *Sprint*. Neste evento, todos os membros da equipa estão presentes e contribuem para decidir quais os itens do *Product Backlog* (lista de requisitos a desenvolver, ordenados por importância) que irão constituir o *Sprint Backlog* (plano composto pelo objetivo do *Sprint*, os itens do *Product Backlog* a desenvolver e o plano de entrega) [22].

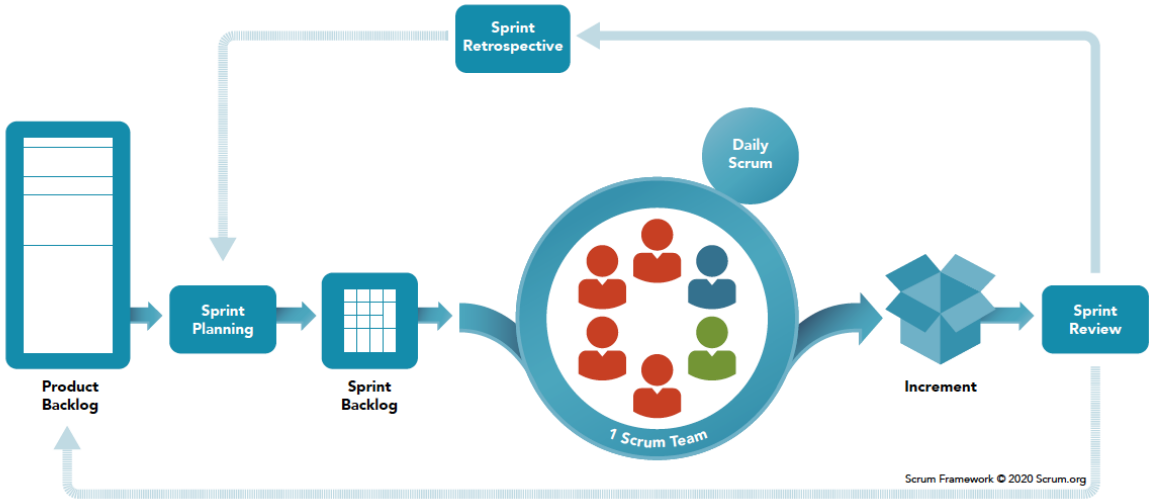

Figura 3.1 – Ciclo das *sprints* do Scrum [22]

Após este evento inicial, ocorre o período de desenvolvimento durante o qual são realizados os *Daily Scrum*. Estes eventos consistem em reuniões diárias, de 15 minutos, realizadas pela equipa de desenvolvimento, que servem para verificar o progresso feito e ajustar o *Sprint Backlog* se necessário [22].

No final do *Sprint* é feita a incrementação do projeto com os itens desenvolvidos até ao momento e acontece o *Sprint Review*. Este é um evento onde todos os elementos envolvidos no projeto estão presentes e onde é avaliado o que foi concluído e as alterações que possam ter existindo. O último evento que acontece é o *Sprint Retrospective*; este serve para planear formas de aumentar a qualidade e a eficácia da equipa. Este evento serve ainda para avaliar como correu o *Sprint* e o que pode ser melhorado. Após este evento, o *Sprint* acaba e começa todo um novo ciclo [22].

Além destes eventos, existem ainda três papéis (*roles*) presentes no *Scrum* [22]:

- *Scrum Master*, esta é a pessoa responsável por manter a harmonia no projeto. Funciona como uma espécie de líder do projeto, servindo como um guia para manter a eficácia e remover impedimentos;
- *Product Owner*, é a pessoa que garante que o produto final tem valor. Funciona como o defensor do cliente, sendo o seu objetivo garantir que os pontos do *Product Backlog* são desenvolvidos de acordo com o pretendido pelo cliente;
- *Developers*, é a equipa responsável por desenvolver o projeto. Deve ser uma equipa multidisciplinar, ou seja, não deve apenas existir elementos para programar, mas também para analisar, testar, projetar, entre outros.

Como dito no início do capítulo, a empresa utiliza uma metodologia ágil, com a aplicação de alguns conceitos *Scrum*. É uma metodologia ágil, porque existe uma primeira fase onde é elaborada uma lista de requisitos, que não é fixa e que vai sendo alterada no decorrer dos projetos. São ainda disponibilizadas conjuntos de funcionalidades para o cliente testar, permitindo obter *feedback* durante o desenvolvimento e ir de encontro ao que o cliente realmente pretende.

Do *Scrum* são utilizados alguns conceitos, como as reuniões internas, realizadas com alguma regularidade, para garantir que os elementos do projeto conhecem os objetivos e estão alinhados entre eles, sendo que estas reuniões servem também para rever a lista dos requisitos São também realizadas reuniões com o cliente, quer para apresentação das funcionalidades já desenvolvidas, quer para realizar um ponto de situação do projeto (o que tem de ser alterado, o que vai ser feito e o que fica pendente).

No entanto, não se pode considerar o uso pleno da *framework Scrum*, uma vez que as reuniões não acontecem em ciclos bem definidos. A definição de requisitos (*Sprint Planning*) acontece em simultâneo com o ponto de situação (*Sprint Review*). Além das "falhas" na implementação dos eventos e artefactos, existe também uma falha na atribuição dos papeis: não existe um *Scrum Master*, nem um *Product Owner*, de acordo com as definições apresentadas, sendo apenas possível identificar os elementos que fazem parte da equipa de desenvolvimento. No entanto, há um elemento da equipa que desempenha algumas das funções de *Scrum* Master e de *Product Owner*, garantindo o bom rumo dos projetos e que os requisitos do cliente são cumpridos.

No projeto *PHC Web* não é possível o uso de versões, devido à ferramenta disponibilizada pela *PHC*, mas no projeto *aWMS* foi utilizado uma ferramenta para controlo de versões, que será abordada na secção 5.1. Já o registo de tarefas a desenvolver é realizada através da ferramenta interna que existe na empresa, como mostra a Figura 3.2. Uma tarefa a verde significa que foi planeada e realizada no tempo previsto, enquanto que uma tarefa a vermelho significa que foi planeada, mas que não foi realizada no tempo previsto inicialmente. Por vezes, o atraso no planeamento deve-se a situações mais urgentes que aparecem ou a processos que se atrasaram no desenvolvimento.

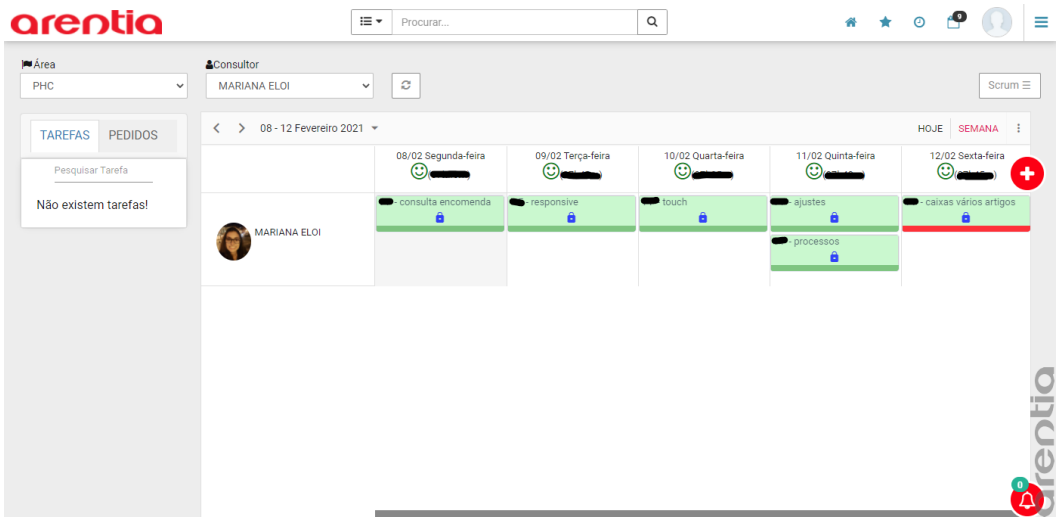

Figura 3.2 – Ferramenta interna para planeamento de tarefas

4. Desenvolvimento – Projeto *PHC Web*

Neste capítulo é apresentado o processo de desenvolvimento do projeto *PHC Web*. Uma vez que este foi um projeto desenvolvido com base num *software* já existente, é feita uma breve apresentação do *software* prévio (secção 4.1), quais as necessidades do cliente (secção 4.2) e o que foi desenvolvido nesse âmbito (secção 4.3).

4.1. Apresentação da solução

4.1.1. *Background*

Este projeto foi desenvolvido sobre o ERP *PHC*, existente no mercado português desde 1989, da empresa com o mesmo nome. A *PHC* está também presente em mais quatro países, sendo o seu foco a distribuição de sistemas de apoio à gestão empresarial [23]. A distribuição é realizada através da sua rede de parceiros, que são empresas certificadas para "comercializar, implementar e dar apoio pós-venda em soluções *PHC*" [23].

O principal produto da marca é o *PHC CS* (doravante referido como *CS*), sendo este um *software* modular e que requer a instalação em cada posto de trabalho. O facto deste *software* ser modular, permite que o cliente apenas adquira os módulos que lhe são necessários, podendo acrescentar módulos à medida que a empresa vai necessitando. Este *software* existe em três gamas, para que se possa adaptar a cada tipo de empresa [24]:

- *PHC CS Corporate*, desenvolvido para microempresas, limita o número de utilizadores em simultâneo, assim como as hipóteses de desenvolvimentos por parte do parceiro e o tamanho da base de dados;
- *PHC CS Advanced*, pensado para pequenas e médias empresas, não tem limite de utilizadores em simultâneo, requere a compra de uma licença de base de dados, mas permite mais desenvolvimentos à medida do cliente;
- *PHC CS Enterprise*, é topo de gama, para empresas de *mid market*, partilhando das mesmas características da gama anterior, mas com uma *framework* que permite ir mais longe.

Para além da versão *desktop*, existem vários complementos, como é o caso do *PHC Web*. O *PHC Web* é uma versão *web*, que permite realizar alguns dos processos disponíveis no *desktop* em vários tipos de dispositivos, sendo apenas necessário uma ligação à Internet para

aceder, dando mobilidade ao sistema, como demonstrado na Figura 4.1. Esta versão serve de complemento à versão *desktop*, uma vez que toda a parametrização é realizada através do *CS*. No entanto, também a versão *web* é constituída por módulos, possibilitando a escolha dos módulos que se pretendem ter disponíveis através do acesso *web* [24].

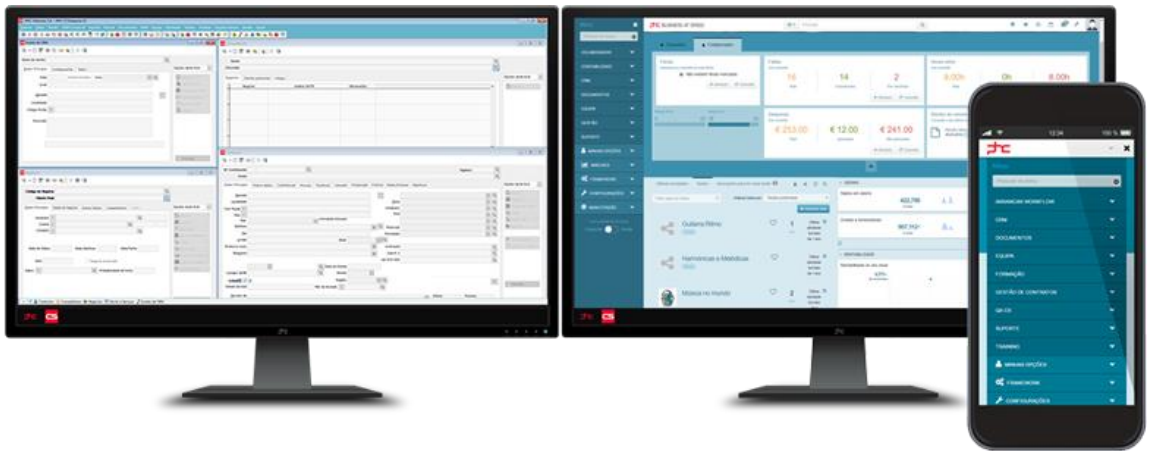

Figura 4.1 – Exemplos da versão *desktop* e *web* [25]

Além dos módulos que partilha com a versão *desktop* para gestão interna, o *PHC Web* disponibiliza ainda módulos que podem ser acedidos por pessoas externas à empresa (*extranet*), assim como a criação de uma loja *online*. O módulo *Extranet* permite a interação da empresa com os seus clientes, como por exemplo, a colocação de pedidos de suporte através de um portal, não sendo necessário o cliente enviar um email, uma vez que o pedido fica logo disponível no sistema [25].

Este projeto foca-se na implementação dos requisitos do cliente, de forma a adaptar a solução *PHC Web* adquirida às necessidades reportadas pelo cliente. Os pormenores abordados em *CS* serão referentes a parametrizações que não são possíveis de realizar através da solução *web*.

4.1.2. Arquitetura da solução

Mesmo sendo uma solução adquirida e modificada para ir de encontro aos requisitos do cliente, esta apresenta uma arquitetura assente em três camadas: serviço de dados, regras de negócio e apresentação. Este tipo de arquitetura consiste na divisão de um projeto em camadas, que utilizam funcionalidades disponibilizadas pelas camadas inferiores. Este tipo de divisão faz com que a camada de dados não tenha qualquer tipo de conhecimento sobre como será a camada de apresentação [26].

Nos sistemas complexos (como é o caso dos ERP), este tipo de arquitetura possibilita que se possa desenvolver novas funcionalidades, sem perceber como funciona toda a estrutura da aplicação. Esta vantagem permite que um dado elemento se foque apenas numa camada, precisando apenas de conhecer qual o ponto de contacto entre camadas [26].

No caso desta solução, o estudo da arquitetura a aplicar não foi realizado, visto ser uma aplicação previamente desenvolvida pela marca que a fornece. Assim, apenas se teve de seguir os requisitos indicados para realizar a instalação da aplicação. A utilização de dois servidores (um servidor com a aplicação *web* e um servidor com a aplicação *desktop* e a base de dados) serve para distribuir recursos e não sobrecarregar um único servidor.

Sendo esta uma aplicação disponibilizada por uma empresa, existem determinados requisitos a serem cumpridos para realizar a sua instalação, sendo eles a instalação do *Internet Information Services* (IIS) e a *Framework .Net*. A existência da base de dados já está garantida, uma vez que é necessário ter uma instalação da versão *desktop* (momento em que é instalada a base de dados). A Figura 4.2 apresenta a estrutura da aplicação *PHC Web*.

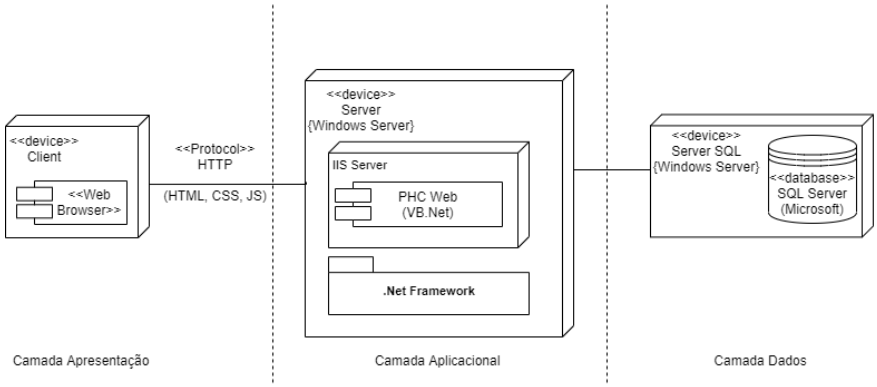

Figura 4.2 – Diagrama de *deployment*

Sendo uma aplicação *web*, esta é uma estrutura distribuída cliente-servidor. Neste caso em específico, existe um servidor onde está alojada a base de dados, que comunica com o servidor onde está alojada a aplicação e este é acessível através de um cliente (*browser*). A *PHC* não tem como requisito a existência de um servidor apenas para a base dados, ou seja, as aplicações e a base de dados podem estar todas alocadas no mesmo servidor.

A camada aplicacional refere-se ao servidor onde foi instalada a aplicação, que tem como sistema operativo uma versão do *Windows Server*. Visto que são necessários recursos *Microsoft*, a utilização do sistema operativo *Windows* garante compatibilidade entre recursos. O IIS é um servidor *web* desenvolvido pela *Microsoft* para sistemas *Windows*,

utilizado para alojamento de *sites* e outros conteúdos na *web* [27]. A *Framework .Net* foi também desenvolvida pela *Microsoft* e é uma plataforma de desenvolvimento para vários tipos de aplicações, incluindo *web* [28]. A base de dados é gerida através do *Microsoft SQL Server*, que é o sistema de gestão de base de dados relacionais da *Microsoft* [29].

A camada de dados, neste projeto, encontra-se num servidor diferente do servidor da aplicação, não sendo este servidor extra uma obrigatoriedade, podendo a camada de dados estar no mesmo servidor que a aplicação (como dito anteriormente). A base de dados em branco é disponibilizada pela *PHC* e instalada pelo técnico que faz a instalação do *CS desktop*. Esta é uma base de dados relacional, ou seja, é uma base de dados que armazena informação que está relacionada uma com a outra. Isto é, numa base de dados relacional cada tabela tem uma coluna que vai servir de identificação para outras tabelas, criando uma relação entre as várias tabelas [30]. A Figura 4.3 apresenta o modelo relacional parcial da base de dados utilizada. Não serão apresentados os atributos completos uma vez que foi desenvolvida pela marca.

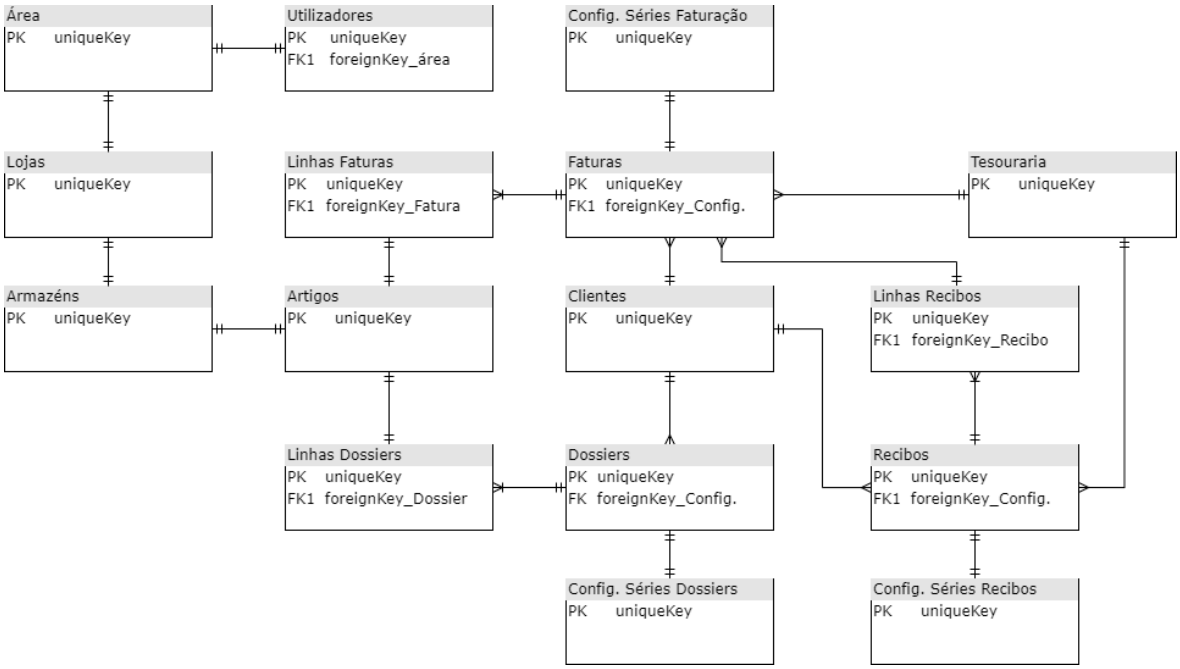

Figura 4.3 – Modelo relacional parcial da base de dados utilizada

Optou-se por representar apenas as tabelas mais utilizadas, já que existem tabelas que apenas são utilizadas em módulos que o cliente não adquiriu. Posto isto, as tabelas mais relevantes são as tabelas Clientes, Artigos, Faturas, Dossiers e Recibos. Estas são as tabelas mais utilizadas durante a utilização da aplicação. As tabelas Lojas, Armazéns, Configuração de Séries de Faturas, Configuração de Séries de Dossiers e Configuração de Séries de Recibos

são tabelas de parametrização, que, após configuradas, servem apenas de consulta para a criação de documentos.

4.1.3. Tecnologias utilizadas

Este projeto foi desenvolvido numa plataforma já existente e estereotipada, pelo que as tecnologias/linguagens utilizadas teriam de ser as indicadas pela marca do programa. Assim sendo, a *PHC* disponibiliza uma *framework*, onde é possível o desenvolvimento de eventos em *VB.Net*, assim como a utilização de HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*), *Bootstrap* e *JavaScript*.

Esta *framework* permite que os técnicos/programadores possam criar eventos, regras, valores por defeito, *scripts* ou opções de ecrã para irem de encontro ao que o cliente necessita. Todas estas opções são desenvolvidas em *VB.Net*.

No caso dos eventos, eles "correm" de acordo com o tipo de evento criado e no ecrã indicado, ou seja, um evento desenvolvido para ocorrer cada vez que é alterado um cliente, apenas irá "correr" quando se grava uma ficha de cliente que está a ser editada. As regras ocorrem apenas ao gravar e servem para validação de dados ao introduzir ou alterar. Os valores por defeito podem ser despoletados ao introduzir, ao editar ou apenas ao gravar, sendo que estes servem para preencher campos com determinados valores (fixos ou dinâmicos). No caso dos *scripts*, estes são úteis para servir como API (*Application Programming Interface*), uma vez que só "correm" ao serem chamados. Por último, as opções de ecrã permitem criar opções personalizadas, além das opções disponibilizadas por defeito.

Para além destes, são ainda disponibilizadas formas de criar ecrãs novos, adicionar novos campos, adicionar filtros para os dados e criar "*JavaScript* de Utilizador".

A criação de ecrãs através da opção disponibilizada permite criar um ecrã novo em segundos e sem a necessidade de desenvolver as funções CRUD (*Create*, *Read*, *Update* e *Delete*), sendo apenas necessário criar os campos (também através da opção presente na *framework*). A opção "*JavaScript* de Utilizador" permite criar funções em *JavaScript* que podem ficar disponíveis de forma global ou apenas para um ecrã; é aqui que se podem fazer as chamadas aos *scripts* desenvolvidos.

Esta *framework* possibilita o acesso a funções desenvolvidas pela *PHC*. A maioria delas, além de facilitar a criação de código, permite garantir que determinados processos são

executados de forma correta, como por exemplo, o cálculo dos totais de uma fatura. A presença de funções deste tipo permite que os técnicos não se tenham de preocupar em realizar os cálculos corretamente ou garantir questões legais, uma vez que as funções garantem isso. Para o uso da *framework*, era frequente consultar a comunidade da *PHC* para procurar informações sobre as funções disponibilizadas ou existentes, assim como possíveis limitações das mesmas.

4.2. Modelo de negócio

O projeto *PHC Web* foi desenvolvido para colmatar necessidades que o cliente X[2] tinha e que a plataforma precisava de apresentar. De forma geral, este cliente dedica-se à comercialização de produtos para reparação automóvel e possui três lojas (uma das quais num dos arquipélagos português), assim como um armazém independente das lojas. O cliente necessitava de um sistema que lhe permitisse centralizar toda informação num único ponto, com a necessidade de os postos serem móveis.

Assim, foi-lhe proposto a aquisição de um posto CS *desktop* e do *PHC Web*, ambos com os módulos de Gestão e Documentos Eletrónicos. O módulo de Gestão é responsável por permitir realizar a faturação nas lojas, assim como permite criar clientes, artigos, realizar gestão de *stocks* e a criação de documentos internos. O módulo "Documentos Eletrónicos" foi adquirido para que fosse possível a comunicação de guias de transporte à Autoridade Tributária através do *webservice* disponibilizado pela mesma.

