

MESTRADO EM CIÊNCIA DA INFORMAÇÃO

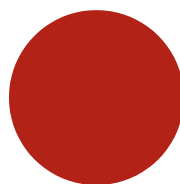
GESTÃO DE INFORMAÇÃO DE COMPONENTES DE SOFTWARE: o caso de estudo da empresa Skrey Software

Maria Dias Lourenço

M
2021

UNIDADES ORGÂNICAS ENVOLVIDAS

FACULDADE DE ENGENHARIA
FACULDADE DE LETRAS



Maria Dias Lourenço

Gestão de Informação de Componentes de Software: o caso de
estudo da empresa Skrey Software

Dissertação realizada no âmbito do Mestrado em Ciência da Informação, orientada
pelo Professor Doutor António Lucas Soares

Faculdade de Engenharia e Faculdade de Letras
Universidade do Porto

Julho de 2021

Gestão de Informação de Componentes de Software: o caso de estudo da empresa Skrey Software

Maria Dias Lourenço

Dissertação realizada no âmbito do Mestrado em Ciência da Informação, orientada pelo Professor Doutor António Lucas Soares

Membros do Júri

Presidente: Professora Doutora Olívia Manuela Marques Pestana
Professora Auxiliar da Faculdade de Letras da Universidade do Porto

Orientador: Professor Doutor António Manuela Lucas Soares
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Arguente: Professora Doutora Maria Cristina Vieira de Freitas
Professora Auxiliar da Faculdade de Letras da Universidade de Coimbra

Agradecimentos

Em primeiro lugar quero agradecer à minha família, por todos os esforços que foram feitos, que sem eles nada disto seria possível, sem eles os restantes agradecimentos não existiam. Em especial ao meu irmão que enche a minha vida e o meu coração de amor desde o primeiro dia.

À Jéssica, minha querida prima, que sempre acreditou em mim e nunca me deixou desistir. Porque em todas as alturas ela faz questão de estar presente tornando as nossas vidas menos solitárias. É um prazer crescer contigo. Mil agradecimentos não chegam.

Aos que, como se costuma dizer, levo desde sempre e para sempre, que apesar de atualmente não fazerem parte do meu dia-a-dia, foram, são e sempre serão um porto de abrigo. Um agradecimento especial às Ritas e ao Manecas, às mil histórias que partilhamos, aos desabafos que nunca mais acabam e à persistência que tanto nos caracteriza.

À Telma, que para uma amizade começar de forma tão peculiar não podia ser mais bonita. Percorremos Portugal, estações do ano e paixões, uma aventura desde o primeiro dia desejando que o último nunca chegue. Um agradecimento por cada vez que me lembraste dos trabalhos que tinha para fazer.

Aos que acompanharam este percurso académico, cheio de altos e baixos, alegrias e dissabores, mas que no fim deixa uma saudade imensa. Aos preferidos, Marta, Fábio, Diana, Tiago, Mariana, Eva, Dantas, Milene e Diogo, que me ensinaram a sonhar em noites de luar. Fizeram-me crescer de dia para dia, tornando-se na melhor surpresa que eu podia ter recebido, um obrigada por tornarem esta etapa memorável.

Ao 77, que não é só uma casa no sentido literal, mas também figurativamente. A todos que fizeram esta etapa mais suportável, principalmente o Zé Pedro que tornou o meu sonho de criança realidade.

À Skrey por me ter recebido, a todos que se mostraram sempre disponíveis e compreensíveis para tudo o que eu precisasse.

Por fim, quero agradecer ao meu orientador, Professor António Lucas Soares, por toda a partilha de conhecimento, disponibilidade e paciência durante a realização deste projeto, orientando-me sempre pelo melhor caminho.

A todos, um muito obrigada!

Resumo

Quando associamos a gestão do conhecimento ao desenvolvimento de software esta tem como objetivo primordial melhorar a produtividade de cada organização, garantir uma melhoria contínua dos seus produtos e serviços, e, conseqüentemente, trazer vantagem competitiva para a mesma. Esta realidade é alcançada quando a organização consegue tornar o seu fluxo de trabalho eficaz e eficiente, transformando o conhecimento tácito e explícito em CO acessível promovendo a autoaprendizagem. Foi identificado um crescente reconhecimento da importância de uma gestão do conhecimento e do conteúdo para a área, contudo, focam apenas o investimento no uso das novas tecnologias e não nas boas práticas que devem acompanhar. Na atualidade este tipo de organizações seguem cada vez mais os princípios fixados no manifesto agile, focando-se na comunicação, partilha do conhecimento e feedback frequente por parte do cliente, o que incentiva a existência de uma colaboração constante entre todos os membros da organização e os clientes. A presente dissertação foca-se no desenho de um processo para a gestão de informação de componentes de software, com o objetivo de aprimorar as práticas de gestão de conhecimento e de conteúdo através da elaboração de documentação técnica e de utilizador. De destacar que um documento em engenharia de software é algo que descreve o produto que está a ser desenvolvido e contém instruções para a sua utilização, tendo como características ser conciso e compreensível. A metodologia seguida garante que sejam identificadas todas as necessidades da empresa, através de uma validação e gestão constante das mesmas. Como resultado foi criado um processo à medida da realidade da empresa e baseado no identificado anteriormente na revisão de literatura, porém o trabalho desenvolvido destina-se apenas a módulos de software, componentes de pequenas dimensões e complexidade que garantem certas funcionalidades à solução criada, ao contrário do constatado previamente. Com o trabalho desenvolvido é esperado um maior envolvimento da empresa em práticas de gestão de informação, contribuindo para um maior envolvimento da ciência da informação na área do desenvolvimento de software que ainda apresenta diversas lacunas.

Palavras-chave: Gestão de informação; Gestão de documentos; Desenvolvimento de software; Módulos de software.

Abstract

When we associate knowledge management with software development, its main objective is to improve the productivity of each organization, to guarantee a continuous improvement of its products and services, and, consequently, to bring competitive advantage to the organization. This reality is achieved when the organization manages to make its workflow effective and efficient, transforming implicit and explicit knowledge into accessible organizational knowledge promoting self-learning. It was identified growing recognition of the importance of knowledge and content management for the area, however, they only focus the investment in the use of new technologies and not in the good practices that should accompany. Nowadays, this type of organization follows more and more the principles established in the agile manifesto, focusing on communication, knowledge sharing, and frequent feedback from the customer, which encourages a constant collaboration between all the members of the organization and the customers. This dissertation focuses on the design of a process for the information management of software components, to improve knowledge and content management practices through the development of technical and user documentation. It should be noted that a document in software engineering is something that describes the product that is being developed and contains instructions for its use, having characteristics to be concise and understandable. The methodology followed guarantees that all the company's needs are identified, through constant validation and management. As a result, it was created a process tailored to the reality of the company and based on what was previously identified in the literature review, however, the work developed is aimed only at software modules, components of small dimensions and complexity that guarantee certain features to the solution created, contrary to what was previously found. With the work developed, it is expected a greater involvement of the company in information management practices, contributing to greater involvement of the information science in the software development area that still presents several gaps.

Keywords: Information management; Document management; Software development; Software modules.

Lista de ilustrações

Figura 1 – Árvore de objetivos	4
Figura 2 – Componentes da Gestão do Conhecimento.....	10
Figura 3 – Métodos tradicionais VS Metodologias ágeis para o desenvolvimento de software	12
Figura 4 – Framework Scrum	14
Figura 5 – Framework XP	15
Figura 6 – Engenharia de Requisitos: Contexto do Sistema	30
Figura 7 – Engenharia de Requisitos: Atividades Principais.....	31
Figura 8 – Engenharia de Requisitos – Artefactos de Requisitos	32
Figura 9 – Processo de identificação de requisitos	37
Figura 10 – Processo de desenvolvimento dos módulos	38
Figura 11 – Processo de realização de testes	39
Figura 12 – Processo de elaboração do manual de utilizador	40
Figura 13 – Ciclo de vida dos documentos.....	57
Figura 14 –Ciclo de vida do documento de requisitos	60
Figura 15 – Ciclo de vida do documento de arquitetura	60
Figura 16 – Ciclo de vida do documento de API	61
Figura 17 – Ciclo de vida dos documentos de testes	62
Figura 18 – Ciclo de vida do manual de utilizador.....	63
Figura 19 – Arquitetura do sistema de registos	66
Figura 20 – Integração dos documentos no Redmine.....	68

Lista de tabelas

Tabela 1 - Documentos propostos para o desenvolvimento de software	21
Tabela 2 – Documentos criados pela Skrey no desenvolvimento dos módulos	42
Tabela 3 – Proposta de documentos a serem criados durante o desenvolvimento dos módulos	53
Tabela 4 – Fase para a criação dos documentos	54
Tabela 5 – Estado dos documentos.....	58

Lista de abreviaturas e siglas

API	– Application Programming Interface
CI	– Ciência da Informação
CO	– Conhecimento Organizacional
DS	– Desenvolvimento de Software
DM	– Desenvolvimento do módulo
ER	– Engenharia de Requisitos
ES	– Engenharia de Software
GC	– Gestão do Conhecimento
GCC	– Gestão do Conhecimento e de Conteúdos
GD	– Gestão Documental
GI	– Gestão de informação
OSD	– Organizações de Desenvolvimento de Software
VSCode	– Visual Studio Code
WHMCS	– WebHost Manager Complete Solution
XP	– Extreme Programming