Visto que o foco do cliente é a comercialização de produtos em várias lojas, a grande parte dos requisitos estão relacionados com a facilitação de processos, aumentando a rapidez na criação de documentos e com a tentativa de diminuir a probabilidade do erro humano provocado por distração e/ou desconhecimento. Além da criação direta de alguns documentos, como é o caso das faturas, existem alguns processos que requerem a utilização de vários documentos, tendo a informação de ser copiada entre eles:

- Encomendas de cliente: tem de ser criado um documento do tipo "Encomenda de cliente", que depois servirá como base para faturar os artigos ao consumidor;

[2] Devido ao RGPD, não será apresentado o nome do cliente, assim como se irá ocultar qualquer tipo de informação que possa servir para identificar o cliente em causa.

- Amostras: é criado um documento do tipo "Amostra", que pode servir de base para: faturar artigos ao consumidor ou criar um documento do tipo "Devolução" para dar entrada dos artigos em *stock*;
- Transferências entre lojas: é criado um documento do tipo "Envio para loja", sendo este utilizado para que a loja que recebe os artigos crie o documento "Receção em loja";
- Guia de transporte: existe uma fatura com artigos que serão movimentados entre o armazém e a morada de descarga, sendo necessário realizar um documento do tipo "Guia de transporte", que seja copiado da fatura, como forma de justificar a movimentação dos artigos.

De forma a ser mais simples identificar os requisitos necessários, estes foram convertidos em processos, podendo estes ser traduzidos em *User Stories* (US). As US (que podem ser consultadas no Anexo A, na Tabela A.1) foram divididas em cinco grupos: Gestão de clientes, artigos e logística, Abertura e fecho de caixa, Vendas, Gestão interna e Análises. Cada grupo apresenta as funcionalidades/regras necessárias em cada área. A área que requereu mais atenção foram as vendas, uma vez que é a área mais importante da aplicação. Cada US está descrita na perspetiva da loja, sendo esta a representação do utilizador.

4.3. Implementação[3]

A implementação deste projeto requereu um período de adaptação e de aprendizagem quer da *framework* da *PHC* como de *VB.Net*, uma vez que era uma linguagem sobre a qual não existiam conhecimentos prévios e havia diferenças de sintaxe em relação a outras linguagens utilizadas previamente. Este processo foi relativamente rápido, contando com o apoio dos colegas para o esclarecimento de dúvidas sobre a *framework*, assim como no desenvolvimento de algumas das funcionalidades.

A maioria do desenvolvimento neste projeto consistiu na criação de eventos. Os eventos consistem em pedaços de código que correm em determinado estágio do ecrã e que podem ocorrer sempre ou de acordo com as condições definidas pelo técnico. Os eventos mais utilizados foram:

- "AposOnInit", ocorre quando os objetos estão a ser colocados no ecrã;

[3] Durante este subcapítulo, todas as imagens apresentadas que sejam referentes à versão do cliente foram sujeitas a edição para ocultar informação confidencial/sensível.

- "AposPreRender", ocorre imediatamente antes de apresentar os dados;
- "AposIntroduzir", ocorre quando é gravado um ecrã que está em modo de introdução (não ocorre caso o ecrã esteja em modo de alteração).

Os eventos "AposOnInit" criados servem para ocultar/visualizar campos dos ecrãs, assim como para manipular as grelhas (ocultar/visualizar colunas, tornar apenas de leitura, entre outros). Os eventos "AposPreRender" servem também para ocultar/visualizar campos dos ecrãs, mas também para adicionar objetos ou alterar informação de objetos já existentes. Os eventos "AposIntroduzir" servem para preencher campos com a informação que é necessária estar previamente preenchida (antes do ecrã estar totalmente preenchido).

Houve também a necessidade de utilizar a funcionalidade para criar "*JavaScript* de Utilizador", para colmatar eventuais necessidades ou minimizar espaços em branco, deixados pela ocultação de campos.

Existem US descritas que envolveram apenas a configuração no *PHC CS*, não sendo necessário qualquer tipo de desenvolvimento, como acontece nas US relativas a permissões e/ou configurações (como, por exemplo, a US1.8, "Enquanto loja, não posso introduzir novos clientes, nem alterar ou eliminar existentes"). Em todos os ecrãs, em que exista o campo "Área", foi criado um valor por defeito para o preencher automaticamente com a área definida na ficha do utilizador (este é o campo que identifica a loja que está com a sessão iniciada) e existem eventos para o colocar apenas de leitura, satisfazendo as US relacionadas com a área da loja (US2.1, US2.3, US3.3, etc.).

4.3.1. Página inicial

A Figura 4.4 apresenta a página inicial da versão de demonstração do *PHC Web*. Esta versão tem as configurações que vêm predefinidas, apresentando dados meramente ilustrativos.

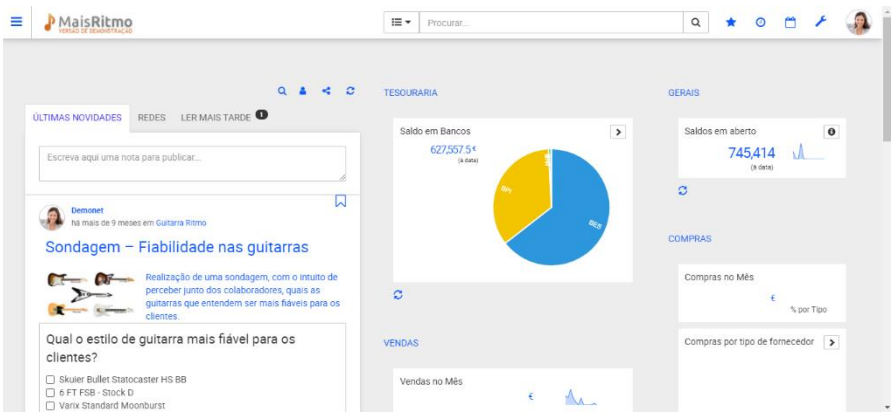

Figura 4.4 – Ecrã da página inicial da versão de demonstração *PHC Web*

A Figura 4.5 apresenta a página inicial que foi implementada no cliente. Para obter a configuração visível no cliente foram configuradas apenas duas colunas, retiradas todas as análises e opções que vinham por defeito, adicionados os botões para funcionarem como atalhos para os menus mais frequentes e adicionados os *snapshots* (análises sem filtros variáveis) para facilitar alguns processos (US5.10). Os botões disponíveis nos *snapshots* permitem criar documentos em poucos cliques, uma vez que o ecrã do documento escolhido (seja fatura ou dossier) irá ser apresentado com a informação preenchida, de acordo com o documento de origem (US5.2, US5.4 e US5.6).

A pedido do cliente, foi adicionada a indicação da loja que tem a sessão iniciada na barra de navegação, à esquerda, e um ícone com a indicação das encomendas pendentes de clientes daquela loja (esta informação é atualizada a cada cinco minutos).

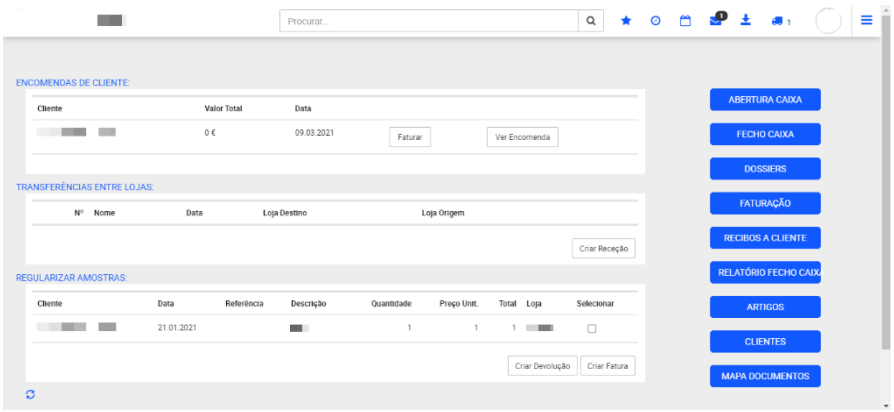

Figura 4.5 – Ecrã da página inicial disponível no cliente

4.3.2. Gestão de Clientes, Artigos e Logística

Os ecrãs que sofreram menos alterações foram os ecrãs dos clientes e dos artigos. Nestes ecrãs apenas foram ocultados campos que não seriam necessários ou aplicados filtros para apenas serem apresentados os registos da loja com sessão iniciada (US1.4 e US1.10). A Figura 4.6 e a Figura 4.7 apresentam uma parte do ecrã dos clientes e dos artigos.

Além disso, no ecrã de Clientes, foram desenvolvidas opções de ecrã para imprimir a conta-corrente completa do cliente (US1.6) e para imprimir a conta-corrente dos movimentos por regularizar (US1.7). Já no ecrã dos artigos foram disponibilizadas opções de ecrã para consultar os dois ficheiros definidos para o artigo (US1.13) e para imprimir etiquetas (US1.14). Foram ainda criadas algumas análises de ecrã, que apenas apresentam os dados relativos ao registo que está a ser consultado.

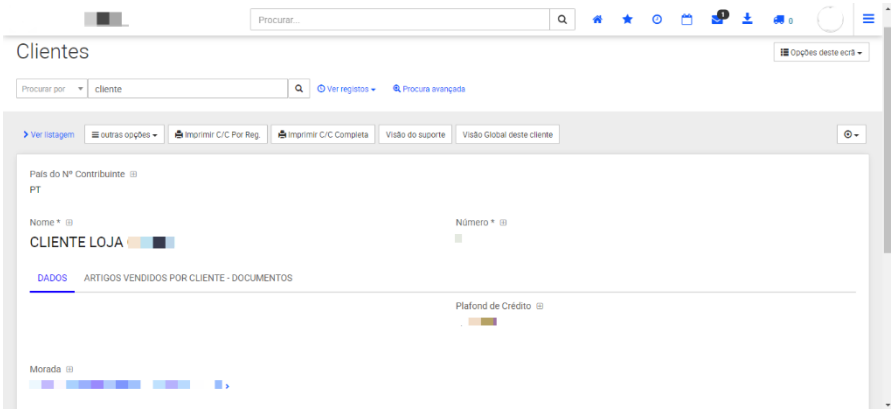

Figura 4.6 – Ecrã dos clientes, em modo de consulta, disponível no cliente

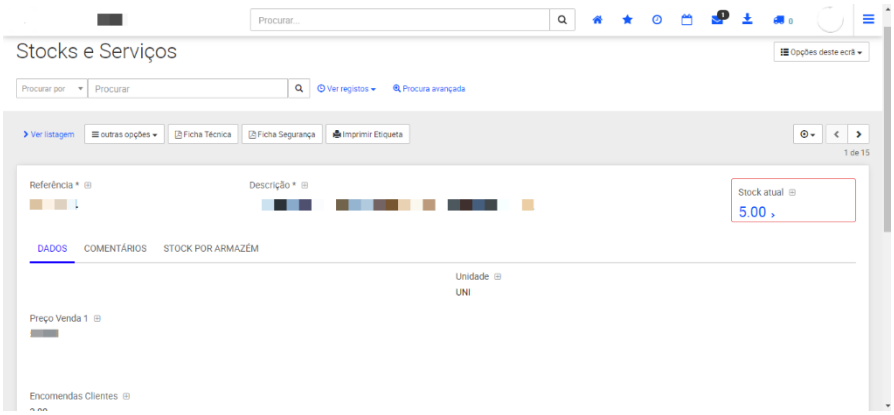

Figura 4.7 – Ecrã dos artigos, em modo de consulta, disponível no cliente

4.3.3. Abertura e Fecho de caixa

A Figura 4.8 apresenta o ecrã de abertura de caixa que foi implementado no cliente. Este ecrã foi criado através da *framework*, facilitando o processo de disponibilização de campos. Apenas é necessário indicar a tabela da base de dados com a qual se vai trabalhar e adicionar os objetos necessários, indicando o campo da tabela e a posição no ecrã.

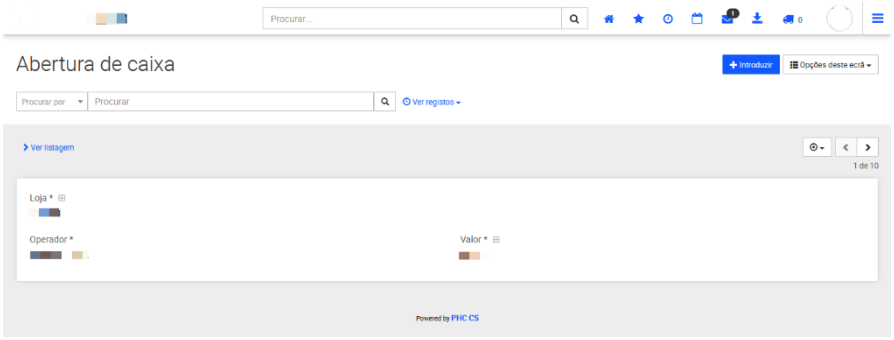

Figura 4.8 – Ecrã de abertura de caixa, em modo de consulta, disponível no cliente

A criação do ecrã de abertura de caixa e do ecrã de fecho de caixa foi realizado através desta opção, uma vez que os ecrãs são iguais, apenas com a diferença do campo onde é registado o valor. A introdução de um novo registo é sempre referente ao próprio dia, uma vez que não existe disponível o campo "Data" para edição (US2.2 e US2.5).

Foram também desenvolvidas regras para que não fosse possível abrir caixa sem existir um fecho de caixa para o último dia de trabalho, assim como o inverso (não é possível realizar um fecho de caixa se não existir uma abertura de caixa) (US2.4 e US2.7). A validação da abertura/fecho de caixa é também feita ao iniciar sessão para prevenir eventuais lapsos.

4.3.4. Faturação e recibos

A Figura 4.9 e a Figura 4.10 apresentam o ecrã de faturação, em modo de introdução, da versão de demonstração do *PHC Web*. A Figura 4.9 apresenta os primeiros campos disponíveis e a Figura 4.10 apresenta a grelha onde são inseridos os artigos, assim como os totais e dois separadores que têm os campos para os dados de transporte (data de carga, moradas de carga e descarga) e outras informações (referências internas, segmentos de mercado, entre outras).

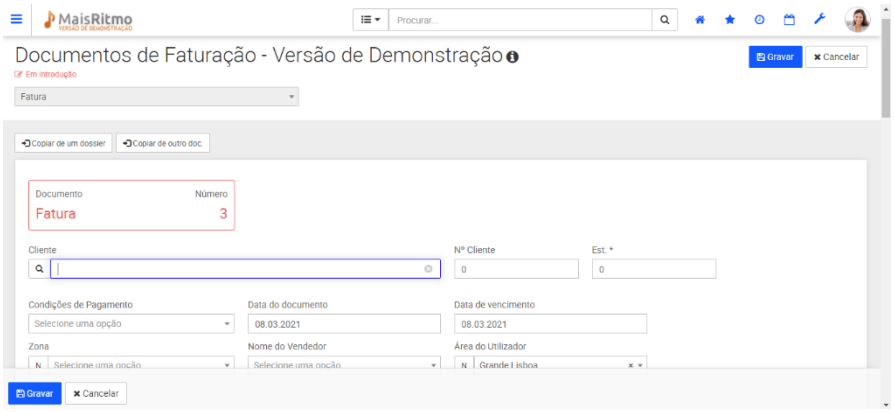

Figura 4.9 – Ecrã da faturação (cabeçalho), modo de introdução, da versão de demonstração *PHC Web*

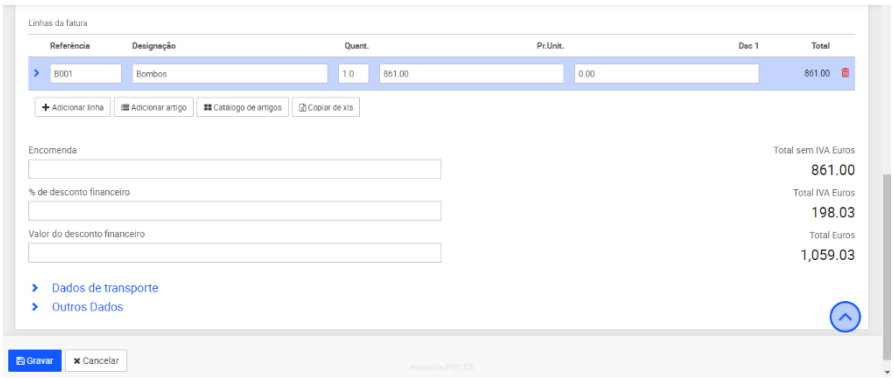

Figura 4.10 – Ecrã da faturação (linhas), modo de introdução, da versão de demonstração *PHC Web*

A Figura 4.11 e a Figura 4.12 apresentam o ecrã de faturação, em modo de introdução, da versão disponibilizada ao cliente. Na Figura 4.11 é possível ver que foram ocultados alguns campos, como "Nome do vendedor", "Zona" e "Área do utilizador", sendo substituídos por outros. Já na Figura 4.12 é possível ver que alguns campos referentes ao transporte foram

posicionados antes da grelha dos artigos (“Data de carga”, “Matricula”, “Hora de carga” e “Data efetiva de entrega”).

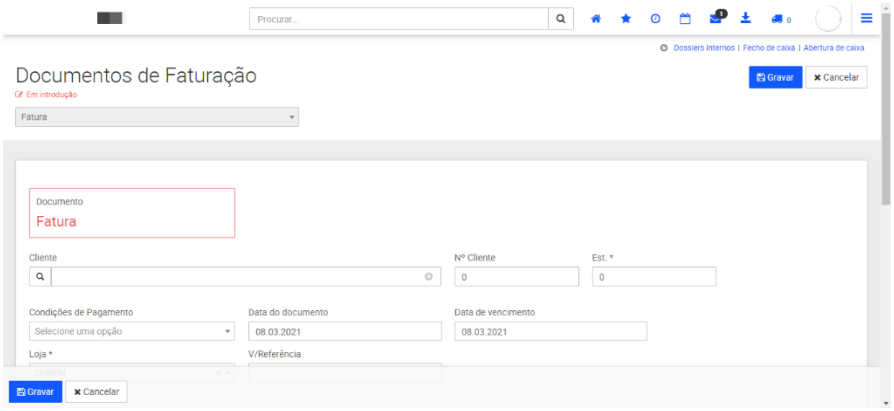

Figura 4.11 – Ecrã da faturação (cabeçalho), modo de introdução, disponível no cliente

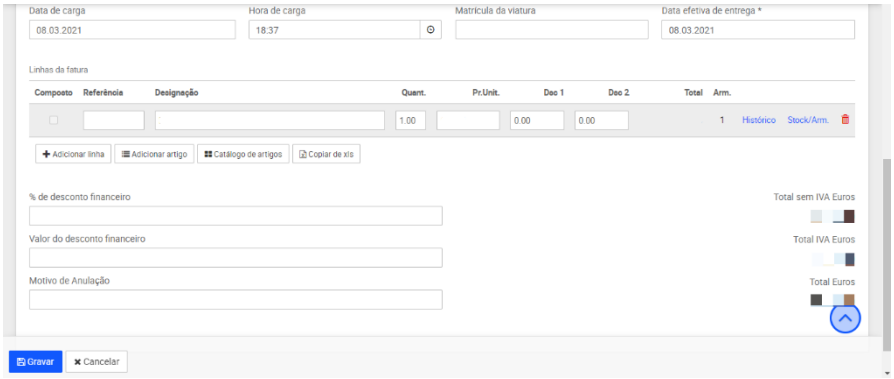

Figura 4.12 – Ecrã da faturação (linhas), modo de introdução, disponível no cliente

Na Figura 4.12 é ainda possível ver que a grelha também sofreu alterações, quando comparada com a Figura 4.10. Foram colocadas visíveis as colunas “Composto”, “Arm.” e “Dsc 2”, que por defeito estão ocultas. Os *links* foram implementados para facilitar a consulta de *stock* para o artigo da linha (US3.11) e o histórico de compra do artigo para o cliente preenchido na fatura. Ambas as consultas são apresentadas numa janela *popup*, como se pode ver na Figura 4.13.

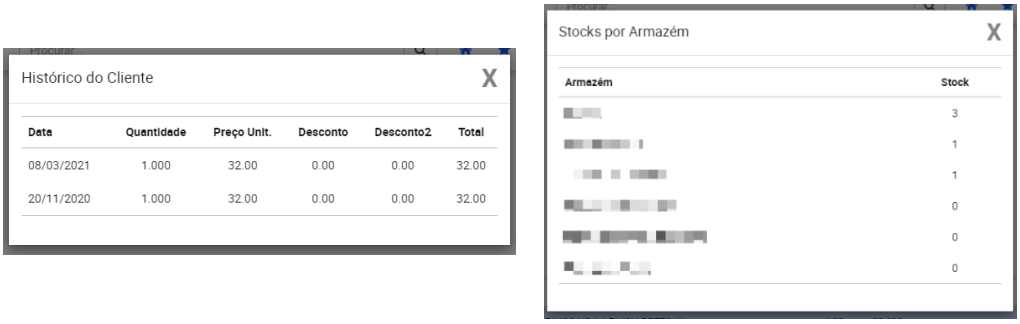

Figura 4.13 – Análises de consulta nas linhas das faturas, disponível no cliente

Além dos aspetos visíveis, esta opção de menu tem ainda associado um evento para, em caso de fatura/recibo, preencher o cliente com o cliente final da loja (US3.6) e apresentar o campo

"Método de pagamento" (US3.12). Existe um evento que apenas corre quando são adicionados artigos, que serve para preencher o armazém da loja na linha do artigo adicionado (US3.8), e ainda regras para validar se existe uma abertura de caixa (US3.4) ou se o cliente já ultrapassou o *plafond*.

Para o ecrã da faturação foram desenvolvidas três opções de ecrã, para a fatura em consulta: uma para a criação de uma nota de crédito partindo da fatura (US3.13); outra para a impressão do documento através do *browser*; e outra para o envio da fatura por email. A opção das notas de crédito copia os dados da fatura e acrescenta uma linha com o documento de origem. Ao consultar as notas de crédito, apenas aparecem disponíveis as opções de impressão e de envio por email.

A Figura 4.14 apresenta o ecrã dos recibos, em modo de consulta, da versão de demonstração do *PHC Web*, enquanto a Figura 4.15 apresenta o mesmo ecrã, mas a versão disponível no cliente. Este ecrã não sofreu muitas alterações, tendo sido apenas adicionado o campo "Método de pagamento" (US3.18) e ocultados os locais de tesouraria. As implementações mais relevantes foram a criação de uma regra para validar a abertura de caixa, como acontece nas faturas (US3.15), e um evento para processar o recibo após a sua gravação (US3.19).

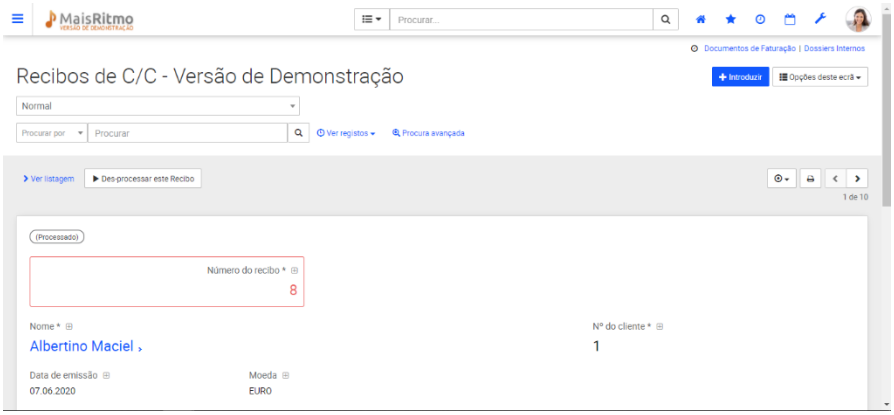

Figura 4.14 – Ecrã dos recibos, em modo de consulta, da versão de demonstração *PHC Web*

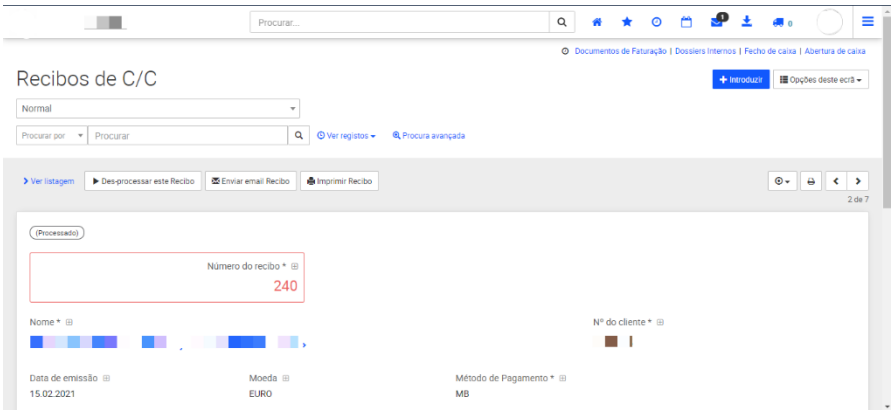

Figura 4.15 – Ecrã dos recibos, em modo de consulta, disponível no cliente

4.3.5. Gestão interna e análises

A Figura 4.16 e a Figura 4.17 apresentam o ecrã dos dossiers, em modo de introdução, da versão de demonstração do *PHC Web*. A Figura 4.16 apresenta os primeiros campos disponíveis e a Figura 4.17 apresenta a grelha onde são inseridos os artigos, assim como os totais e um separador com mais alguns campos.

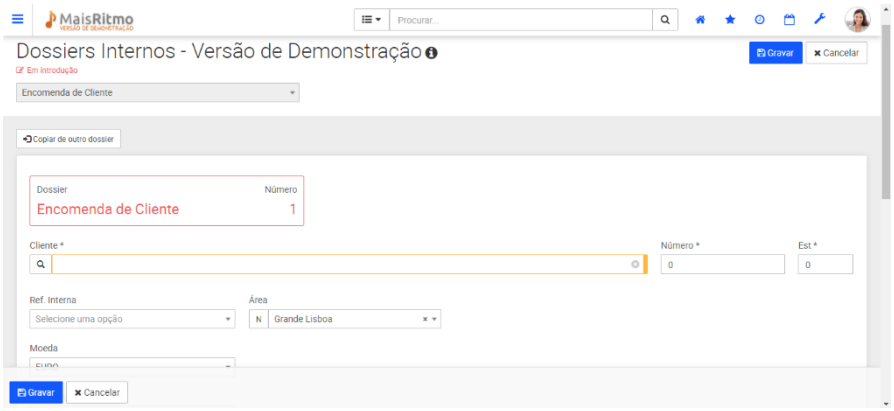

Figura 4.16 – Ecrã dos dossiers (cabeçalho), modo de introdução, da versão de demonstração *PHC Web*

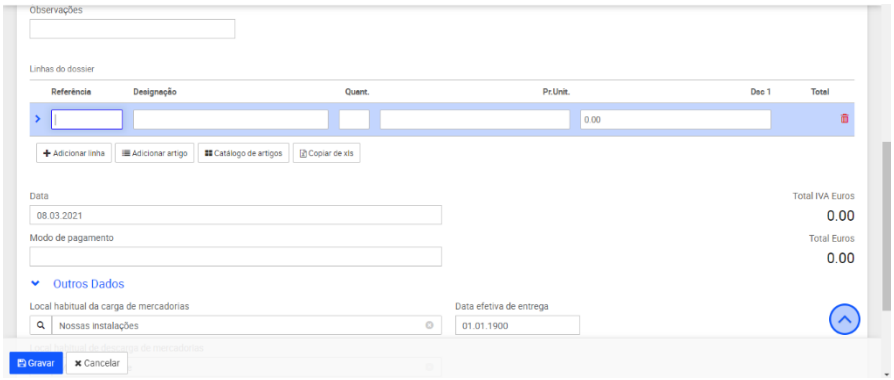

Figura 4.17 – Ecrã dos dossiers (linhas), modo de introdução, da versão de demonstração *PHC Web*

A Figura 4.18 e a Figura 4.19 apresentam o ecrã dos dossiers, em modo de introdução, da versão disponibilizada ao cliente. Na Figura 4.18 é possível ver que os campos não são os mesmos. Na Figura 4.19 é possível ver que o separador foi ocultado, assim como os seus campos. Ainda na Figura 4.19 é possível ver que a grelha também sofreu alterações, ficando com as colunas idênticas à faturação, dependendo do dossier selecionado.

Além dos aspetos visíveis, esta opção de menu tem associado um valor por defeito para preencher o nome do cliente em determinados dossiers (US4.4). Também neste ecrã existe um evento que apenas "corre" quando são adicionados artigos, para preencher o armazém da loja na linha do artigo adicionado. Não sendo opções visíveis no ecrã, ao copiar os dados de um dossier, seja para outro dossier ou para uma fatura, é acrescentada uma linha com o documento de origem (US4.5, US4.6 e US4.7).

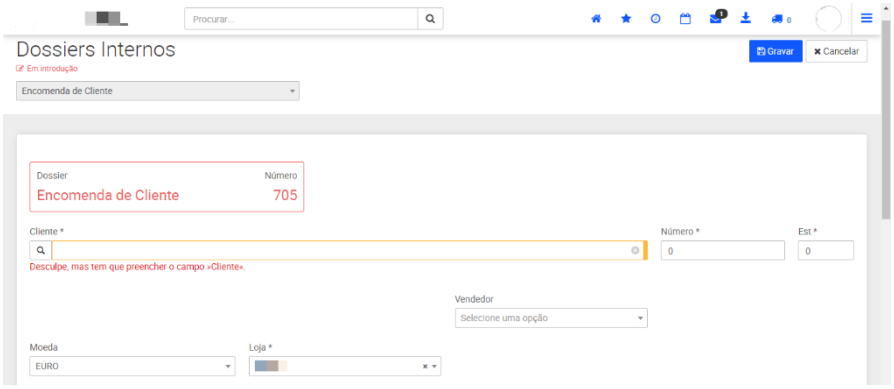

Figura 4.18 – Ecrã dos dossiers (cabeçalho), modo de introdução, disponível no cliente

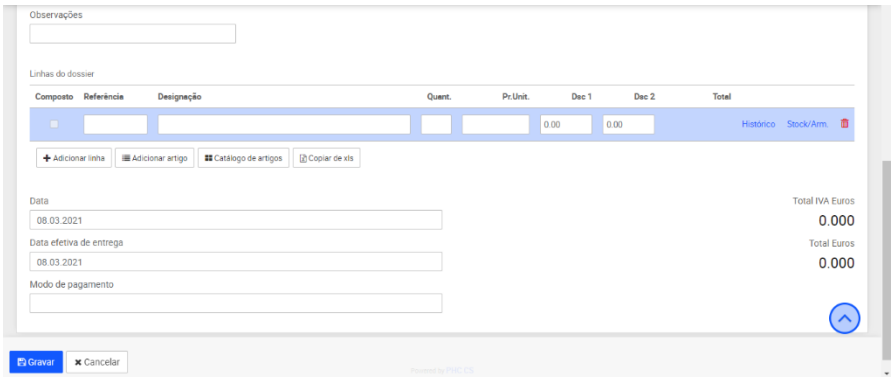

Figura 4.19 – Ecrã dos dossiers (linhas), modo de introdução, disponível no cliente

No que se refere às análises, estas são desenvolvidas de acordo com a informação que o cliente pretende visualizar, apresentando filtros variáveis para que o utilizador possa ajustar conforme necessite. Praticamente todas as análises apresentam um campo "Área", onde aparece selecionada a área do utilizador, e algum tipo de campo temporal (data de início, data de fim ou mês) (US5.3, US5.5, US5.7 e US5.9), como apresentado no exemplo da Figura 4.20.

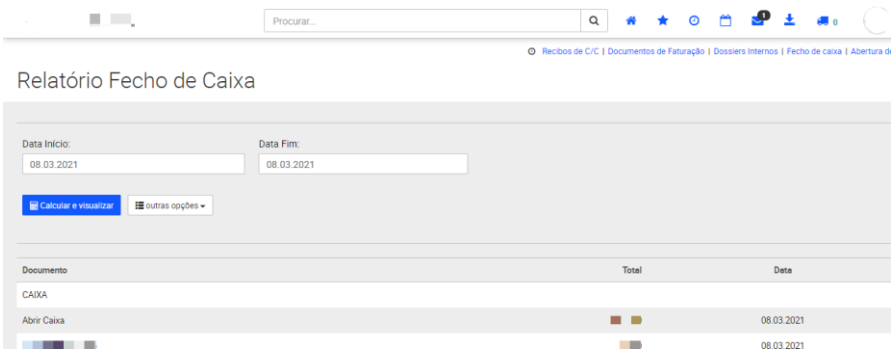

Figura 4.20 – Exemplo de uma análise, disponível no cliente

5. Desenvolvimento – Projeto aWMS

Neste capítulo é apresentado o processo de desenvolvimento do Projeto aWMS. Neste projeto já havia um desenvolvimento inicial, pelo que é feita uma breve apresentação do que já estava realizado (secção 5.1), quais as necessidades do cliente (secção 5.1.3) e o que foi desenvolvido nesse âmbito (secção 5.3).

5.1. Apresentação da solução

5.1.1. *Background*

Este projeto foi inicialmente desenvolvimento por outros dois colaboradores da empresa, que já não fazem parte da empresa. Assim, houve a necessidade de realizar algum enquadramento para perceber o que estava feito e qual o seu objetivo.

Esta aplicação foi proposta ao cliente em questão em substituição de um sistema que eles já possuíam: o *GESOBRA*. O *GESOBRA* é um *software* para a gestão de obras, com "o objetivo de dar um controlo total e personalizado sobre o decorrer da obra", desenvolvido pela empresa *CPS – Advanced Management, S.A* [31]. Este *software* é uma solução integrada no *ARTSOFT* (outro ERP, distribuído pela empresa *ARTSOFT*), que utiliza informação já existente de outros módulos (por exemplo, artigos) e apresenta uma curva de aprendizagem menor, uma vez que partilha um funcionamento idêntico ao *ARTSOFT*.

Através do *GESOBRA*, é possível controlar todo o processo de produção de obra, começando pela orçamentação, processo de produção e arrumação (disposição dos armazéns e locais disponíveis para receber obra produzida). Uma vez que o *GESOBRA* está integrado no *ARTSOFT*, o objetivo principal da aplicação proposta seria permitir ao cliente a gestão do armazém, através da informação disponibilizada pelo *PHC*.

Este projeto foca-se na implementação de uma aplicação (doravante identificada como aplicação *aWMS*), que vá de encontro aos requisitos pedidos pelo cliente e de acordo com algumas das especificações já apresentadas pelo sistema *PHC* que o cliente possui. Todo o projeto foi desenvolvido em parceria com os dois elementos da equipa que estavam responsáveis pelo projeto *PHC* do cliente, como forma de manter a consistência entre as regras das duas aplicações.

5.1.2. Arquitetura da solução

Sendo um dos modelos mais utilizados no desenvolvimento *web*, também este projeto apresenta uma arquitetura *Model-View-Controller* (MVC), ou seja, a aplicação está separada em Modelo (*Model*), Vista (*View*) e Controlador (*Controller*). Esta separação significa que a vista servirá apenas para apresentar a informação do modelo, sendo que o modelo apenas é manipulado pelo controlador. Assim, existe uma separação da apresentação do modelo, que torna o modelo independente da vista, ou seja, um modelo pode adaptar-se a qualquer vista, mas o inverso já não será possível [26].

A vantagem deste tipo de arquitetura é que o mesmo modelo pode ser adaptado a múltiplas vistas, como aconteceu neste projeto, em que o mesmo modelo é apresentado em múltiplas vistas, em apresentações diferentes e manipulados pelo controlador. Isto permitiu ir adaptando as vistas às necessidades do cliente, sem ter de alterar drasticamente a construção dos modelos [26].

Neste projeto também não houve a necessidade de realizar o estudo da arquitetura, uma vez que este já vinha estruturado e com a arquitetura definida. Apenas foi necessário estudar o que já estava desenvolvido e adaptar a implementação de acordo com os novos requisitos.

Para a sua instalação, foi apenas necessário disponibilizar o projeto no IIS do servidor do cliente (que já se encontrava instalado com outra solução da empresa) e validar se a conexão à base de dados era realizada sem problemas. Neste caso, a base dados estava instalada no mesmo servidor, pelo que não houve dificuldades no decorrer da instalação, nem das atualizações seguintes. A Figura 5.1 apresenta a estrutura da aplicação *aWMS*.

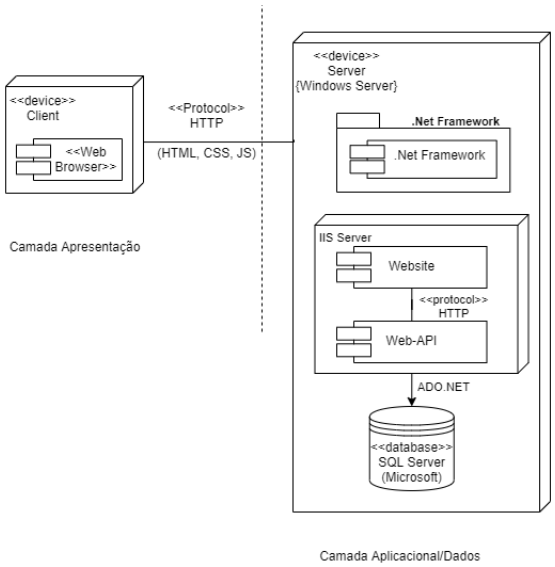

Figura 5.1 – Diagrama de *deployment*

Assim como acontece no primeiro projeto, esta aplicação tem uma estrutura distribuída cliente-servidor. Neste caso, o servidor onde está alojado o *website* é também o servidor onde está a base de dados, estando garantida a comunicação entre a aplicação e a base de dados, estando depois disponível através do *browser*. A especificidade desta aplicação é que apenas está disponível na rede interna do cliente, ou seja, é necessário que o cliente esteja ligado à rede da empresa, seja nas instalações ou através de uma *Virtual Private Network* (VPN). O servidor *web* utilizado foi o IIS e a base de dados é gerida através do *Microsoft SQL Server*, uma vez que a base de dados utilizada é a do *PHC*.

A comunicação com a base de dados é feita através de *ADO.NET*, que é um conjunto de classes que permite aceder aos dados em aplicações *.NET Framework*, o que permite separar o acesso aos dados da manipulação dos mesmos [32]. No entanto, esta manipulação dos dados foi feita através da *Entity Framework*, que consiste numa *framework* ORM (*Object-Relational Mapping*) *open-source*, que será explicada mais à frente (ver secção 5.1.3) [33]. A Figura 5.2 apresenta a arquitetura de uma aplicação que utiliza *Entity Framework*.

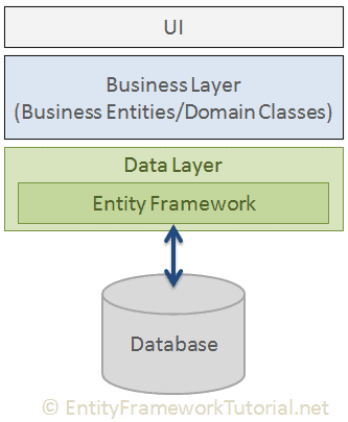

Figura 5.2 – Arquitetura de uma aplicação com Entity Framework [33]

Sendo uma base de dados *PHC*, esta é uma base de dados relacional, como já foi referido anteriormente no subcapítulo 4.1.2. A Figura 5.3 apresenta o modelo relacional das tabelas utilizadas para este projeto, não sendo apresentadas todas as tabelas que constituem a base de dados. Mais uma vez, não serão apresentados os atributos completos uma vez que existem tabelas desenvolvidas pela marca e outras especificamente para processos internos do cliente.

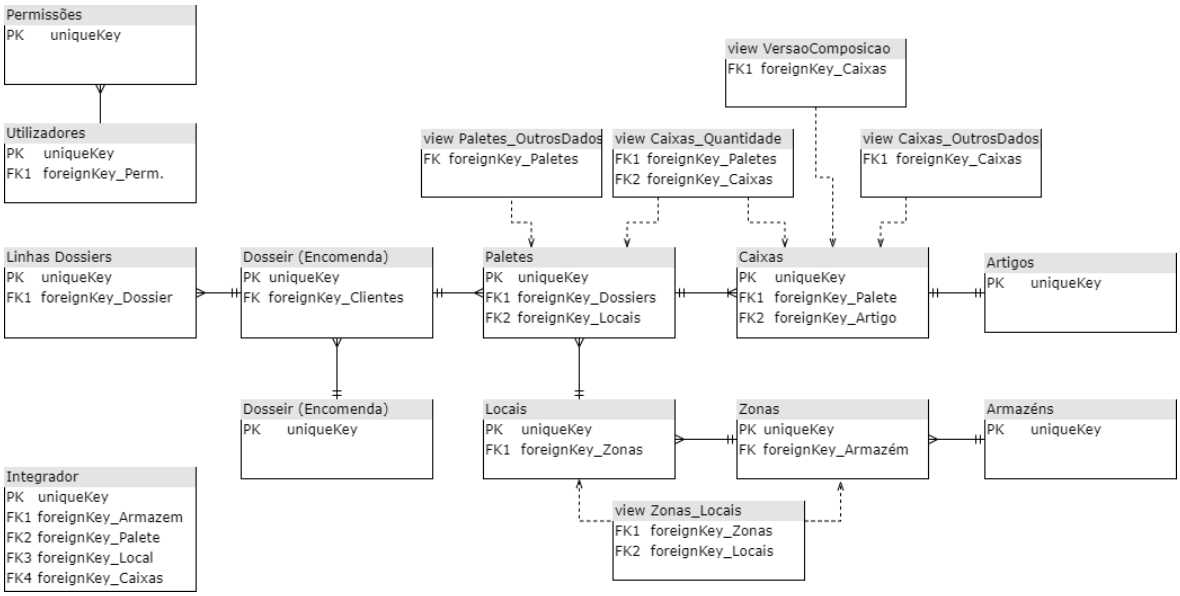

Figura 5.3 – Modelo relacional parcial da base de dados utilizada

Apenas estão representadas as tabelas utilizadas na aplicação *aWMS*, uma vez que as restantes não são alteradas/consultadas pela aplicação. Assim sendo, as tabelas mais importantes no desenvolvimento desta aplicação são as tabelas Paletes e Caixas, uma vez que todos os processos envolvem uma ou ambas. A tabela Integrador é uma tabela que, mesmo apresentando chaves estrangeira, não se encontra relacionada com nenhuma das restantes, servindo apenas como tabela intermédia para o armazenamento de informação a processar. As restantes tabelas servem como auxiliares ou como parametrizações da aplicação *aWMS*, como são os casos das tabelas Permissões, Zonas e Locais.

5.1.3. Tecnologia utilizada

Uma vez que o projeto já tinha sido iniciado, não houve necessidade de realizar um estudo sobre quais as tecnologias que poderiam ser utilizadas, estando escolhido o *C#*, em conjunto com a utilização da *Entity Framework*, *Razor*, *JavaScript* e *jQuery*.

C# é uma linguagem de programação lançada em 2002 e desenvolvida para a *.Net Framework*, tendo a sua origem em *C*, acabando por ser idêntica a outras linguagens como *C++* ou *Java* [34]. Havendo já algum conhecimento de outras linguagens semelhantes, o processo de adaptação foi relativamente simples, até porque esta linguagem partilha semelhanças com o *VB.Net*, abordado no projeto *PHC Web* (secção 4.1).

Entity Framework é uma *framework* ORM *open-source* (como já referido anteriormente), que permite trabalhar modelos em vez de tabelas [33]. Ou seja, em vez de ser necessário descrever todas as comunicações com a base de dados através de *ADO.Net*, com a *Entity*

Framework são trabalhados modelos relativos à camada de negócio (chamados de *Entity Data Model* (EDM)), sendo depois os métodos da *framework* que criam as *queries* necessárias para comunicar com a base de dados. As *queries* podem ser escritas em *LINQ* ou em *Entity SQL*, sendo que neste caso foi utilizado *LINQ to Entities*, utilizando métodos para a criação das *queries* [35].

No desenvolvimento da aplicação, optou-se por utilizar *LINQ to Entities* baseada em métodos, ou seja, as *queries* foram escritas com base em métodos e expressões *lambda*. A utilização de *LINQ* permite prevenir ataques *SQL injection*, uma vez que as *queries* não são construídas através de concatenações, mas sim passadas para o SQL (*Structured Query Language*) como parâmetros [35]. A Figura 5.4 apresenta um exemplo da construção destas *queries*.

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    var query = context.Products
        .Select(product => new
        {
            ProductId = product.ProductID,
            ProductName = product.Name
        });

    Console.WriteLine("Product Info:");
    foreach (var productInfo in query)
    {
        Console.WriteLine("Product Id: {0} Product name: {1} ",
            productInfo.ProductId, productInfo.ProductName);
    }
}
```

Figura 5.4 – Exemplo de código em *LINQ* [36]

A Figura 5.5 representa a arquitetura geral do *Entity Framework*. À esquerda está discriminado o conceito EDM, que é constituído pelas classes modelo e as suas relações (modelo conceptual), pelo modelo da base de dados (modelo de armazenamento) e o mapeamento entre ambas. À direita está a "lógica" que estabelece a comunicação com a base de dados: a primeira linha são as linguagens utilizadas; a segunda linha é o serviço que é responsável por manter o rastreamento dos objetos, mantendo o registo de possíveis alterações nas propriedades do objeto; a linha seguinte é a camada onde ocorre a conversão de *LINQ* ou *Entity SQL* para a *query* SQL, passando a informação à camada seguinte, que efetivamente estabelece a comunicação com a base de dados através de *ADO.Net* [35],[37].

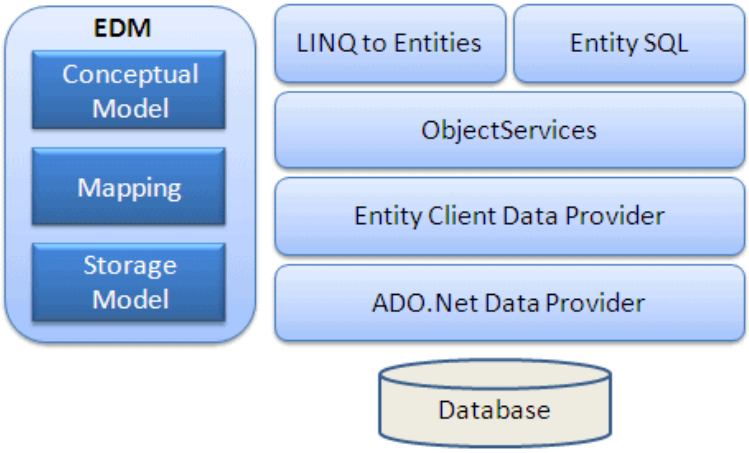

Figura 5.5 – Arquitetura da *Entity Framework* [37]

A *Entity Framework* tem três tipos de abordagens [33], [35]:

- *Database-first*, onde o modelo é criado partindo da base de dados já existente. Este processo pode ser simplificado criando os modelos através do assistente do *Visual Studio*;
- *Code-first*, esta abordagem pressupõe que não existe uma base de dados e que esta será gerada através de *migrations* criadas a partir dos modelos desenvolvidos;
- *Model-first*, aqui é possível desenhar os modelos e depois gerar o esquema da base de dados.

No desenvolvimento desta aplicação foi utilizada a abordagem *Model-first*, uma vez que era a que já estava a ser utilizada. No entanto, uma vez que já existia uma base de dados, teria sido possível utilizar a *Database-first*, simplificando o processo de mapeamento quer das tabelas, como das vistas e dos procedimentos.

O *Razor* é uma ferramenta com sintaxe de *script*, que permite criar *templates* e páginas *web*. Esta não é uma nova linguagem, mas permite escrever código em linguagens já conhecidas (como *C#* ou *VB.Net*). O *Razor* acaba por ser um complemento ao HTML, permitindo uma sintaxe mais limpa e com funcionamento dinâmico [38].

O *JavaScript* é uma linguagem de programação *just-in-time*, orientada a eventos e que permite manipular o *Document Object Model* (DOM). É uma linguagem que apenas "corre" no *browser* e, portanto, é mais rápida porque não precisa de comunicar com o servidor [39], [40].

Além da utilização do *JavaScript*, foram utilizadas algumas bibliotecas que têm como base esta linguagem, sendo elas:

- *jQuery* é uma biblioteca *JavaScript*, que permite manipular o DOM, assim como eventos e até simplificar o uso do *Ajax* (utilizado para realizar pedidos à API); isto significa que não seria necessário recorrer a outra biblioteca para a realização de pedidos ao servidor [41];
- *jQuery UI*, que funciona como uma extensão do *jQuery*, mas direcionado para a interação do utilizador com as interfaces; permite criar interações (como tornar os objetos *draggable*) ou dar efeitos aos objetos [42];
- *Bootbox.js*, que permite criar caixas de diálogo de forma rápida e personalizada; esta biblioteca utiliza os *modals* do *Bootstrap*, mas o programador não tem de se preocupar com a gestão destes [43];
- *DataTables*, que é uma extensão do *jQuery*, que permite transformar qualquer tabela HTML, de forma a tornar a informação mais acessível [44].

Além das tecnologias, é importante referir que foram utilizadas duas ferramentas para o desenvolvimento da aplicação, sendo elas o *Azure DevOps* e o *Visual Studio*.

O *Azure DevOps* é um serviço *Microsoft*, que fornece vários serviços para o desenvolvimento e implementação contínua de aplicações [45]. O serviço utilizado durante o projeto foi o *Azure Repos*, que permite criar repositórios para controlo de versões. Estes sistemas permitem a monitorização do código, sendo possível consultar mudanças realizadas em versões anteriores. O *Azure Repos* tem dois tipos de controlo de versões: *Git* e *Team Foundation Version Control* (TFVC); foi este último o utilizado. Este é um sistema de controlo de versões centralizado, que permite manter uma cópia de cada ficheiro no posto de desenvolvimento dos membros da equipa [46].

O *Visual Studio* é um *Integrated development environment* (IDE), ou seja, um ambiente de desenvolvimento integrado, que permite criar e/ou editar código, assim como fazer o *debug* e a publicação da aplicação. Este IDE permite compilar o código para a sua publicação, assim como permite testar sem ter de publicar, facilitando o processo de *debug*. Tem ainda disponível a funcionalidade de associar um repositório ao projeto, facilitando o controlo de versões [47].

5.2. Modelo de negócio

A aplicação *aWMS* foi desenvolvida para auxiliar processos presentes no cliente Y[4] e que seria integrado com o ERP *PHC*. Este cliente dedica-se à transformação de matéria plástica, através de vários processos e possui dois armazéns de produção. O cliente necessitava de um sistema que lhe permitisse ter uma visão virtual dos armazéns físicos, havendo a necessidade de garantir a mobilidade do sistema.

O objetivo inicial do cliente era apenas o mapeamento virtual dos armazéns, passando depois a haver a necessidade de assegurar o processo de qualidade após a produção. Assim, os requisitos são estipulados de acordo com os pressupostos que têm de ser cumpridos na qualidade após a produção e na gestão do armazém. Existem ainda alguns requisitos que foram desenvolvidos para tentar diminuir a probabilidade do erro humano provocado por distração e/ou desconhecimento.

Inerente a esta aplicação, existe apenas um único processo principal, que depois pode seguir diversos caminhos, de acordo com o estipulado pelo departamento de qualidade do cliente. O esquema deste processo está apresentado na Figura 5.6.

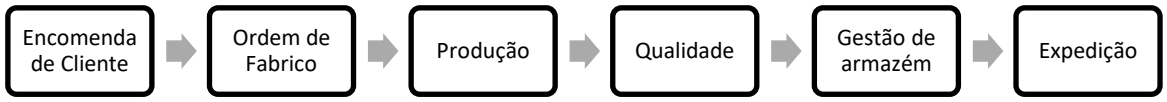

Figura 5.6 – Processo de produção/logística do cliente Y

A Encomenda de Cliente dará origem a uma Ordem de Fabrico, sendo estes dois passos realizados no *PHC*. Após a criação da Ordem de Fabrico, dá-se início à produção, que utiliza uma solução Arentia para auxílio na área da produção. No fim do produto produzido, é da responsabilidade do departamento de qualidade, garantir que o produto chega ao cliente de acordo com o pedido, após o qual o produto será arrumado no armazém até ao momento da sua expedição.

Deste processo, surgem dois subprocessos: um referente à qualidade e outro à gestão de armazém. Assim a Figura 5.7 apresenta o processo de qualidade, que estabelecerá ligação com o processo da Figura 5.8, no ponto "Por arrumar". Sucintamente, a área da qualidade

[4] Devido ao RGPD, não será apresentado o nome do cliente, assim como se irá ocultar qualquer tipo de informação que possa servir para identificar o cliente em causa.

tem de decidir se uma palete será aprovada ou condicionada; estando condicionada, terá de ser novamente aprovada ou rejeitada.

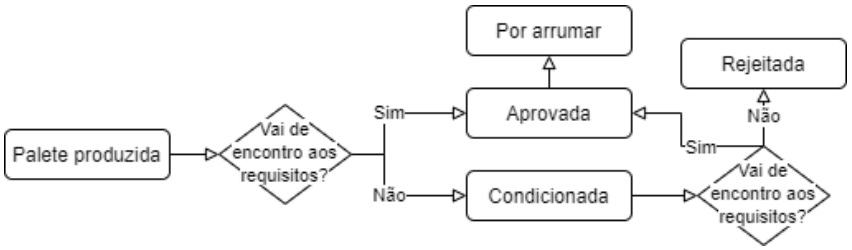

Figura 5.7 – Processo de qualidade do cliente Y

Já o subprocesso da gestão de armazém, representado na Figura 5.8, apresenta quatro hipóteses para uma palete (esteja arrumada ou não): condicionar, caso esteja danificada e fazendo com que regresse ao subprocesso anterior; reservar, expedir para o cliente ou transferir de armazém.

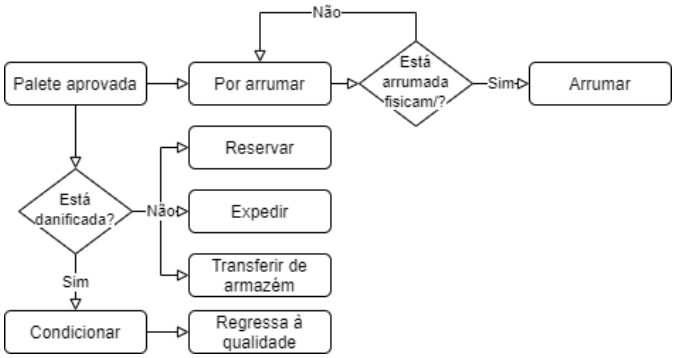

Figura 5.8 – Processo de gestão de armazém do cliente Y

A lista de requisitos foi construída partindo do que já existia desenvolvido, em conjunto com os requisitos funcionais e não funcionais que o cliente reportou nas várias reuniões realizadas. Esta lista foi depois traduzida para US, de forma a simplificar a sua leitura e a execução dos testes funcionais, para os processos já apresentados e restantes funcionalidades.

As US (disponíveis no Anexo B, na Tabela B.1) e os requisitos (disponíveis no Anexo C, na Tabela C.1[5]) foram divididas por ecrã/funcionalidade: Geral, Utilizadores, Armazéns, Aprovação de paletes, Gestão de armazéns, Detalhes da palete, Transferências, Verificar caixas, Condicionar paletes e Rejeição. Cada grupo apresenta as funcionalidades e as regras necessárias em cada área. Os pontos que requereram mais atenção foram as transferências e a rejeição, uma vez que possuem várias regras que devem ser validadas.

[5] O Anexo C está relacionado com o Anexo B: os primeiros dois dígitos depois das letras RQ identificam a US.

5.3. Implementação[6]

A implementação deste projeto não requereu um período de adaptação como o anterior, uma vez que já tinha sido adquirida experiência em programação *C#* e com a *Entity Framework*. Houve, no entanto, necessidade de explorar o que já estava realizado e qual o ponto inicial do projeto, para saber como prosseguir.

Numa primeira fase, houve necessidade de reestruturar as vistas, como forma de reorganizar melhor o espaço existente, adaptar o desenho responsivo (para ser possível trabalhar em *tablets* de 10 polegadas[7]) e fazer adaptações de acordo com o pedido do cliente.

No que se refere ao desenho responsivo, o desenho também foi testado em dispositivos móveis com cinco polegadas, mas estes revelaram-se ecrãs de dimensão demasiado reduzida para as funcionalidades existentes. Além do *tablet* de 10 polegadas, ainda se adaptaram os ecrãs para dispositivos de sete polegadas, mas também estes dificultam a execução de algumas funcionalidades por parte do utilizador. Um ponto importante era ser possível arrumar itens através do movimento *Drag&Drop*.

Para o desenho responsivo, foram utilizados maioritariamente os *breakpoints* do *Bootstrap*, tentando evitar (sempre que possível) a criação de *media queries* em CSS. Quando necessário, foram definidas *media queries* com base na largura mínima ou máxima do *viewport* em que teria de ser realizado o "*break*". Os tamanhos utilizados foram os disponibilizados na documentação do *Bootstrap* (o *Bootstrap* define um conjunto de larguras máximas e mínimas para diferentes tipos de dispositivos: muito pequenos, pequenos, médios, grandes e muito grandes) [48].

Durante toda a implementação foi efetuada otimização de código, quer extraindo parcelas de código para funções que pudessem ser chamadas por métodos diferentes, quer através do reaproveitamento de funções com pequenas diferenças. A redução de código repetido em múltiplos métodos garante que processos idênticos mantenham o mesmo tipo de comportamento. Por exemplo, ao ver os detalhes da palete, a função que faz o pedido ao servidor é sempre a mesma, assim como a vista devolvida pelo servidor é sempre a mesma. Outro exemplo são os processos de rejeição e de transferência entre armazéns: uma vez que

[6] Durante este subcapítulo, todas as imagens apresentadas foram sujeitas a edição para ocultar informação confidencial/sensível. A presença de marca de água deve-se ao facto de ser um produto Arentia. Por este mesmo motivo, não será apresentado código relacionado com o projeto, considerando-se o mesmo como propriedade da empresa e, por isso, informação confidencial.

[7] A resolução do monitor do computador era 1366x768 e a resolução do *tablet* de 800x1280 (em modo retrato).

a única diferença entre eles é o preenchimento do motivo de rejeição, em que ambas as funções do lado do cliente fazem um pedido para o mesmo método.

Existem US descritas que envolveram apenas a configuração no *PHC CS*, não sendo necessário qualquer tipo de desenvolvimento, como acontece nas US relativas a utilizadores e permissões (US1.1 e US1.2).

5.3.1. Versão inicial

A Figura 5.9, a Figura 5.10 e a Figura 5.11 apresentam alguns dos ecrãs que existiam na versão original, antes de terem sido realizadas as alterações pedidas. A primeira figura apresenta a grelha de arrumação de paletes, a segunda imagem é o ecrã ao consultar um local (onde iriam aparecer as paletes arrumadas na parte superior, em formato de separador), e a terceira imagem é a consulta dos detalhes de uma palete, presente na lista de paletes à direita do ecrã.

Originalmente, o ecrã de "Aprovação de paletes" não existia no *aWMS*, estando este processo pensado para ser realizado no *PHC*. No entanto, visto que, em qualquer das plataformas (*PHC* ou *web*), o processo teria de ser desenvolvido de raiz, este foi passado para o *aWMS*. Assim, é dada mobilidade para que este processo possa ser realizado diretamente no local, através de um *tablet*.

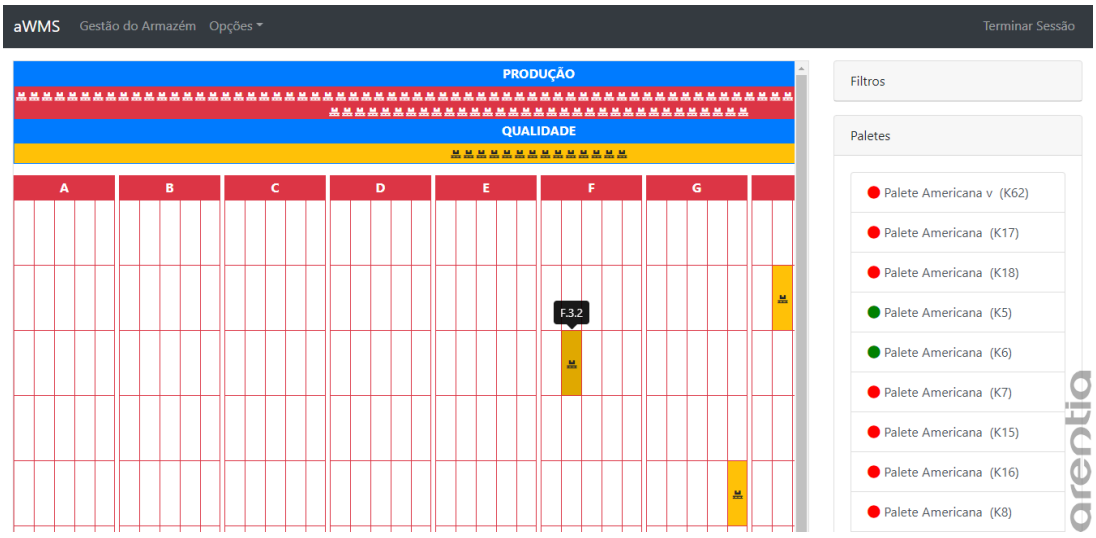

Figura 5.9 – Página inicial da versão original

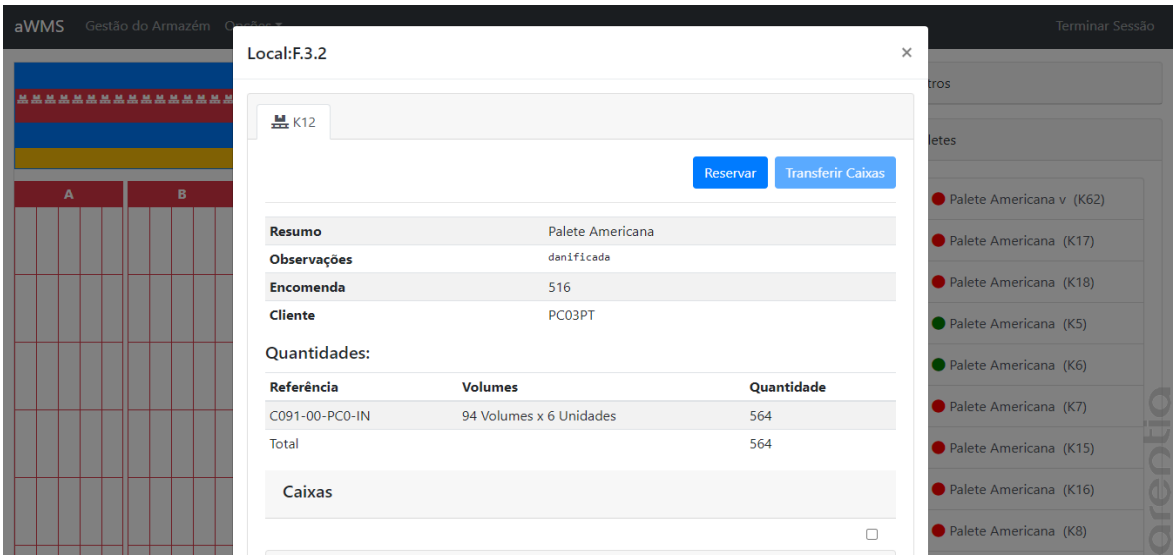

Figura 5.10 – Consulta de paletes de um local, na versão original

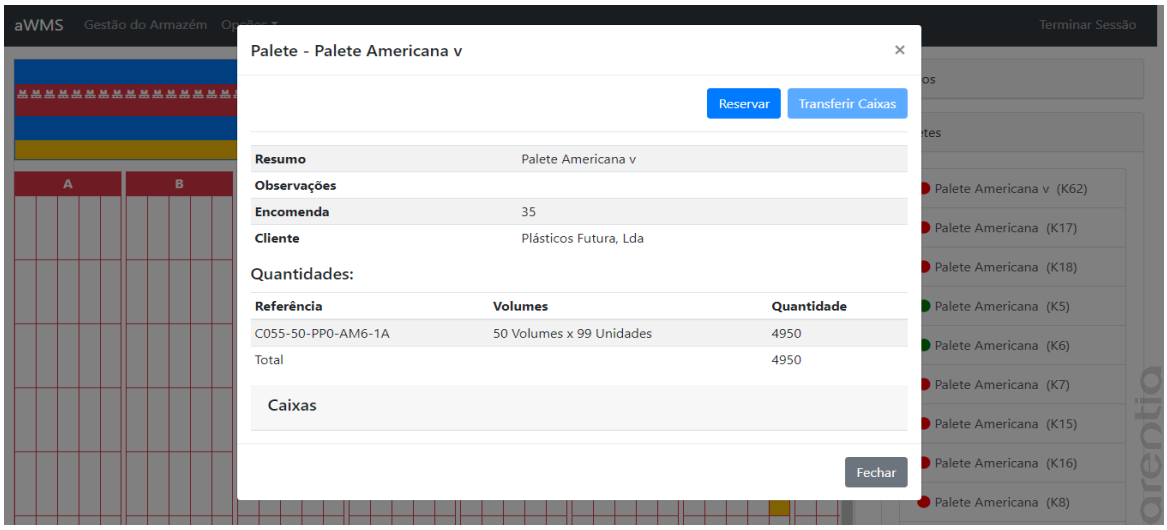

Figura 5.11 – Consulta de paletes da listagem, na versão original

5.3.2. Gestão de armazém

A Figura 5.12 (versão *desktop*) e a Figura 5.13 (versão *tablet*) apresentam o ecrã final da Gestão de Armazém, sendo esta a versão que se encontra para testes, no cliente. A maior diferença, quando comparando com a versão inicial, é a forma de apresentação das zonas/locais, sendo possível visualizar logo quais os locais ocupados e quantas paletes existem no local (US4.1, US4.2 e US4.3). Passou a existir apenas uma zona "Por arrumar", em vez de existirem duas zonas ("Produção" e "Qualidade"); é nesta zona que se vão encontrar todas as paletes produzidas, validadas e que ainda não estão arrumadas (US4.4).

A arrumação das paletes é possível através de eventos *Drag&Drop* (arrastar e largar) (US4.6 e US4.8). Para isso, basta arrastar as paletes (através das setas que se encontram em cada item da listagem de paletes) e largar num dos locais da grelha ou na zona "Por Arrumar".

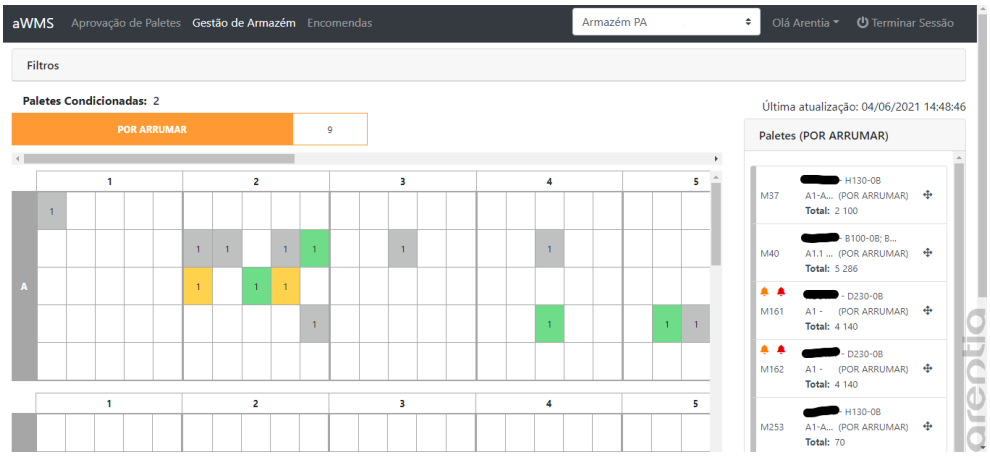

Figura 5.12 – Ecrã inicial da Gestão de Armazém em *desktop*

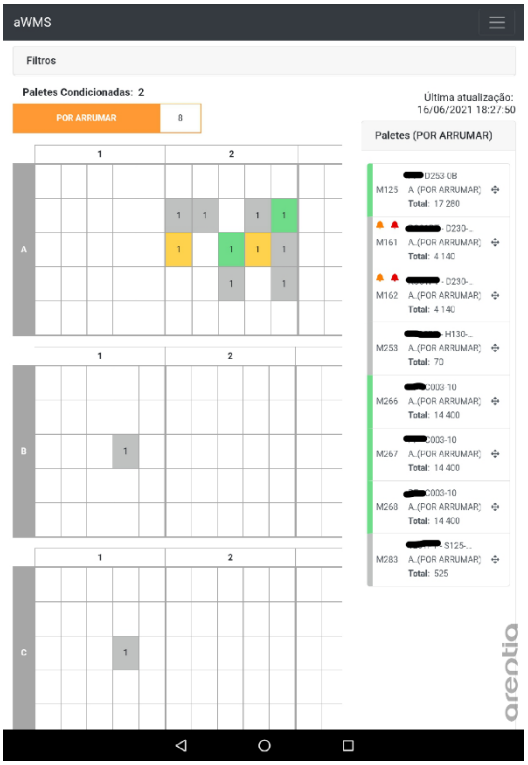

Figura 5.13 – Ecrã inicial da Gestão de Armazém em *tablet*

Inicialmente, esta funcionalidade foi desenvolvida com recurso à API do HTML, mas não era suportada pelos dispositivos móveis, assim como começaram a aparecer algumas dificuldades em despoletar certos comportamentos. Mais tarde, esta funcionalidade foi trocada por uma biblioteca de *JavaScript* (*jQuery UI*), que facilitou a implementação dos comportamentos pedidos pelo cliente (alteração de cor ao passar, despoletar *tooltip*, inicialização dos eventos *Drag&Drop*, entre outros). Contudo, também esta biblioteca apresenta limitações nos dispositivos móveis, pelo que teve de ser adicionado um ficheiro ao projeto que fizesse a conversão dos eventos do rato para os eventos *touch*, possibilitando a utilização dos movimentos *Drag&Drop* nos *tablets*.

A Figura 5.14 mostra os filtros disponibilizados para a procura de informação no ecrã Gestão de Armazém (US4.5). Já na Figura 5.15 é possível ver qual o resultado após a aplicação dos filtros. Ao aplicar filtros, apenas os locais com paletes que correspondem ao filtro aplicado, irão aparecer com cores, todos os outros apenas mantêm o número de paletes. No caso de ser pesquisada uma referência ou uma versão de composição, irá aparecer a quantidade total do artigo pesquisado no armazém, por zona e em cada palete onde este exista.

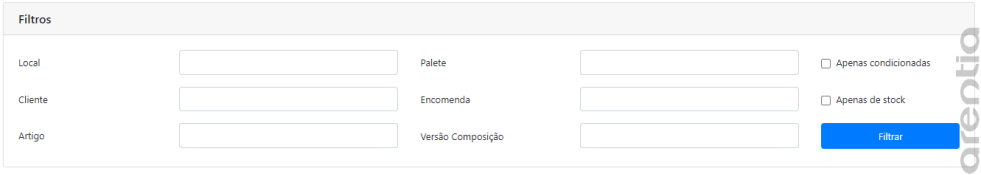

Figura 5.14 – Filtros disponibilizados

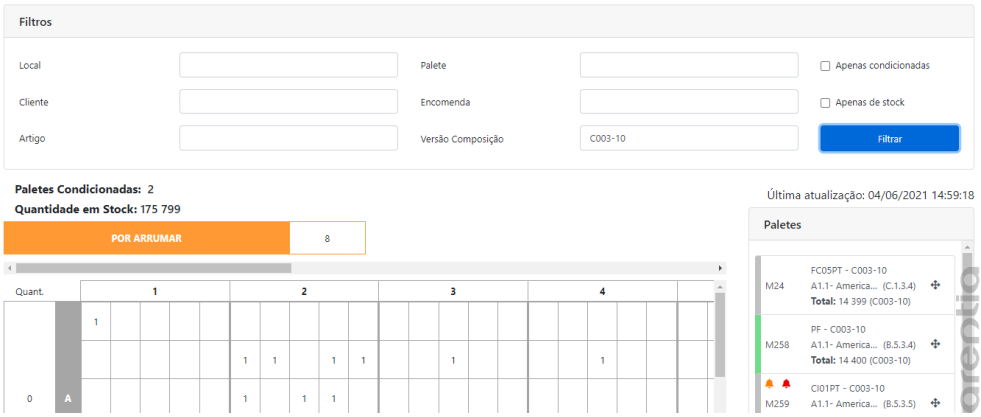

Figura 5.15 – Após a aplicação de filtros (neste caso, pesquisa por Versão Composição)

Para consultar os detalhes de uma palete (US4.7 e US5.1) basta clicar sobre o item da mesma, que irá aparecer a informação da palete, como se pode ver na Figura 5.16 (versão *desktop*) e na Figura 5.17 (versão *tablet*). Na Gestão de Armazém apenas são possíveis algumas ações, sendo elas: transferências (peças, caixas e entre armazéns), condicionar paletes e verificar caixas.

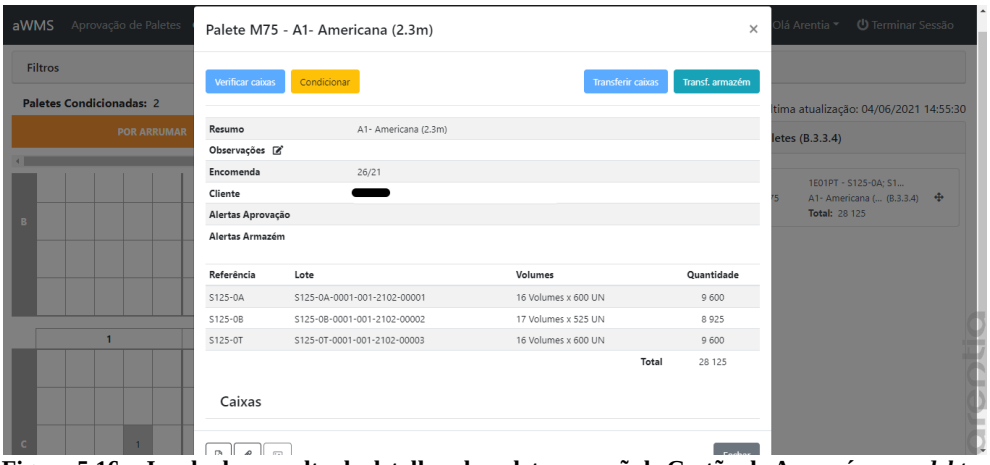

Figura 5.16 – Janela de consulta de detalhes da palete no ecrã de Gestão de Armazém, em *dektop*

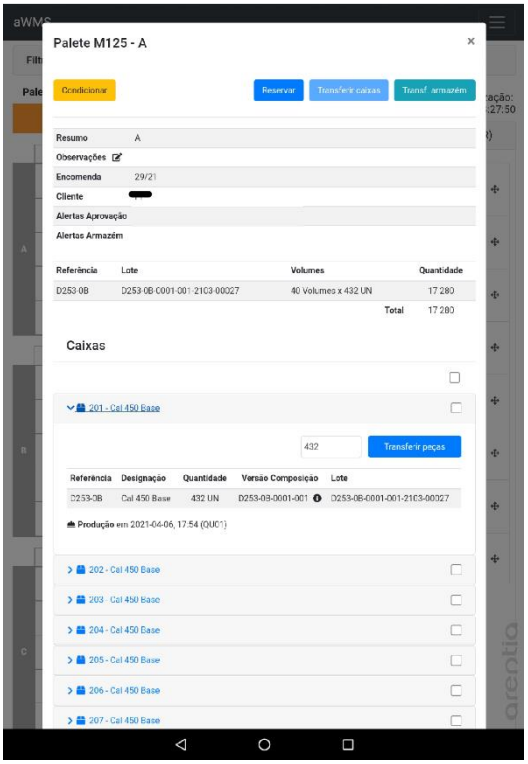

Figura 5.17 – Janela de consulta de detalhes da palete no ecrã de Gestão de Armazém, em *tablet*

5.3.3. Aprovação de paletes

A Figura 5.18 (versão *desktop*) e a Figura 5.19 (versão *tablet*) apresentam o ecrã da Aprovação de Paletes, sendo que este ecrã não se encontrava na versão original. Este ecrã é constituído por uma tabela com todas as paletes produzidas e não aprovadas, assim como todas as paletes condicionadas (US3.1) (mesmo que já tenham sido aprovadas anteriormente).

Esta tabela é constituída pela informação mais relevante, tendo presente campos que permitem a filtragem das linhas (US3.2 e US3.3). No caso da coluna "Condicionada", aparece a informação de quando e quem condicionou a palete, assim como o lugar onde ela se encontra.

Para facilitar a pesquisa de dados na tabela, foi aplicado o *plugin DataTables*, que permite realizar a pesquisa nas colunas da tabela, sem ser necessário realizar um pedido ao servidor. Ou seja, a pesquisa é feita na informação que o utilizador está a ver no momento, permitindo que os vários campos sejam preenchidos para ir refinando a procura. Neste caso, a utilização de *DataTables* não condiciona o funcionamento do ecrã, uma vez que este deverá possuir poucas linhas na versão de produção (devido aos processos internos existentes no cliente).

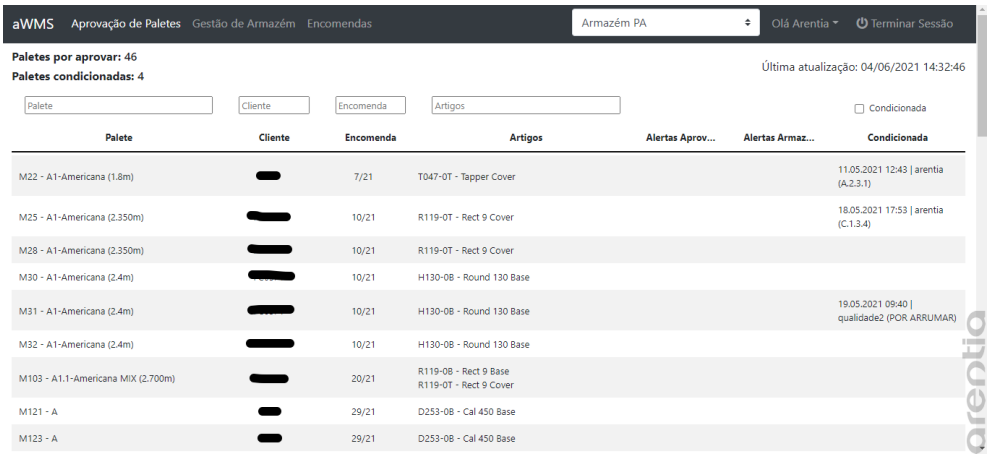

Figura 5.18 – Ecrã inicial da Aprovação de Paletes em *desktop*

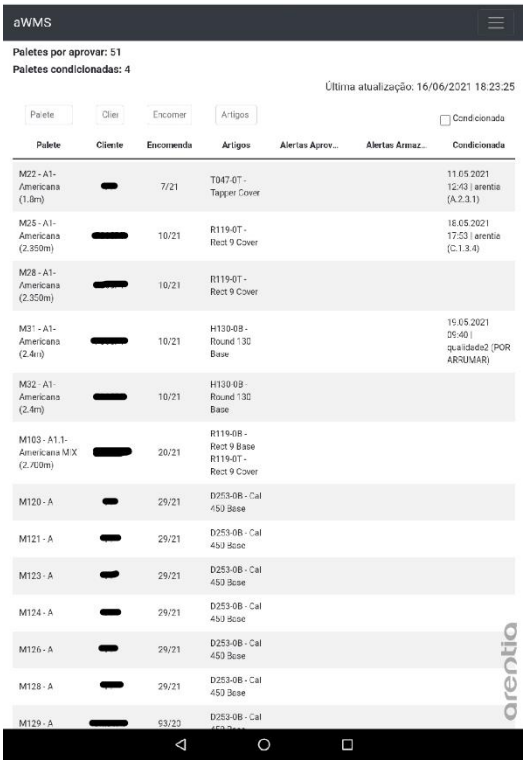

Figura 5.19 – Ecrã inicial da Aprovação de Paletes em *tablet*

Para consultar os detalhes de uma palete (US3.4 e US5.1) basta clicar sobre a linha da mesma e irá aparecer a informação da palete, como se pode ver na Figura 5.20 (versão *desktop*) e na Figura 5.21 (versão *tablet*). Na Aprovação de Paletes estão disponíveis as ações: aprovação de palete, transferências (peças e caixas), rejeição (peças e caixas), condicionar paletes, verificar caixas e corrigir quantidade máxima das caixas.

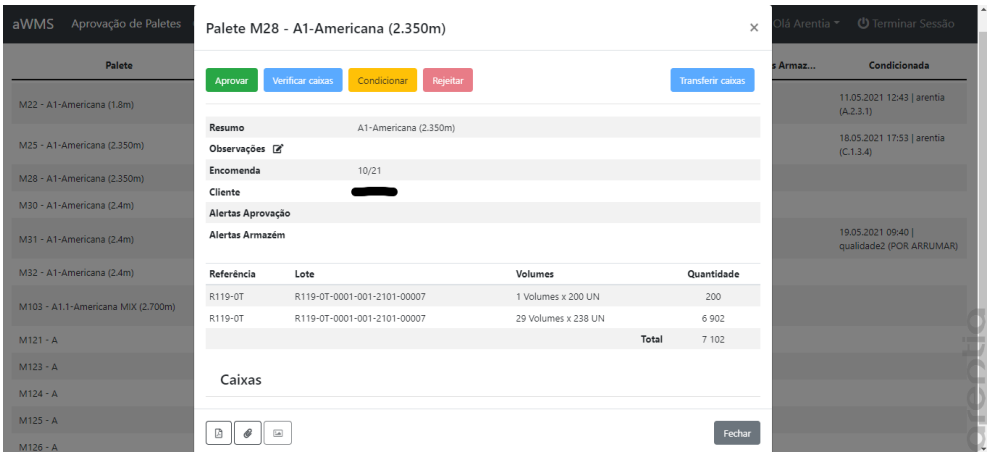

Figura 5.20 – Janela de consulta de detalhes da palete no ecrã de Aprovação de Paletes em *desktop*

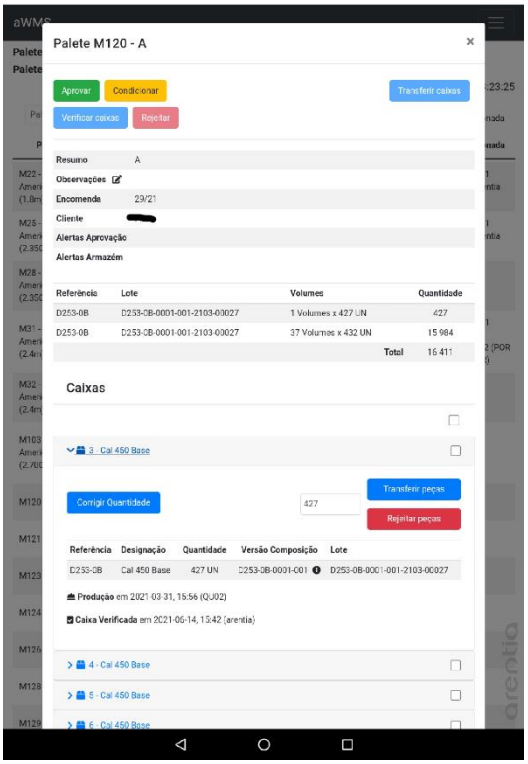

Figura 5.21 – Janela de consulta de detalhes da palete no ecrã de Aprovação de Paletes em *tablet*

5.3.4. Detalhes da palete

A Figura 5.22 mostra os detalhes das caixas e as opções disponíveis no ecrã de Aprovação de Paletes, enquanto a Figura 5.23 apresenta os detalhes e as opções disponíveis no ecrã de Gestão de Armazém (US5.4 e US5.5). No caso da Figura 5.22 é possível ver como aparecem duas caixas, estando a primeira completa e a segunda incompleta (o facto de estar incompleta permite ver uma das ações disponíveis) (US3.7).

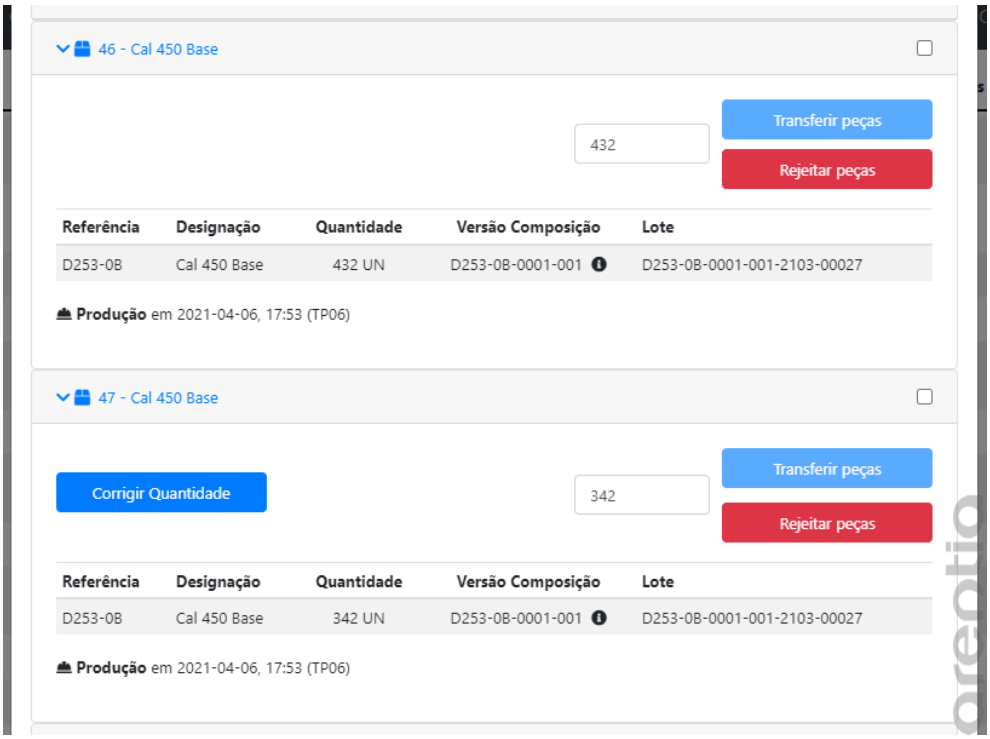

Figura 5.22 – Detalhes das caixas na janela de detalhes da palete, no ecrã Aprovação de Paletes

Já na Figura 5.23 é possível visualizar como aparecem referenciadas as caixas partilhadas (são caixas que têm duas referências de artigos diferentes), assim como a única ação sobre as caixas que está disponível neste ecrã. No caso das caixas partilhadas, tem de ser expandida a subcaixa pretendida para aparecerem os campos relacionadas com as ações.

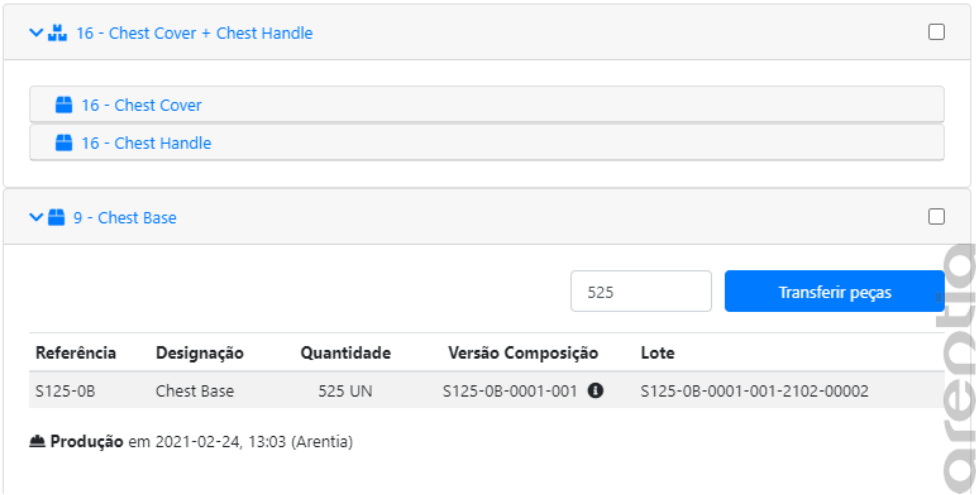

Figura 5.23 – Detalhes das caixas na janela de detalhes da palete, no ecrã Gestão de Armazém

Para a apresentação das caixas foi contruída uma lista, formada por *cards*, com a componente *collapse*. Isto permite ao utilizador visualizar apenas uma caixa de cada vez e focar-se na informação que precisa, em vez de consultar a palete com toda a informação das caixas visível (que podem ser em elevado número). Além disto, sempre que possível, a informação é disponibilizada em forma de tabela para facilitar a consulta/leitura.

5.3.5. Ações sobre a palete

A Figura 5.24 apresenta a janela de confirmação da aprovação da palete selecionada (US3.5). Já a Figura 5.25 mostra a janela para inserir a nova quantidade máxima da caixa selecionada (US3.6 e US3.8). Esta opção está disponível nos detalhes da caixa como apresentado anteriormente na Figura 5.22. Estas ações são exclusivas do ecrã de Aprovação de Paletes.

Para as janelas referentes a estas ações foi utilizada a biblioteca *Bootbox.js*, uma vez que permitia apresentar a informação de forma mais consistente com o *design* da aplicação, em vez da utilização das janelas de alerta e confirmação que vêm por defeito no *JavaScript*. A implementação é rápida e permite escolher os campos fornecidos, assim como as funções a realizar para cada opção disponibilizada ao utilizador.

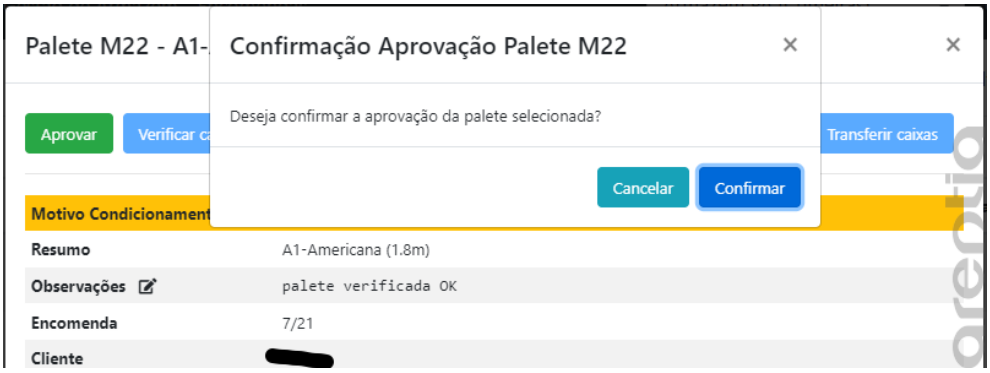

Figura 5.24 – Ação de aprovação de paletes

Figura 5.25 – Ação de corrigir a quantidade máxima das caixas

Do lado do servidor, no caso da aprovação de paletes, é validado se a palete existe e se não está inativa, sendo aprovada caso exista. Se a palete a ser aprovada estiver condicionada, este campo é colocado a *false* e os dados atualizados. Para a correção da quantidade máxima das caixas, é validado se o número enviado é maior que zero (esta validação também é feita do lado do cliente) e se a caixa existe. Caso seja verdade, é validado se o número enviado é superior ou igual à quantidade existente na caixa; caso isto não se comprove, não é alterada a caixa e é alertado o utilizador de que a quantidade não é válida.

Outra ação exclusiva do ecrã de Aprovação de Paletes é a rejeição (US9.1). Na Figura 5.26 é possível ver a janela para rejeição de caixas/peças (US9.4 e US9.8). A diferença entre estas rejeições é que, no caso das peças, tem de ser indicado o número de peças antes de clicar em "Rejeitar", nos detalhes da caixa (US9.3), enquanto que, para as caixas, basta selecionar as caixas a rejeitar e clicar em "Rejeitar", nos detalhes da palete (US9.7).

No caso das caixas partilhadas, para rejeitar peças tem de ser escolhida a subcaixa, respeitante às peças que se está a rejeitar (US9.5). Já para as caixas, apenas é necessário escolher a caixa principal para rejeitar todas as subcaixas que lhe estão associadas (US9.9).

Figura 5.26 – Ação de rejeitar caixas/peças de uma palete

Também esta janela é construída com a ajuda da biblioteca *Bootbox.js*. A *dropdown* disponível é construída pela biblioteca, sendo apenas necessário indicar o *array* com a informação que será passada como valor e como texto de apresentação.

Do lado do servidor, é validado se a palete onde está a caixa ainda está disponível, assim como se a(s) caixa(s) também estão disponíveis. Caso estejam, são guardadas as caixas com as quantidades a rejeitar numa tabela intermédia. Após escolher o motivo, é novamente validado se a palete e a(s) caixa(s) ainda estão disponíveis e, em caso afirmativo, é dado início ao processo de rejeição: são removidas as caixas da tabela intermédia e novamente introduzidas, desta vez com o motivo e com a indicação para chamar o procedimento SQL que cria o documento no *PHC*, com a indicação das caixas/peças que foram rejeitadas e o motivo de rejeição. Em caso afirmativo, é apresentado ao utilizador o número do documento criado.

As ações que se seguem estão disponíveis quer no ecrã de Aprovação de Paletes, quer no ecrã de Gestão de Armazém, sendo elas: verificação de caixas (US7.2), condicionamento de paletes (US8.2), transferência de caixas (US6.5) e transferência de peças (US6.1).

Na Figura 5.27 é possível ver a janela de confirmação das caixas verificadas (US7.3). No caso de ser uma caixa partilhada, a verificação é feita na caixa principal e replicada para todas as caixas que lhe estão associadas, ou seja, apenas é necessário escolher a caixa agrupadora (US7.4).

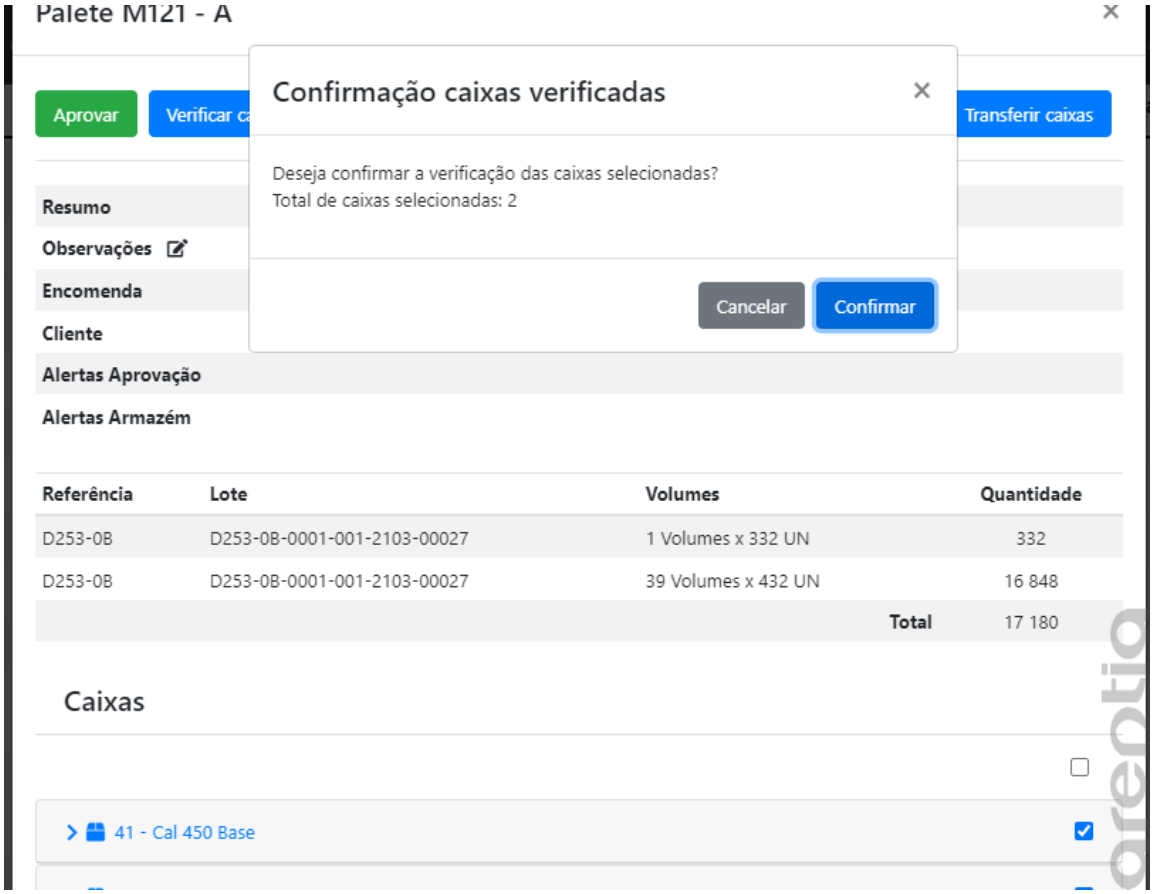

Figura 5.27 – Ação de verificar caixas

Na Figura 5.28 é possível ver a janela para condicionar paletes (US8.1). Após escolher a ação para condicionar, tem de ser escolhido o motivo do condicionamento (US8.4), podendo ser acrescentadas observações. Após a confirmação, a palete fica condicionada, podendo ser arrumada na grelha de locais (US8.5), mas fica impossibilitada a transferência de peças e/ou caixas desta para outras paletes (US8.6).

Neste caso, mesmo sendo utilizada a biblioteca *Bootbox.js*, apenas a *dropdown* é construída pela biblioteca. O campo das observações é adicionado à janela através da função *onShow*.

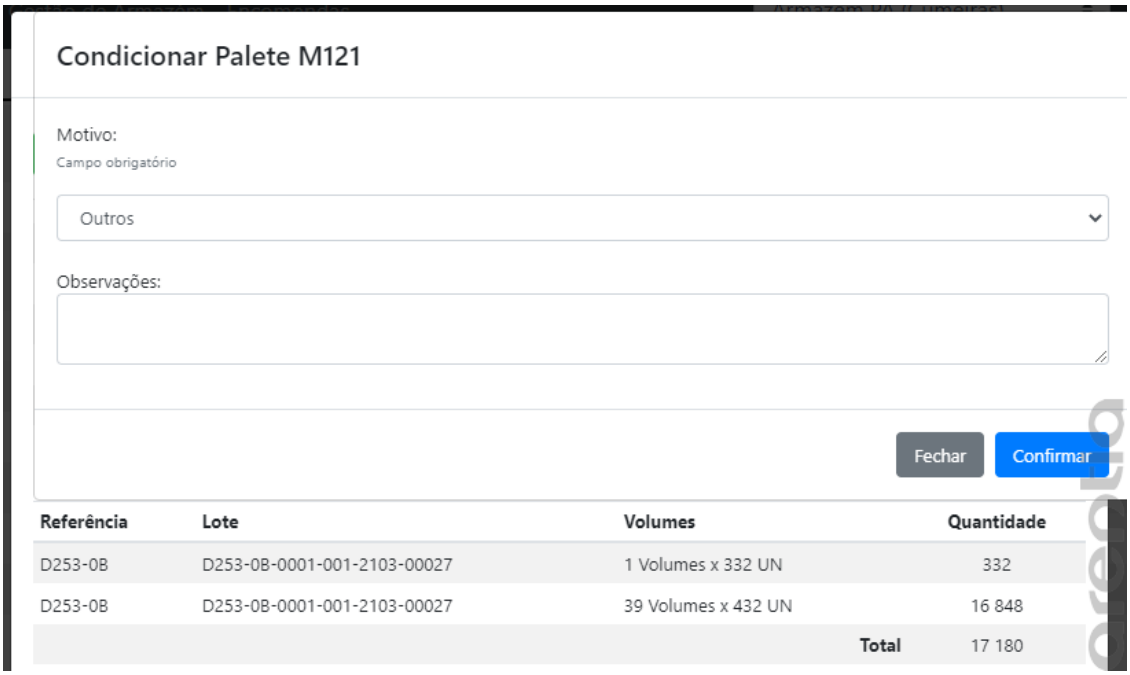

Figura 5.28 – Ação de condicionar paletes

Antes de condicionar uma palete, do lado do servidor, é validado se o motivo está preenchido e, caso sejam passadas observações, estas têm de ter entre 5 a 250 caracteres (estas validações também são feitas do lado do cliente). Após isto, é validado se a palete ainda existe e só depois é colocada a palete como condicionada.

No caso da transferência de peças e da transferência de caixas o processo é idêntico. No caso da transferência de caixas, após selecionar as caixas a transferir (US6.6), irá aparecer a listagem como demonstrado na Figura 5.29, onde é escolhida a palete de destino (US6.7). No caso das caixas partilhadas, apenas é possível escolher a caixa agrupadora, pelo que as subcaixas é que são transferidas (US6.8).

Quando a informação chega ao servidor, este verifica se o código da palete está preenchido e, em caso afirmativo, se a palete destino possui caixas com a mesma versão de composição que as caixas que estão a ser transferidas. Se a palete para onde se está a transferir não tiver nenhuma caixa com a versão de composição igual às caixas que estão a ser transferidas, é dada essa indicação ao utilizador, através de uma mensagem de erro. Caso a palete de origem fique sem caixas, esta é atualizada para ficar como inativa.

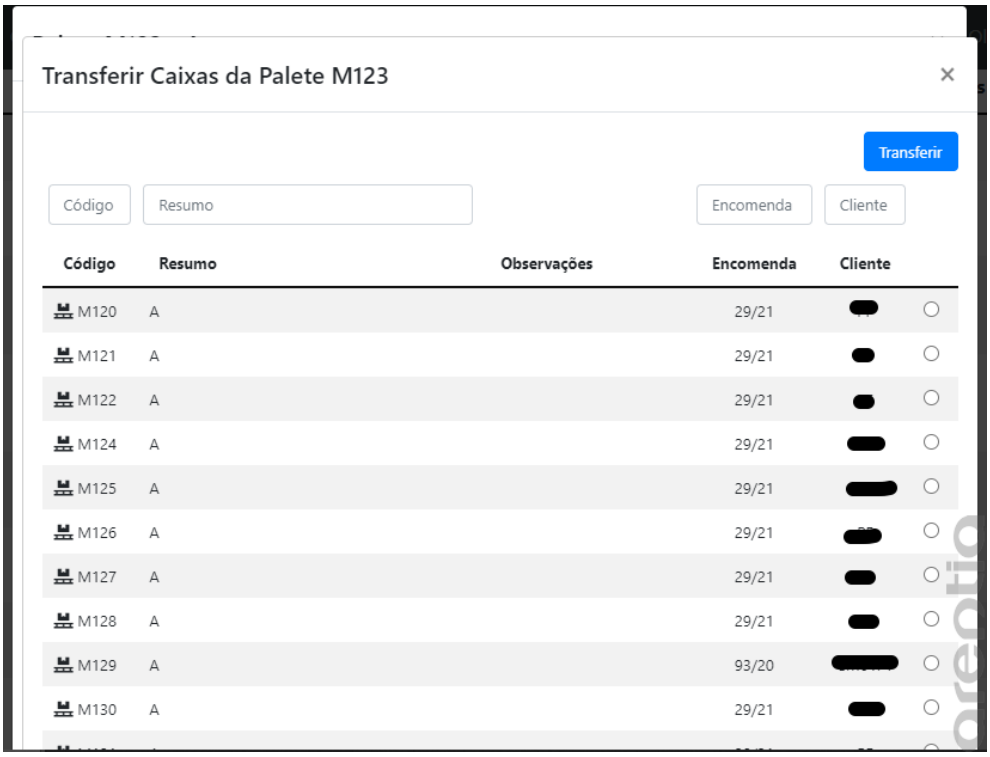

Figura 5.29 – Ação de transferir caixas de uma palete

No caso da transferência de peças, após indicar o número de peças a transferir (US6.2), irá aparecer a listagem como demonstrado na Figura 5.30, onde é escolhida a caixa de destino (US6.3). Sendo caixas partilhadas, este processo é realizado na subcaixa escolhida, sendo feita a transferência apenas de peças com essa referência (US6.4).

Figura 5.30 – Ação de transferir peças de uma caixa

Ao chegar o pedido ao servidor, é validado se a caixa de origem e a caixa de destino ainda existem, assim como se a versão de composição é igual em ambas. Caso isto se verifique, é realizada a transferência das peças. Se a caixa de origem ficar vazia, esta é atualizada para

ficar como inativa e se a palete da caixa de origem ficar sem caixas ativas, também esta é colocada como inativa.

Em ambos os casos, foi utilizado o *plugin DataTables* para facilitar a pesquisa através dos filtros (disponíveis nos cabeçalhos das tabelas). Estes filtros facilitam a pesquisa da palete e/ou caixa que será o destino da transferência (US6.9). No entanto, o filtro apenas é aplicado à informação que se encontra nas linhas principais, ou seja, no caso da transferência de peças, que apresenta linhas de caixas agrupadas por palete, a pesquisa apenas pode ser feita nas paletes. O *plugin* disponibiliza forma de realizar a pesquisa nas linhas-filho, mas a informação tem de estar disponível na linha-pai, mesmo que esteja oculta.

Por último, a Figura 5.31 mostra a janela da ação para a transferência de paletes entre armazéns (US6.10). Deve ser escolhido o armazém de destino, da lista apresentada, e confirmar a ação (US6.12). Ao mudar de armazém, a palete irá aparecer na lista "Por Arrumar" do armazém de destino (US6.13). Esta ação irá criar um documento no *PHC* a indicar a palete/caixas que foram transferidas e qual o armazém de destino. Esta é uma ação exclusiva do ecrã "Gestão de Armazém".

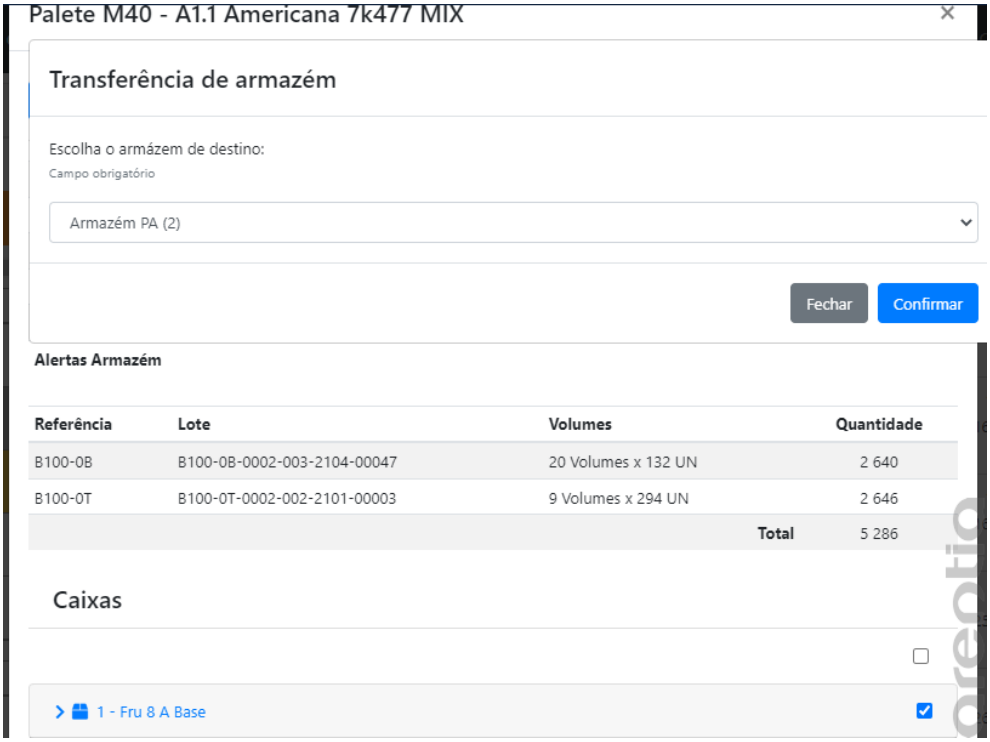

Figura 5.31 – Ação de transferir paletes para outro armazém

Esta ação utiliza a biblioteca *Bootbox.js*, que apenas apresenta os armazéns disponíveis, excluindo o armazém onde se está. No servidor, o processo é igual à rejeição, com a diferença que não é necessário indicar um motivo.

5.3.6. Outros pontos

Havendo a necessidade de guardar palavras-passe em base de dados, estas foram encriptadas de forma a garantir que não existia o acesso indevido por parte de utilizadores sem permissões. Assim, foi criado um método que devolve os *bytes* da *string* enviada em texto, fazendo depois a *hash* SHA256 do *array* desses *bytes*. No fim, este conjunto de *bytes* é convertido para uma *string*, sendo esta o texto guardado na base de dados. Ao iniciar sessão, é realizado este processo para a palavra-passe introduzida e verificado se os textos são iguais.

Além destas cifragens, para garantir que duas palavras-passe não ficam guardadas com a mesma *hash* na base de dados, é concatenada à palavra inserida pelo utilizador, um elemento único em cada utilizador. Também a chave API utilizada para os pedidos *Ajax* é gerada de forma idêntica, através da concatenação de vários elementos. A cada início de sessão é gerada uma chave nova, prevenindo que o mesmo utilizador tenha múltiplas sessões iniciadas.

Ao longo do projeto, houve sempre a necessidade de que todo o código estivesse bem organizado, otimizado e que não existissem métodos iguais em diferentes pontos. Houve também a preocupação em organizar os métodos de acordo com o grupo a que pertenciam, por exemplo, o controlador das caixas tem apenas os métodos responsáveis por alterações nas caixas; enquanto que o controlador qualidade apresenta métodos que apenas estão disponíveis no ecrã "Aprovação de Paletes".

Para a criação das páginas, foi sempre dada prioridade à utilização do *Bootstrap*, uma vez que permite criar páginas com desenho responsivo sem ser necessário criar múltiplas páginas de estilo. Em relação ao desenho responsivo foram utilizados, maioritariamente, os *breakpoints small* (col-sm), *medium* (col-md) e *large* (col-lg), assim como o *breakpoint* que os abrange a todos (col). Houve também o cuidado de fazer a construção do HTML de acordo com as regras definidas pela W3C. Foi sempre importante manter a informação coerente entre ecrãs, pelo que são utilizadas vistas parciais para que a apresentação da informação seja coerente entre ecrãs. Isto acontece, por exemplo, nos detalhes da palete (ver Figura 5.16 e Figura 5.20), em que foi criada uma única vista para apresentar a mesma informação nos ecrãs "Aprovação de paletes" e "Gestão de armazém".

6. Testes

Neste capítulo é apresentado o conjunto de testes realizados para ambos os projetos. É realizada uma apresentação geral dos testes realizados, sendo de seguida discutidos os testes efetivamente executados em cada projeto.

Em ambos os projetos, os testes possíveis ficaram limitados a testes de usabilidade, de funcionalidade e de integração.

Os testes de usabilidade focam-se na avaliação de tarefas predefinidas e reais que os utilizadores do sistema realizam. Com o evoluir das tecnologias, estes testes têm sido utilizados, em conjunto com outras ferramentas, para a avaliação da experiência do utilizador com o sistema em desenvolvimento [49].

Não foram aplicados testes de usabilidade criados especificamente para os projetos, mas houve a realização de testes por parte dos clientes, como forma de perceber como funcionavam as aplicações e o que poderia ser feito para melhorar a sua utilização. Durante o processo de desenvolvimento, os clientes foram comunicando o que pretendiam que fosse alterado, sendo realizado o pedido quando possível.

No caso do projeto *PHC Web*, houve algumas indicações do cliente que não foram possíveis de desenvolver devido às limitações da aplicação, como por exemplo alterações nos campos de pesquisa. No entanto, houve outras que foi possível de realizar, como as apresentadas na Tabela 6.1. No caso das últimas duas linhas da tabela, o desenvolvimento teve de ser adaptado ao que era possível realizar através da *framework* disponibilizada pela *PHC*.

Tabela 6.1 – Alterações pedidas pelo cliente do projeto PHC Web

Como o cliente testou	Alteração pedida pelo cliente
Faturação de encomendas de cliente sempre para a séria fatura	Questionar ao utilizador qual a série a utilizar (fatura ou fatura/recibo). Caso houvesse alguma fatura em memória, perguntar ao utilizador se pretendia apagar ou cancelar a ação
Análises	Foi pedido para adicionar filtros ou preenchê-los de acordo com a loja com sessão iniciada
Envio por email de documentos	Criar a opção para enviar para o cliente, sem ser necessário o preenchimento de dados. O email é enviado de acordo com a informação que existe na ficha do cliente que está no documento

Impressão/download de documentos	Foi pedido para simplificar o processo padrão da *PHC Web*
Cliente genérico	Criação de uma janela *pop-up* que pede ao utilizador o nome, o número de contribuinte e a morada do cliente

Já no caso do projeto *aWMS*, as indicações que foram pedidas, mas que não foram realizadas, foi por não fazer sentido o seu desenvolvimento (como a criação de paletes vazias, que iria desorganizar o esquema de paletização criado no *PHC*). Contudo, houve alguns pedidos que eram possíveis de desenvolver, como apresentados na Tabela 6.2. A tabela não apresenta todas as alterações pedidas que foram realizadas, apenas as mais importantes.

Tabela 6.2 – Alterações pedidas pelo cliente do projeto aWMS

Como o cliente testou	**Alteração pedida pelo cliente**
Versão inicial com múltiplos templates para apresentar a mesma informação	Que as paletes tivessem todas o mesmo formato e informação, independentemente da fase do processo onde se encontram
Os filtros estavam associados a um form, com autocomplete ativado	Foi pedido para que a aplicação não guardasse informação (apenas foi mantido o *autocomplete* na página de *login*)
Paletes condicionadas não podiam ser alteradas de local	Passa a ser possível alterar o local de qualquer palete que esteja produzida e aprovada
Não havia a indicação de totais de paletes por ecrã	Disponibilizar o total de paletes condicionadas em cada ecrã e número total de paletes por aprovar no ecrã de aprovação
Local marcado com cor apenas a indicar que estava ocupado	O local deve aparecer com a cor da palete que lá se encontra. Em caso de múltiplas paletes, a ordem é: condicionada, cliente e *stock*
O motivo de condicionamento seria escrito numa caixa de texto livre, com preenchimento obrigatório	Os motivos passam a ser estereotipados. Deve ser escolhido um motivo da lista e a caixa de texto livre para observações passa a ser facultativa
Ao aplicar um filtro, os locais que continham as paletes filtradas apareciam a uma cor diferente	Foi pedido para, ao aplicar os filtros, apenas ficassem coloridos os locais que tivessem as paletes que resultaram da pesquisa
Listagem das paletes apenas com o ID da palete, resumo da palete e local onde está localizada	A listagem das paletes passa a ter também a informação do cliente, dos artigos existentes na palete e o total de artigos

Os testes de integração servem para testar o envio e receção de informação entre vários componentes/módulos de um sistema, para garantir que não ocorre nenhum problema. São aplicados através do uso de cenários, criados a partir dos requisitos do sistema [50].

No caso do projeto *PHC Web*, os testes de integração não são muito relevantes, uma vez que os desenvolvimentos que foram realizados não são extensos e são pequenos ajustes aos já existentes. No entanto, quando existe informação a ser passada entre duas "funcionalidades" distintas (dossiers e faturas), há a necessidade de garantir que os comportamentos das faturas não são perdidos.

Já no projeto *aWMS*, os testes de integração serviram para garantir a comunicação com a API criada e que a informação necessária para criar documentos no *PHC* é guardada. Para a criação dos documentos, existe uma tabela intermédia para garantir que não existe informação perdida.

Os testes de funcionalidade permitem validar se as funcionalidades produzem o resultado esperado, ou seja, cada US é testada como sendo única. Quem realiza estes testes sabe a informação que tem de inserir e o resultado que deve visualizar, sem a necessidade de saber o que está a ser feito entre a entrada e a saída de informação [51]. A maioria dos testes realizados em ambos os projetos foram deste tipo, visto que após o desenvolvimento de uma funcionalidade, estas eram validadas para verificar se estavam de acordo com o esperado.

Durante o desenvolvimento do projeto *PHC Web*, além da validação se o desenvolvimento estava de acordo com o pretendido, houve também um apoio por parte do cliente, em indicar se era aquele o resultado esperado. Durante o projeto, o cliente foi sempre realizando alguns testes e dando *feedback* do que estava incorreto. Não sendo possível o desenvolvimento de testes funcionais para o código, foi de extrema relevância a realização de testes por parte do cliente, durante todo o processo. Este tipo de *feedback* permitiu descobrir e corrigir situações específicas, antes de a aplicação entrar em fase de produção e começar a ser usada numa base diária. A Tabela 6.3 apresenta alguns dos problemas identificados pela equipa responsável pelo projeto e/ou pelo cliente. Para cada problema é indicada a correção que foi necessário concretizar.

Tabela 6.3 – Problemas identificados durante os testes de usabilidade do projeto PHC Web

Problema identificado	**Correção**
Campos das linhas por preencher/incorretamente preenchidos	Alteração na forma como eram preenchidas as colunas
Ecrãs inacessíveis	Retificação das permissões
Informação em falta	Tornar campos obrigatórios
Gravação de documentos sem abertura de caixa	Retificação da *query* que ia buscar a abertura caixa para a loja em questão

Já com o projeto em produção, foi necessário voltar a realizar testes para garantir que o *upgrade* da versão do *PHC* não iria afetar nenhum desenvolvimento realizado previamente ou qualquer outro comportamento *standard* do *software*. Neste caso, os testes foram apenas realizados internamente, uma vez que havia a necessidade de utilizar uma base de dados para testes e garantir que nenhum documento produzido seria considerado inválido.

No que se refere ao projeto *aWMS*, durante o seu desenvolvimento, foram realizadas algumas reuniões com o cliente para apresentação das funcionalidades desenvolvidas até ao momento, assim como foi disponibilizada uma versão para o cliente testar. Novamente, a realização de testes pelo cliente e o seu *feedback* foi de extrema importância, uma vez que este projeto tinha um elevado número de requisitos que tinham de ser cumpridos, para ir de encontro às regras de negócio do cliente.

Sendo apenas possível a realização de testes sobre o uso da aplicação, o *feedback* do cliente foi crucial para perceber quais os requisitos em falta ou quais as funcionalidades que não estavam totalmente de acordo com o esperado. A Tabela 6.4 tem descritos alguns dos problemas identificados durante o processo de testes, quer pelo cliente, quer internamente pelos colegas envolvidos no projeto. Para cada situação está descrita a correção efetuada.

Tabela 6.4 – Problemas identificados durante os testes de usabilidade do projeto aWMS

Problema identificado	**Correção**
Disposição/usabilidade em dispositivo móvel	Ajuste dos *breakpoints* do *Bootstrap* e/ou do comportamento
Aplicação de múltiplos filtros	Redefinida a forma como a *query* era construída de acordo com os filtros utilizados
Atualização de ecrãs automaticamente	Corrigidas as funções que atualizam os ecrãs
Ao limpar os filtros das tabelas, caso tivesse linhas-filhos, estes ficavam expandidos	Após a aplicação dos filtros (estejam os campos vazios ou não), é forçada a ocultação das linhas-filhos
Realocar duas paletes em "simultâneo"	Foi adicionada uma validação para verificar se a palete ainda se encontra no local de origem ou se a localização já foi alterada
Não estava a ser possível alterar a quantidade máxima da caixa para a quantidade existente (se a caixa tivesse 30 peças, não deixava colocar a quantidade máxima como 30)	Foi alterada a validação para permitir colocar a quantidade máxima igual à quantidade existente na caixa

7. Conclusão

Este documento apresenta uma descrição do trabalho desenvolvido durante o estágio realizado na Arentia. O estágio foi multifacetado, não tendo sido focado apenas num projeto e permitindo expandir os conhecimentos na área da Gestão.

Pode-se afirmar que os principais objetivos do estágio foram atingidos, na medida em que o primeiro projeto iniciado foi colocado em produção com sucesso e o segundo projeto arrancou para a fase de testes massiva pouco depois do término do estágio. Já havendo enquadramento na empresa, não houve dificuldades na adaptação, sendo apenas necessário focar nas tarefas a desenvolver.

O primeiro projeto – *PHC Web* – foi implementado sem grandes dificuldades, havendo um *feedback* quer do cliente, quer dos colegas envolvidos no projeto, do que era necessário melhorar ou corrigir. Sendo uma plataforma já existente, os desenvolvimentos a realizar foram de dimensões reduzidas, focando-se em ajustes para obedecer às regras de negócio definidas pelo cliente. Os requisitos foram, na sua maioria, cumpridos. Os requisitos que não foram cumpridos na sua plenitude foi devido às limitações da plataforma de desenvolvimento da aplicação.

O segundo projeto – *aWMS* – foi também implementado sem grandes dificuldades, sendo aqui fundamental o apoio dos colegas envolvidos no projeto para perceber o modelo de negócio do cliente e ajudar a testar a aplicação. Era uma aplicação que já estava começada, pelo que a maior dificuldade foi perceber o que estava feito e como estava feito. No entanto, podemos dizer que os requisitos, que foram sendo definidos ao longo dos meses, foram sempre sendo cumpridos. Neste projeto, também se contou com a ajuda do cliente para ir realizando alguns testes e dando indicações do que podia ser melhorado ou do que tinha de ser alterado.

É de salientar também a importância de muitas unidades curriculares lecionadas no mestrado para o sucesso do estágio realizado. Uma das unidades curriculares que facilitou a integração nos projetos e na empresa foi “Sistemas de Informação Empresarias”, da licenciatura em Engenharia Informática, uma vez que permitiu ter conhecimento da arquitetura das aplicações empresarias, assim como a desenvolver o pensamento para o delineamento das regras de negócio. Também as unidades curriculares de “Qualidade de Software” e “Gestão

de Projetos Informáticos", lecionadas durante o mestrado em Computação Móvel, se revelaram importantes para uma melhor gestão das tarefas a realizar durante o decorrer de ambos os projetos e para o reconhecimento da importância dos testes.

Mesmo não tendo sido possível o desenvolvimento de testes específicos para o código, foi possível notar a importância de haver múltiplos elementos a testar uma mesma funcionalidade (especialmente elementos sem conhecimento do "*happy path*").

O balanço final do estágio é extremamente positivo, tendo sido ultrapassados todos os desafios propostos com sucesso. Após a conclusão do estágio, foi realizada uma proposta para permanecer na empresa, tendo esta sido aceite. Durante o estágio, houve o enquadramento numa das equipas de *PHC*, com o objetivo de realizar apenas desenvolvimento para o *PHC Web*, tendo depois sido enquadrada na equipa de *Web*, que desenvolve aplicações externas para completar a informação do *PHC*, podendo-se concluir que houve reconhecimento do trabalho realizado.

Como trabalho futuro, não é esperado o desenvolvimento de novas funcionalidades em nenhum dos projetos. No caso do projeto *PHC Web*, este apenas terá associada a manutenção da versão, sendo realizados *upgrades* devido a questões legais. Já o projeto *aWMS* teve todas as funcionalidades desenvolvidas, estando apenas previsto a correção de *bugs* que sejam identificados no decorrer dos testes.

Bibliografia

[1] "Quem Somos," Arentia, 2020. [Online]. Available: https://www.arentia.pt/pt/empresa/quem-somos. [Accessed: 29-Dec-2020]

[2] "Software Gestão," Arentia, 2020. [Online]. Available: https://www.arentia.pt/pt/solucoes/software-de-gestao. [Accessed: 29-Dec-2020]

[3] L. F. L. Sismeiro, "Projectos de consultoria em SAP e tecnologias microsoft: análise e desenvolvimento de soluções de software à medida," Politécnico de Leiria - ESTG, Leiria, 2014 [Online]. Available: http://hdl.handle.net/10400.8/2123. [Accessed: 20-Jan-2021]

[4] C. I. P. Gomes, "A importância dos sistemas ERP (Enterprise Resource Planning) na área financeira das organizações," Mestrado, ESTG - IPLeiria, Leiria, 2020 [Online]. Available: https://iconline.ipleiria.pt/handle/10400.8/5395. [Accessed: 18-Apr-2021]

[5] A. I. M. Duarte, "Sistemas de informação: ciclo de vida e análise do seu sucesso nas organizações," Dissertação de mestrado, Iscte - Instituto Universitário de Lisboa, Lisboa, 2012 [Online]. Available: https://repositorio.iscte-iul.pt/handle/10071/5373. [Accessed: 03-May-2021]

[6] Ij. Oudshoorn, "Development of Packaged Software." Vrije Universiteit Amsterdam, 2004 [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.876&rep=rep1&type=pdf. [Accessed: 20-Jan-2021]

[7] M. Tsunoda, A. Monden, K. Matsumoto, S. Ohiwa, and T. Oshino, "Analysis of Attributes Relating to Custom Software Price," in 2012 Fourth International Workshop on Empirical Software Engineering in Practice, 2012, pp. 16–22, doi: 10.1109/IWESEP.2012.19.

[8] E. Carmel, "A discussion of special characteristics for software package development life cycle models," ACM SIGSOFT Softw. Eng. Notes, vol. 18, no. 2, pp. 23–24, Apr. 1993, doi: 10.1145/159420.155832. [Online]. Available: https://doi.org/10.1145/159420.155832. [Accessed: 18-Apr-2021]

[9] S. Sawyer, "Packaged software: implications of the differences from custom approaches to software development," Eur. J. Inf. Syst., vol. 9, pp. 47–58, 2000, doi: 10.1057/palgrave.ejis.3000345.

[10] H. Tian and K. Zhang, “A Review on Analysis/Design Methods and Life-Cycle Models for System Development,” IFAC Proc. Vol., vol. 31, no. 20, pp. 965–970, Jul. 1998, doi: 10.1016/S1474-6670(17)41923-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017419239. [Accessed: 21-Apr-2021]

[11] J. Navascués, I. Ramos, and M. Toro, “A Hybrid Model for Dynamic Simulation of Custom Software Projects in a Multiproject Environment,” in Trustworthy Software Development Processes, Berlin, Heidelberg, 2009, pp. 173–185, doi: 10.1007/978-3-642-01680-6_17.

[12] N. B. Ruparelia, “Software development lifecycle models,” ACM SIGSOFT Softw. Eng. Notes, vol. 35, no. 3, pp. 8–13, May 2010, doi: 10.1145/1764810.1764814. [Online]. Available: https://doi.org/10.1145/1764810.1764814. [Accessed: 21-Apr-2021]

[13] “Modernization - Clearing a pathway to success,” The Standish Group International, Inc, 2010 [Online]. Available: https://www.standishgroup.com/sample_research_files/Modernization.pdf. [Accessed: 03-May-2021]

[14] H. Francisco, “Differences, advantages and disadvantages between in-house development IT systems and industry standard ERP system.” 2012 [Online]. Available: https://www.academia.edu/4865003/Differences_advantages_and_disadvantages_between_in_house_development_IT_systems_and_industry_standard_ERP_system. [Accessed: 04-May-2021]

[15] L. Xu and S. Brinkkemper, “Concepts of product software,” Eur. J. Inf. Syst., vol. 16, no. 5, pp. 531–541, Oct. 2007, doi: 10.1057/palgrave.ejis.3000703. [Online]. Available: https://orsociety.tandfonline.com/doi/full/10.1057/palgrave.ejis.3000703. [Accessed: 04-May-2021]

[16] G. Kulkarni, P. Chavan, H. Bankar, K. Koli, and V. Waykule, “A new approach to software as service cloud,” in 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), 2012, pp. 196–199, doi: 10.1109/TSSA.2012.6366050.

[17] A. A. Al-Ghofaili and M. A. Al-Mashari, “ERP system adoption traditional ERP systems vs. cloud-based ERP systems,” in Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014), 2014, pp. 135–139, doi: 10.1109/INTECH.2014.6927770.

[18] I. Orosz, A. Selmeci, and T. Orosz, “Software as a Service operation model in cloud based ERP systems,” in 2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2019, pp. 345–354, doi: 10.1109/SAMI.2019.8782739.

[19] S. M. Mohammad, “DevOps Automation and Agile Methodology,” Int. J. Creat. Res. Thoughts IJCRT, vol. 5, no. 3, pp. 946–949, Aug. 2017 [Online]. Available: https://ssrn.com/abstract=3655581. [Accessed: 18-May-2021]

[20] K. Beck et al., “Manifesto for Agile Software Development,” Manifesto for Agile Software Development, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed: 18-May-2021]

[21] A. Srivastava, S. Bhardwaj, and S. Saraswat, “SCRUM model for agile methodology,” in 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 864–869, doi: 10.1109/CCAA.2017.8229928.

[22] Scrum.org, “What is Scrum?,” Scrum.org, 2021. [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Accessed: 18-May-2021]

[23] “PHC | Sobre nós,” PHC Software. [Online]. Available: https://www.phcsoftware.com/sobre-nos/. [Accessed: 01-Feb-2021]

[24] “PHC CS | O software de gestão que o ajuda a atingir a excelência,” PHC Software. [Online]. Available: https://www.phcsoftware.com/solucoes/produtos/phc-cs/. [Accessed: 01-Feb-2021]

[25] “Módulos PHC CS web e desktop,” PHC Software. [Online]. Available: https://www.phcsoftware.com/modulos/. [Accessed: 01-Feb-2021]

[26] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, Patterns of Enterprise Application Architecture. Addison Wesley, 2020.

[27] P. Christensson, “IIS Definition,” TechTerms, 11-Dec-2013. [Online]. Available: https://techterms.com/definition/iis. [Accessed: 02-Feb-2021]

[28] “What is .NET? An open-source developer platform.,” Microsoft. [Online]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet. [Accessed: 02-Feb-2021]

[29] “What is SQL Server? - Definition from Techopedia,” Techopedia.com. [Online]. Available: http://www.techopedia.com/definition/1243/sql-server. [Accessed: 02-Feb-2021]

[30] "What is a relational database?," Oracle Portugal, 2021. [Online]. Available: https://www.oracle.com/pt/database/what-is-a-relational-database/. [Accessed: 09-Feb-2021]

[31] "CPS - Consultores de Informática, S.A. | Leiria - CPS - Consultores de Informática, S.A. | Leiria - Gesobra." [Online]. Available: http://www.cps-ci.com/index.php/pt/gesobra-pt. [Accessed: 24-Feb-2021]

[32] "Overview - ADO.NET," Microsoft | Docs, 30-Mar-2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview. [Accessed: 22-Mar-2021]

[33] "What is Entity Framework?," Entity Framework Tutorial, 2020. [Online]. Available: https://www.entityframeworktutorial.net/what-is-entityframework.aspx. [Accessed: 18-Mar-2021]

[34] C. Nagel, Professional C# 7 and .NET Core 2.0. John Wiley & Sons, 2018.

[35] J. Lerman, Programming Entity Framework: Building Data Centric Apps with the ADO.NET Entity Framework. O'Reilly Media, Inc., 2010.

[36] "Method-Based Query Syntax Examples: Projection - ADO.NET," Microsoft | Docs. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/method-based-query-syntax-examples-projection. [Accessed: 26-Jul-2021]

[37] "Entity Framework Architecture," Entity Framework Tutorial, 2020. [Online]. Available: https://www.entityframeworktutorial.net/EntityFramework-Architecture.aspx. [Accessed: 18-Mar-2021]

[38] J. Chadwick, Programming Razor: Tools for Templates in ASP.NET MVC or WebMatrix. O'Reilly Media, Inc., 2011.

[39] "JavaScript | MDN," MDN Web Docs, 21-Feb-2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed: 23-Mar-2021]

[40] A. Pereira and C. Poupa, Linguagens Web, 5a. Lisboa: Edições Sílabo, 2015.

[41] J. F.- js.foundation, "jQuery," jQuery, 2021. [Online]. Available: https://jquery.com/. [Accessed: 24-Mar-2021]

[42] J. F.- js.foundation, "jQuery UI," jQuery UI, 2021. [Online]. Available: https://jqueryui.com/. [Accessed: 24-Mar-2021]

[43] N. Payne, "Bootbox.js," Bootbox.js, 2019. [Online]. Available: http://bootboxjs.com/. [Accessed: 24-Mar-2021]

[44] SpryMedia Ltd, "Manual," DataTables. [Online]. Available: https://datatables.net/manual/. [Accessed: 20-Jun-2021]

[45] "Plan, code, collaborate, ship applications - Azure DevOps," Microsoft | Azure DevOps, 22-Jan-2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops. [Accessed: 08-May-2021]

[46] "Collaborate on code - Azure Repos," Microsoft | Azure DevOps, 01-Jun-2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos. [Accessed: 08-May-2021]

[47] "Overview of Visual Studio," Microsoft | Visual Studio, 19-Mar-2019. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide. [Accessed: 08-May-2021]

[48] M. O. contributors Jacob Thornton, and Bootstrap, "Overview," Layout. [Online]. Available: https://getbootstrap.com/docs/4.6/layout/overview/. [Accessed: 22-Sep-2021]

[49] C. M. Barnum, Usability Testing Essentials: Ready, Set ...Test! Morgan Kaufmann, 2020.

[50] M. A. S. Brito, S. R. S. Souza, and P. S. L. Souza, "Integration testing for robotic systems," Softw. Qual. J., Nov. 2020, doi: 10.1007/s11219-020-09535-w. [Online]. Available: https://doi.org/10.1007/s11219-020-09535-w. [Accessed: 10-Feb-2021]

[51] W. E. Lewis, Software Testing and Continuous Quality Improvement. CRC Press, 2017.

Anexo A

Na Tabela A.1 são apresentados os *user stories* dos processos de negócio relativos ao projeto *PHC Web*.

Tabela A.1 – *User Stories* dos processos de negócio (Projeto *PHC Web*)

US	Ecrã	Descrição
1. Gestão de Clientes, Artigos e Logística		
US1.1	Logística	Enquanto loja tenho de ter uma área associada.
US1.2	Logística	Enquanto loja tenho de ter um armazém associado. Deve ainda existir um armazém geral para consignações/amostras (produtos que saíram de *stock*, mas que ainda não foram faturados).
US1.3	Clientes	Enquanto loja quero conseguir consultar a lista de clientes associados à minha área.
US1.4	Clientes	Enquanto loja devo conseguir consultar nome, número, país e número de contribuinte, morada, telefone, telemóvel, email, tabela de preços, zona, nome do vendedor, área e saldo em conta-corrente da ficha de cliente selecionado.
US1.5	Clientes	Enquanto loja devo ter disponível a opção para consultar os artigos vendidos ao cliente (análise), com identificação do documento, do cliente selecionado.
US1.6	Clientes	Enquanto loja devo ter disponível a opção para imprimir a conta-corrente completa do cliente selecionado.
US1.7	Clientes	Enquanto loja devo ter disponível a opção para imprimir a conta-corrente dos movimentos por regularizar do cliente selecionado.
US1.8	Clientes	Enquanto loja não posso introduzir novos clientes, nem alterar ou eliminar existentes.
US1.9	Artigos	Enquanto loja devo conseguir consultar a lista de artigos.
US1.10	Artigos	Enquanto loja devo conseguir consultar a referência, designação, preço, *stock* total, família do artigo e imagem (se disponível) da ficha do artigo selecionado.
US1.11	Artigos	Enquanto loja só posso ver o preço do artigo utilizado na minha área.
US1.12	Artigos	Enquanto loja devo conseguir consultar o *stock* existente do artigo selecionado em qualquer um dos armazéns.
US1.13	Artigos	Enquanto loja devo ter disponível opções para consultar a ficha técnica e a ficha de segurança do artigo escolhido.
US1.14	Artigos	Enquanto loja devo ter disponível a opção para imprimir a etiqueta do artigo selecionado, com descrição, referência e código de barras.
US1.15	Artigos	Enquanto loja não posso introduzir novos artigos, nem alterar ou eliminar existentes.
2. Abertura e Fecho de caixa		

US2.1	Geral	Enquanto loja só devo realizar a abertura ou fecho de caixa para a minha área. O campo da área deve aparecer preenchido e apenas de leitura.
US2.2	Abertura Caixa	Enquanto loja devo conseguir realizar a abertura de caixa para o próprio dia.
US2.3	Abertura Caixa	Enquanto loja ao realizar a abertura de caixa, devo ver o campo "Loja" preenchido com a minha área, o campo "Utilizador" deve apresentar os utilizadores da minha área e o campo "Valor" deve ser de preenchimento livre.
US2.4	Abertura Caixa	Enquanto loja não devo conseguir introduzir nova abertura de caixa se não existir um fecho de caixa para o último dia de trabalho para a minha área.
US2.5	Fecho Caixa	Enquanto loja devo conseguir realizar o fecho de caixa.
US2.6	Fecho Caixa	Enquanto loja ao realizar o fecho de caixa, devo ver o campo "Loja" preenchido com a minha área, o campo "Utilizador" deve apresentar os utilizadores da minha área e o campo "Valor" deve ser de preenchimento livre e não pode ter o valor zero.
US2.7	Fecho Caixa	Enquanto loja não devo conseguir introduzir novo fecho de caixa sem existir uma abertura de caixa no último dia de trabalho para a minha área.
US2.8	Ao iniciar sessão	Enquanto loja ao iniciar sessão, caso haja um fecho de caixa para o último dia de trabalho, deve aparecer disponível uma janela para realizar a abertura de caixa.
US2.9	Ao iniciar sessão	Enquanto loja ao iniciar sessão, caso não haja um fecho de caixa para o último dia de trabalho, deve aparecer disponível uma janela com essa indicação e a possibilidade de navegar para o ecrã de fecho de caixa.
3. Venda		
US3.1	Faturação	Enquanto loja devo conseguir introduzir novos documentos de faturação e alterar ou eliminar documentos que estejam em rascunho.
US3.2	Faturação	Enquanto loja só posso faturar para clientes que façam parte da minha área.
US3.3	Faturação	Enquanto loja não posso alterar a área do documento de faturação que estou a criar.
US3.4	Faturação	Enquanto loja não posso emitir documentos de faturação sem abertura de caixa para o próprio dia.
US3.5	Faturação	Enquanto loja ao criar uma fatura, se não existir abertura de caixa para o próprio dia, deverá aparecer uma janela para realizar a abertura de caixa.
US3.6	Faturação	Enquanto loja ao criar nova fatura, deve ser preenchido, por defeito, os campos referentes ao cliente, com o cliente genérico criado para a minha área.
US3.7	Faturação	Enquanto loja devo conseguir editar os dados do cliente genérico, sem necessidade de criar novo cliente (campos a serem preenchidos: número de contribuinte, país do contribuinte, nome e morada).

US3.8	Faturação	Enquanto loja ao adicionar linhas na fatura, deve ser preenchido o armazém de acordo com a minha área.
US3.9	Faturação	Enquanto loja ao faturar um dossier "Amostra", o armazém das linhas deve ser o armazém das amostras.
US3.10	Faturação	Enquanto loja não posso alterar a tabela de IVA dos artigos adicionados às faturas.
US3.11	Faturação	Enquanto loja devo conseguir consultar o *stock* de um artigo adicionado à fatura, em todos os armazéns, sem sair do ecrã de faturação.
US3.12	Faturação	Enquanto loja ao criar uma fatura/recibo ou uma nota de crédito FR, devo preencher obrigatoriamente o campo "Método de pagamento", escolhendo entre as opções "CAIXA" e "MB".
US3.13	Faturação	Enquanto loja devo conseguir criar uma nota de crédito partindo de uma fatura existente. Devem ser copiados todos os dados da fatura e adicionada uma linha nos artigos com a designação da fatura copiada.
US3.14	Recibos	Enquanto loja devo conseguir introduzir novos recibos e alterar ou eliminar recibos que estejam por processar.
US3.15	Recibos	Enquanto loja não posso emitir recibos sem abertura de caixa para o próprio dia.
US3.16	Recibos	Enquanto loja só posso criar recibos para clientes que façam parte da minha área.
US3.17	Recibos	Enquanto loja não posso alterar a área do recibo que estou a criar.
US3.18	Recibos	Enquanto loja ao criar um recibo, devo preencher obrigatoriamente o campo "Método de pagamento", escolhendo entre as opções "CAIXA" e "MB".
US3.19	Recibos	Enquanto loja ao gravar um recibo, este deve ser processado automaticamente.
4. Gestão interna		
US4.1	Dossiers	Enquanto loja devo conseguir introduzir novos dossiers e alterar ou eliminar dossiers que não estejam fechados.
US4.2	Dossiers	Enquanto loja só posso criar dossiers para clientes que façam parte da minha área.
US4.3	Dossiers	Enquanto loja não posso alterar a área do dossier que estou a criar.
US4.4	Dossiers	Enquanto loja ao criar dossiers "Envio para loja" e "Receção de loja", o cliente por defeito será X (nome do cliente do projeto).
US4.5	Dossiers	Enquanto loja devo conseguir faturar um dossier "Encomenda de cliente". Devem ser copiados todos os dados da encomenda e adicionada uma linha nos artigos com a designação do dossier copiado.
US4.6	Dossiers	Enquanto loja devo conseguir faturar um dossier "Amostra". Devem ser copiados todos os dados do dossier e adicionada uma linha nos artigos com a designação do dossier copiado.
US4.7	Dossiers	Enquanto loja devo conseguir criar um dossier "Devolução de Amostra", partindo de um dossier "Amostra". Devem ser copiados todos

		os dados do dossier e adicionada uma linha nos artigos com a designação do dossier copiado.
US4.8	Dossiers	O dossier "Amostra" será configurado com guia de transporte e comunicado à Autoridade Tributária (AT), através do *webservice* da AT.
US4.9	Guias Transporte	As guias de transporte têm de ter disponíveis e visíveis os campos "Morada de carga", "Morada de descarga" e "Código de identificação atribuído pela AT".
5. Análises		
US5.1	Geral	Enquanto loja as análises devem apresentar apenas a informação relativa à minha área. Caso exista o filtro "Loja", este deve ser preenchido com a minha área.
US5.2	Análise	Enquanto loja devo conseguir faturar uma encomenda de cliente através da análise "Encomendas de cliente".
US5.3	Análise	A análise "Transferências de loja" deve ter os filtros "Loja de origem" e "Loja de destino". O filtro "Loja de origem" deve apresentar todas as áreas e o filtro "Loja de destino" deve ser preenchido com a minha área.
US5.4	Análise	Enquanto loja devo conseguir copiar linhas da análise "Transferência de loja", para criar um novo dossier "Receção de loja".
US5.5	Análise	Enquanto loja devo conseguir filtrar a análise "Regularizar amostras" através dos filtros "Loja" e "Nome do cliente".
US5.6	Análise	Enquanto loja devo conseguir selecionar linhas da análise "Regularizar amostras", para criar um novo dossier "Devolução de amostrar" ou faturar os artigos selecionados.
US5.7	Análise	A análise "Desvios de caixa" deve ter os filtros "Loja", "Data de início" e "Data de fim".
US5.8	Análise	A análise "Relatório movimentos por conta" deve ter os filtros "Loja", "Data de início" e "Data de fim".
US5.9	Análise	A análise "Mapa Vendas" deve ter os filtros "Loja" e "Mês", a fim de visualizar as vendas para uma dada loja num dado mês.
US5.10	Apresentação	Enquanto loja quero visualizar as análises "Encomenda de cliente", "Transferências de loja" e "Regularizar amostras" na página inicial (em formato *snapshot*).
US5.11	Apresentação	Enquanto loja quero aceder as restantes análises através de opções no menu.

Anexo B

Na Tabela B.1 são apresentados os *user stories* dos processos de negócio relativos ao projeto *aWMS*.

Tabela B.1 – *User Stories* dos processos de negócio (Projeto *aWMS*)

US	**Tipo**	**Descrição**
	1. Utilizadores	
US1.1	Iniciar sessão	Enquanto utilizador só posso aceder à aplicação depois de iniciar sessão.
US1.2	Iniciar sessão	Enquanto utilizador só posso aceder se tiver permissão para aceder à aplicação.
US1.3	Iniciar sessão	Enquanto utilizador devo preencher os campos utilizador e palavra-passe para iniciar sessão.
US1.4	Sessão iniciada	Enquanto utilizador devo ser redirecionado de acordo com as permissões definidas.
US1.5	Permissões	Enquanto utilizador só devo realizar ações ou aceder a ecrãs para os quais tenha permissões.
US1.6	Alterar palavra-passe	Enquanto utilizador devo ter disponível uma opção para alterar a minha palavra-passe.
US1.7	Alterar palavra-passe	Enquanto utilizador devo preencher os campos "Nova palavra-passe" e "Confirmar palavra-passe" para concluir o processo de alteração de palavra-passe.
	2. Armazéns	
US2.1	Selecionar armazém	Enquanto utilizador devo conseguir alterar o armazém selecionado.
	3. Aprovação de paletes	
US3.1	Paletes por aprovar	Enquanto utilizador devo visualizar todas as paletes produzidas e por aprovar ou condicionadas no ecrã "Aprovação de paletes".
US3.2	Paletes por aprovar	Enquanto utilizador devo ver o código e a designação da palete, o nome abreviado do cliente, as referências e designações dos artigos, o número da encomenda, os alertas associados à palete e o estado (condicionada ou não).
US3.3	Filtragem	Enquanto utilizador devo ter disponíveis campos para filtrar as paletes apresentadas.
US3.4	Consultar paletes	Enquanto utilizador, ao clicar numa palete, devo visualizar os detalhes da mesma.
US3.5	Aprovar	Enquanto utilizador devo consultar uma palete para a aprovar. Após clicar na opção "Aprovar", devo confirmar a ação para esta ser concluída.
US3.6	Corrigir quantidades	Enquanto utilizador devo conseguir corrigir a quantidade máxima das caixas de uma palete.

US3.7	Corrigir quantidades	Enquanto utilizador só devo ter disponível a opção para corrigir as quantidades se a caixa não estiver completa.
US3.8	Corrigir quantidades	Enquanto utilizador devo clicar na opção "Corrigir Qtt". Após a indicação da nova quantidade, devo confirmar a ação para esta ser finalizada.
	4. Gestão de armazéns	
US4.1	Armazém virtual	Enquanto utilizador devo conseguir visualizar o armazém virtual no ecrã "Gestão de armazém".
US4.2	Armazém virtual	Enquanto utilizador devo conseguir identificar quais as posições que estão livres e quantas paletes existem nas posições ocupadas.
US4.3	Listagem de paletes	Enquanto utilizador devo conseguir visualizar a lista de paletes presentes numa dada posição.
US4.4	Listagem de paletes	Enquanto utilizador, ao entrar no ecrã, devo visualizar a lista de paletes que estão por arrumar.
US4.5	Filtragem	Enquanto utilizador devo conseguir filtrar as paletes. Caso a palete esteja arrumada, devo conseguir identificar o local visualmente.
US4.6	Arrumar paletes	Enquanto utilizador só devo arrumar paletes que estejam produzidas e aprovadas ou condicionadas (caso já tivessem sido aprovadas antes de condicionadas).
US4.7	Consultar palete	Enquanto utilizador, ao clicar numa palete, devo visualizar os detalhes da mesma.
US4.8	Arrumar paletes	Enquanto utilizador devo conseguir arrastar uma palete para um certo local, passando a palete a estar alocada àquela posição.
	5. Detalhes da palete	
US5.1	Consultar palete	Enquanto utilizador quero ver os detalhes da palete escolhida no ecrã "Aprovação de paletes" e no ecrã "Gestão de armazém".
US5.2	Consultar palete	Enquanto utilizador devo ver a designação e o código da palete, as observações, o número de encomenda a que está associada, o cliente da encomenda e os alertas associados à palete.
US5.3	Consultar palete	Enquanto utilizador devo ver as quantidades por artigo que existem na palete.
US5.4	Consultar caixas	Enquanto utilizador quero ver os detalhes das caixas associadas à palete que estou a consultar.
US5.5	Consultar caixas	Enquanto utilizador devo ver a referência e designação do artigo da caixa, a quantidade em unidades, o lote e a versão de composição. Deve ainda estar disponível a data/hora e o operador responsável pela produção.
US5.6	Editar observações	Enquanto utilizador devo ter disponível a opção para editar as observações da palete.
	6. Transferências	
US6.1	Transf. de peças	Enquanto utilizador devo ter sempre disponível a opção para transferir peças.
US6.2	Transf. de peças	Enquanto utilizador devo preencher o número de peças a transferir e selecionar a opção "Transferir peças".

US6.3	Transf. de peças	Enquanto utilizador devo escolher a caixa para onde será transferido o número de peças selecionadas.
US6.4	Transf. de peças	Enquanto utilizador, em caso de caixa partilhada, para fazer uma transferência de peças, devo selecionar a caixa-filho e realizar o processo normal (US 6.2 e US 6.3).
US6.5	Transf. de caixas	Enquanto utilizador devo ter disponível a opção para transferir caixas.
US6.6	Transf. de caixas	Enquanto utilizador devo selecionar as caixas a transferir e selecionar a opção "Transferir caixas".
US6.7	Transf. de caixas	Enquanto utilizador devo escolher a palete para onde serão transferidas as caixas selecionadas.
US6.8	Transf. de caixas	Enquanto utilizador, em caso de caixa partilhada, para fazer uma transferência de caixa, devo selecionar a caixa-pai a transferir.
US6.9	Filtragem	Enquanto utilizador devo ter disponível campos para filtrar as paletes disponíveis para transferência, seja de caixas ou peças.
US6.10	Transf. de armazém	Enquanto utilizador devo ter disponível a opção para realizar transferências entre armazéns.
US6.11	Transf. de armazém	Enquanto utilizador, para transferir uma palete para outro armazém, devo clicar na opção "Trans. armazém".
US6.12	Transf. de armazém	Enquanto utilizador devo indica o armazém de destino e confirmar a ação para esta ser finalizada.
US6.13	Transf. de armazém	Enquanto utilizador devo conseguir ver a palete transferida no armazém de destino.
7. Verificar caixas		
US7.1	Verificar caixas	Enquanto utilizador devo conseguir verificar caixas.
US7.2	Verificar caixas	Enquanto utilizador devo ter sempre disponível a opção para verificar caixas.
US7.3	Verificar caixas	Enquanto utilizador devo selecionar todas as caixas verificadas e clicar na opção "Verificar caixas". Devo confirmar a ação para esta ser realizadas.
US7.4	Verificar caixas	Enquanto utilizador, em caso de caixa partilhada, para fazer a verificação de caixa, devo selecionar a caixa-pai verificada.
8. Condicionar paletes		
US8.1	Condicionar	Enquanto utilizador devo conseguir condicionar paletes.
US8.2	Condicionar	Enquanto utilizador devo ter sempre disponível a opção para condicionar paletes.
US8.3	Condicionar	Enquanto utilizador, para condicionar uma palete, devo clicar na opção "Condicionar".
US8.4	Condicionar	Enquanto utilizador devo indicar o motivo e confirmar a ação para esta ser finalizada. Devo ainda ter disponível um campo para observações.
US8.5	Palete condicionada	Enquanto utilizador posso realocar uma palete que esteja condicionada.

US8.6	Palete condicionada	Enquanto utilizador não posso transferir caixas ou peças, nem expedir uma palete condicionada.
9. Rejeição		
US9.1	Rejeitar	Enquanto utilizador só posso rejeitar peças ou caixas se estas pertencerem a uma palete por aprovar ou condicionada.
US9.2	Rejeitar peças	Enquanto utilizador devo ter disponível a opção para rejeitar peças.
US9.3	Rejeitar peças	Enquanto utilizador devo preencher o número de peças a rejeitar e selecionar a opção "Rejeitar peças".
US9.4	Rejeitar peças	Enquanto utilizador devo escolher um dos motivos apresentados para a rejeição e confirmar a ação para esta ser finalizada.
US9.5	Rejeitar peças	Enquanto utilizador, em caso de caixa partilhada, para fazer uma rejeição de peças, devo selecionar a caixa-filho e realizar o processo normal (US 9.3 e US 9.4).
US9.6	Rejeitar caixas	Enquanto utilizador devo ter disponível a opção para rejeitar caixas.
US9.7	Rejeitar caixas	Enquanto utilizador devo selecionar todas as caixas a rejeitar e clicar na opção "Rejeitar".
US9.8	Rejeitar caixas	Enquanto utilizador devo escolher um dos motivos apresentados para a rejeição e confirmar a ação para esta ser finalizada.
US9.9	Rejeitar caixas	Enquanto utilizador, em caso de caixa partilhada, para rejeitar uma caixa, devo selecionar a caixa-pai a rejeitar.

Anexo C

Na Tabela C.1 são apresentados os requisitos dos processos de negócio relativos ao projeto *aWMS*.

Tabela C.1 – Requisitos dos processos de negócio (Projeto *aWMS*)

RQ	Tipo	Descrição
	0. Geral	
RQ 0.1	Não funcional	O sistema deve ser uma aplicação *web*.
RQ 0.2	Não funcional	O sistema deve permitir a sua utilização em *tablets*.
RQ 0.3	Não funcional	O sistema deve ser fácil de usar.
RQ 0.4	Não funcional	O sistema deve apresentar uma baixa curva de aprendizagem.
RQ 0.5	Não funcional	O sistema deve permitir a utilização por vários utilizadores em simultâneo.
RQ 0.6	Não funcional	O sistema deve apresentar informação uniforme quando se consultam os mesmos objetos nos vários ecrãs.
RQ 0.7	Funcional	A página inicial deverá ser o ecrã de início de sessão.
RQ 0.8	Não funcional	A aplicação deve apresentar URL's uniformes, a exemplo de uma *single page application*.
RQ 0.9	Não funcional	A aplicação deverá ser atualizada a cada cinco minutos.
	1. Utilizadores	
RQ 1.0.1	Funcional	Os utilizadores são definidos na ficha de utilizador do *PHC*.
RQ 1.2.1	Funcional	Para ter acesso, o utilizador deve estar ativo, ter ativada a permissão para aceder à aplicação e a palavra-passe definida.
RQ 1.3.1	Funcional	Ao iniciar sessão, o utilizador deve ser informado se a palavra-passe estiver incorreta.
RQ 1.3.2	Funcional	Ao iniciar sessão, o utilizador deve ser informado se não tiver acesso à aplicação.
RQ 1.4.1	Funcional	Se um utilizador tiver acesso a múltiplos ecrãs, é redirecionado para o ecrã de menu.
RQ 1.4.2	Funcional	Se um utilizador só tiver acesso a um ecrã, será redirecionado para esse ecrã.
RQ 1.5.1	Funcional	As permissões para cada utilizador são estipuladas no *PHC*, através da lista disponível para tal.
RQ 1.5.2	Funcional	Um utilizador administrador tem acesso a todas as ações/ecrãs da aplicação.

RQ 1.6.1	Funcional	A opção para alterar a palavra-passe está disponível na barra de menu, ao clicar no nome do utilizador com sessão iniciada.
RQ 1.7.1	Funcional	A palavra-passe nova deve ser validada: não pode ser vazia, tem de ter mais de quatro caracteres e tem de ser igual nos dois campos.
RQ 1.7.2	Funcional	Não é permitido alterar a palavra-passe pela palavra-passe já existente.
RQ 1.7.3	Não funcional	A palavra-passe deve ser encriptada antes de ser gravada na base de dados.
RQ 1.7.4	Funcional	Se o campo da palavra-passe estiver vazio, deve ser forçada a alteração da palavra-passe (campo vazio significa que houve um *reset* à palavra-passe).
2. Armazéns		
RQ 2.0.1	Funcional	Os armazéns são configurados no *PHC*.
RQ 2.0.2	Funcional	Os armazéns disponíveis para a aplicação devem estar configurados para tal.
RQ 2.1.1	Funcional	Os armazéns disponíveis aparecem numa *dropdown* na barra de navegação, com o armazém atual selecionado.
RQ 2.1.2	Funcional	Por defeito, deve aparecer o primeiro armazém disponível selecionado.
3. Aprovação de paletes		
RQ 3.2.1	Funcional	As paletes devem ser apresentadas em forma de tabela.
RQ 3.2.2	Funcional	Caso uma palete esteja condicionada, deve aparecer a data/hora em que foi condicionada, o utilizador que fez a ação e onde está arrumada.
RQ 3.3.1	Funcional	Deve ser possível filtrar por código da palete, artigo, número de encomenda e número de cliente.
RQ 3.5.1	Funcional	Podem ser aprovadas paletes por aprovar ou condicionadas.
RQ 3.5.2	Funcional	As paletes são aprovadas individualmente.
RQ 3.5.3	Funcional	A opção para aprovar uma palete está disponível na consulta da palete e disponível através de um botão verde.
RQ 3.5.4	Funcional	Ao aprovar uma palete deve ficar registada a data/hora, assim como o utilizador que aprovou.
RQ 3.5.5	Funcional	As paletes só podem ser aprovadas se estiverem completas, ou seja, se não existirem caixas e/ou peças rejeitadas.
RQ 3.5.6	Funcional	Se uma palete estiver condicionada e for aprovada, ela deixa de estar condicionada. Deve ser atualizada a data/hora e o utilizador que aprovou.
RQ 3.7.1	Funcional	A opção para corrigir a quantidade máxima está disponível na consulta dos detalhes da caixa e através de um botão azul, à esquerda.
RQ 3.7.2	Funcional	A opção para corrigir quantidades fica disponível desde que a caixa não esteja completa (tenha sido por rejeição ou transferência de peças).
RQ 3.8.1	Funcional	Ao corrigir o máximo de uma caixa deve ficar registada a data/hora, assim como o utilizador que fez a correção.
RQ 3.8.2	Funcional	Ao corrigir a quantidade máxima de uma caixa tem de ser indicada a nova quantidade pelo utilizador.

RQ 3.8.3	Funcional	A nova quantidade tem de ser validada: é um campo obrigatório, do tipo numérico e o valor não pode ser superior à quantidade de peças existentes na caixa.
RQ 3.8.4	Funcional	As caixas que fiquem com o valor máximo igual a zero devem ser colocadas como inativas (deixam de existir, passam a ser caixas virtuais).
4. Gestão de armazéns		
RQ 4.1.1	Funcional	O armazém virtual será composto por zonas, estantes, andares e posições.
RQ 4.1.2	Funcional	Cada zona terá uma letra e estará disposta horizontalmente.
RQ 4.1.3	Funcional	Cada estante terá um número e estará disposta verticalmente.
RQ 4.1.4	Funcional	Cada zona será composta por cinco andares, dispostos horizontalmente e numerados de baixo para cima.
RQ 4.1.5	Funcional	Cada andar será composto por cinco posições.
RQ 4.2.1	Funcional	Cada posição deverá ter um contador de quantas paletes estão alocadas ao espaço.
RQ 4.2.2	Funcional	Cada posição deve aparecer pintada de acordo com a palete: amarelo se for condicionada, cinzento se for de cliente e verde se for de *stock*. Será apenas contabilizado o tipo da primeira palete arrumada no local.
RQ 4.3.1	Funcional	Ao clicar numa posição, as paletes presentes nela devem ser apresentadas na listagem de paletes.
RQ 4.5.1	Funcional	Deve ser possível filtrar por local, palete, artigo, número de encomenda, número de cliente e versão de composição da caixa. Deve ainda ser possível filtrar por apenas de *stock* e/ou apenas condicionadas.
RQ 4.5.2	Funcional	Ao filtrar por cliente, a pesquisa deve contemplar os campos nome, nome abreviado e o número de cliente.
RQ 4.6.1	Funcional	A listagem das paletes para arrumar deve apresentar apenas as paletes já aprovadas e as paletes aprovadas que foram condicionadas.
RQ 4.6.2	Funcional	A listagem das paletes deve apresentar o cliente, as referências dos artigos, o local onde está arrumada e a quantidade total da palete.
RQ 4.8.1	Funcional	A arrumação de paletes deve ser realizada através de uma ação de arraste.
RQ 4.8.2	Funcional	Ao arrumar uma palete apenas é atualizado o local na palete.
5. Detalhes da palete		
RQ 5.1.1	Funcional	O detalhe da palete deve ser igual em todos os locais onde apareça.
RQ 5.3.1	Funcional	O detalhe das quantidades por palete deve ser agrupado por lote.
RQ 5.3.2	Funcional	O detalhe das quantidades por palete deve apresentar o total de unidades.
RQ 5.4.1	Funcional	Uma caixa pode ter múltiplos artigos.
RQ 5.4.2	Funcional	Se uma caixa tiver múltiplos artigos, deve aparecer a caixa-pai na listagem de caixas e as caixas-filhos dentro desta, com os detalhes já definidos na US 5.5.
RQ 5.5.1	Funcional	Deve ser possível consultar os detalhes sobre a versão de composição de cada caixa (opção de informação [*i*] ou “Ver mais”).
RQ 5.6.1	Funcional	A opção para editar as observações disponibiliza uma janela *popup* para inserir a informação necessária.

RQ 5.6.2	Funcional	A janela *popup* abre com as observações já existentes e não é feita qualquer validação ao conteúdo.
		6. Transferências
RQ 6.1.1	Funcional	A opção para transferir peças está disponível na consulta dos detalhes da caixa e através de um botão azul, à direita.
RQ 6.1.2	Funcional	As transferências só devem ser possíveis a partir de paletes que não estejam condicionadas.
RQ 6.2.1	Funcional	Ao transferir peças deve ficar registada a data/hora, assim como o utilizador que fez a correção.
RQ 6.2.2	Funcional	A quantidade de peças a transferir de uma caixa tem de ser indicada pelo utilizador.
RQ 6.2.3	Funcional	A quantidade a ser transferida deve ser validada: não pode ser zero e não pode ultrapassar a quantidade existente em caixa.
RQ 6.3.1	Funcional	Só devem aparecer as paletes que tenham caixas com uma versão de composição igual à caixa das peças que estão a ser transferidas, que não tenham a quantidade máxima atingida e que estejam ativas.
RQ 6.3.2	Funcional	Deve ser possível transferir peças de uma caixa para outra na própria palete.
RQ 6.3.3	Funcional	Se uma caixa ficar com a quantidade igual a zero deve ser colocada como inativa.
RQ 6.3.4	Funcional	Se a palete de destino estiver aprovada, passa a estar por aprovar.
RQ 6.5.1	Funcional	A opção para transferir caixas está disponível na consulta dos detalhes da palete e através de um botão azul, à direita.
RQ 6.5.2	Funcional	As transferências só devem ser possível a partir de paletes que não estejam condicionadas.
RQ 6.6.1	Funcional	Ao transferir peças deve ficar registada a data/ hora, assim como o utilizador que fez a correção.
RQ 6.7.1	Funcional	Só devem aparecer as paletes que tenham caixas com uma versão de composição igual às caixas que estão a ser transferidas.
RQ 6.7.2	Funcional	Só deve ser possível transferir caixas dentro do mesmo armazém.
RQ 6.7.3	Funcional	Não deve ser possível transferir caixas para a própria palete.
RQ 6.7.4	Funcional	Se a palete de destino estiver aprovada, passa a estar por aprovar.
RQ 6.8.1	Funcional	Ao transferir uma caixa partilhada devem ser transferidas todas as caixas-filhos da caixa-pai selecionada.
RQ 6.9.1	Funcional	Deve ser possível filtrar por código e designação da palete e número da encomenda.
RQ 6.10.1	Funcional	A opção para transferir de armazém está disponível na consulta dos detalhes da palete e através de um botão azul-claro, à direita.
RQ 6.10.2	Funcional	As paletes são transferidas de armazém individualmente.
RQ 6.11.1	Funcional	Só podem ser transferidas para outro armazém paletes que não estejam condicionadas.

RQ 6.12.1	Funcional	Ao transferir de armazém uma palete tem de ser indicado o armazém de destino.
RQ 6.12.2	Funcional	Na listagem de armazéns disponíveis deve ser logo excluído o armazém atual.
RQ 6.12.3	Funcional	Ao transferir para outro armazém deve ser criado um dossier do tipo "Transferência de armazém".
RQ 6.12.4	Funcional	O dossier "Transferência de armazém" será criado através de um procedimento SQL.
RQ 6.13.1	Funcional	Ao transferir para outro armazém a palete deve ficar por arrumar no armazém de destino.
7. Verificar caixas		
RQ 7.2.1	Funcional	A opção para verificar as caixas de uma palete está disponível na consulta dos detalhes da palete e através de um botão azul, à esquerda.
RQ 7.3.1	Funcional	As caixas são verificadas em massa (selecionadas múltiplas de uma vez).
RQ 7.3.2	Funcional	Ao verificar caixas deve ficar registada a data/hora, assim como o utilizador que realizou a ação.
RQ 7.3.3	Funcional	A informação de verificação passa a estar disponível nos detalhes das caixas, ao consultar uma palete (a exemplo da data/hora de produção).
RQ 7.4.1	Funcional	Ao verificar uma caixa partilhada devem ser dadas como verificadas todas as caixas-filhos da caixa-pai selecionada.
8. Condicionar paletes		
RQ 8.1.1	Funcional	Todas as paletes podem ser condicionadas (com exceção das paletes rejeitadas).
RQ 8.2.1	Funcional	A opção para condicionar uma palete está disponível na consulta dos detalhes da palete e através de um botão amarelo, à esquerda.
RQ 8.3.1	Funcional	As paletes são condicionadas individualmente.
RQ 8.3.2	Funcional	Ao condicionar uma palete deve ficar registada a data/hora, assim como o utilizador que aprovou.
RQ 8.4.1	Funcional	Ao condicionar uma palete tem de ser indicado o motivo de ela ser condicionada.
RQ 8.4.2	Funcional	O motivo de condicionamento é obrigatório. A lista dos motivos é definida numa tabela do *PHC*.
RQ 8.4.3	Funcional	As observações no condicionamento são facultativas, mas não podem ter um mínimo de cinco caracteres e não podem ser submetidas apenas com espaços em branco ou pontos.
RQ 8.4.4	Funcional	Uma palete deve ter a indicação de que está condicionada, assim como deve estar disponível o motivo do condicionamento, ao consultar uma palete. Esta informação deve estar destacada.
RQ 8.5.1	Funcional	A localização de uma palete condicionada pode ser alterada.
RQ 8.5.2	Funcional	Se uma palete estiver arrumada e for condicionada, ela mantém o local onde está arrumada.
RQ 8.6.1	Funcional	Uma palete condicionada não pode ser alterada, ou seja, não pode ter caixas/peças transferidas (nem entrada, nem saída).

RQ 8.6.2	Funcional	Uma palete condicionada não pode ser expedida.
		9. Rejeição
RQ 9.1.1	Funcional	A opção para rejeitar só deve ficar disponível no ecrã "Aprovação de paletes".
RQ 9.2.1	Funcional	A opção para rejeitar peças está disponível na consulta dos detalhes da caixa e através de um botão vermelho.
RQ 9.3.1	Funcional	Ao rejeitar peças deve ficar registada a data/hora, assim como o utilizador que rejeitou.
RQ 9.3.2	Funcional	A quantidade a ser rejeitada deve ser validada: não pode ser zero e não pode ultrapassar a quantidade existente em caixa.
RQ 9.4.1	Funcional	Ao rejeitar peças tem de ser selecionado o motivo da sua rejeição; é um campo obrigatório.
RQ 9.4.2	Funcional	Ao rejeitar peças, a palete passa a estar incompleta, por aprovar e por arrumar.
RQ 9.4.3	Funcional	Ao rejeitar peças deve ser criado um dossier do tipo "Transferência de armazém", para o armazém das rejeições.
RQ 9.6.1	Funcional	A opção para rejeitar caixas/palete está disponível na consulta dos detalhes da palete e através de um botão vermelho.
RQ 9.7.1	Funcional	Ao rejeitar caixas deve ficar registada a data/hora, assim como o utilizador que rejeitou.
RQ 9.8.1	Funcional	Ao rejeitar caixas tem de ser selecionado o motivo da sua rejeição; é um campo obrigatório.
RQ 9.8.2	Funcional	Ao rejeitar caixas, a palete passa a estar incompleta, por aprovar e por arrumar.
RQ 9.8.3	Funcional	Se forem rejeitadas todas as caixas, a palete passa a estar inativa.
RQ 9.8.4	Funcional	Ao rejeitar caixas deve ser criado um dossier do tipo "Transferência de armazém", para o armazém das rejeições.
RQ 9.9.1	Funcional	Ao rejeitar uma caixa partilhada devem ser dadas como rejeitadas todas as caixas-filhos da caixa-pai selecionada.