Índice

1	Introdução.....	1
1.1	Enquadramento.....	1
1.1.1	Descrição do problema	2
1.1.2	Objetivos e resultados esperados.....	3
1.2	Estrutura da dissertação.....	6
2	Revisão de Literatura.....	7
2.1	Gestão do Conhecimento em Organizações de Desenvolvimento de Software	7
2.2	Gestão de Conteúdos no Desenvolvimento de Software	11
2.2.1	Processo de Desenvolvimento de Software	11
2.2.2	Processo de Gestão Documental no Desenvolvimento de Software	17
2.3	Ferramentas para a Gestão de Conteúdos no Desenvolvimento de Software.....	22
2.3.1	Ferramentas de Gestão de Projetos	22
2.3.2	Ferramentas para a Gestão de Conhecimento	25
2.3.3	Ferramentas de Comunicação.....	27
3	Metodologia	29
4	A Gestão de informação no desenvolvimento de módulos de software	34
4.1	Caso de Estudo: Skrey Software	34
4.2	Caracterização do processo de desenvolvimento dos módulos de software	36
4.2.1	Caracterização dos documentos.....	40
4.2.2	Ferramentas utilizadas.....	42
4.3	Diagnostico da Gestão de Informação na empresa	47
5	Proposta para processo de Gestão de Informação para componentes de software	49
5.1	Documentos a serem criados.....	49
5.1.1	Ciclo de vida dos documentos.....	53
5.2	Processo para a Gestão dos Documentos	63
5.3	Validação	68

6	Análise dos Resultados e Perspetivas futuras.....	72
7	Conclusões	75
	Referência bibliográficas.....	77
	Anexos	83
	Anexo A: Abordagem metodológica	83
	Anexo B: Processo de desenvolvimento dos módulos de software	84
	Anexo C: Proposta de documentos a serem criados.....	85
	Anexo D: Processos para o desenvolvimento de documentos	88
	Anexo E: Integração da solução no Redmine.....	93
	Anexo F: Exemplo da integração no Redmine	94
	Anexo G: Exemplo de documentos.....	95

1 Introdução

1.1 Enquadramento

A proposta de dissertação a ser desenvolvida foi apresentada pela Skrey E-Commerce Software - Skrey Software, designada por Skrey daqui por diante. A Skrey é uma software house que lida com retalho digital, nomeadamente plataformas de comércio eletrónico, marketplaces online e aplicações móveis, construindo soluções avançadas para os seus clientes, recomendando-lhes qual a melhor forma de se vincularem ao comércio eletrónico com a finalidade de retirarem proveito da melhor tecnologia disponível no mercado.

As organizações de desenvolvimento de software (ODS) servem-se, em grande escala, das metodologias ágeis - um conjunto de métodos que defendem diferentes práticas para o desenvolvimento de software (DS), porém todas elas seguem uma série de princípios que podem ser agrupados na partilha, comunicação, e no envolvimento do cliente permitindo a flexibilidade do projeto. O desenvolvimento de um software de grandes dimensões é composto por vários componentes de software de menor escala, nomeados daqui por diante de módulos. Esses módulos garantem à solução uma série de funcionalidades previamente identificadas, não sendo exclusivos de um só projeto ou de um software, podendo ser reutilizados para outras soluções. Na prática podemos identificar duas diretrizes que podem ser seguidas no desenvolvimento dos módulos (DM), sendo a primeira quando estes são pensados e desenhados especificamente para um projeto, porém são desenvolvidos como unidades independentes não ficando apenas associados ao projeto para o qual foram criados podendo ser utilizados noutros projetos, através da sua integração na solução que esteja a ser desenvolvida. Para além do DM no contexto de um projeto específico é também comum criar módulos que tragam valor através da sua diferenciação para que depois possam ser integrados em projetos ou sejam vendidos de forma independente, sendo esta a segunda diretriz para o desenvolvimento de componentes de software.

A Skrey não mantém nenhuma metodologia ágil específica, mas uma customização do método Scrum e Extreme Programming (XP), chegando a ser considerada uma metodologia própria da empresa para se dedicar ao DM, para além dos módulos que são criados num contexto de um projeto específico para um cliente, mantém a prática de desenvolver módulos que possam posteriormente ser vendidos a futuros clientes que queiram apenas adquirir as funcionalidades que os mesmos dispõem. Contudo, os módulos que são desenvolvidos para clientes podem também ser reutilizados posteriormente para novos projetos ou interessados, podendo ainda

sofrer alterações consoante as necessidades apresentadas por quem os requisitar. Com a construção dos módulos a empresa tem em vista procurar novas funcionalidades que possam agregar valor ao seu produto final, ganhando assim vantagem competitiva e reconhecimento no mercado, essa prática é feita através de um estudo de mercado para realizar o que existe e o que ainda pode ser criado, mas também através do conhecimento dos seus colaboradores.

Para empresas de DS o seu grande ativo é o conhecimento tácito de cada membro, denominado de Conhecimento Organizacional (CO). É a partir do aproveitamento deste ativo que as organizações conseguem ganhar vantagem competitiva, mais concretamente através da diferenciação que podem causar, na reutilização do conhecimento e na melhor gestão dos seus recursos. No caso de estudo que será apresentado, a empresa dedica-se ao aproveitamento desse conhecimento para conceber produtos e serviços inovadores que possam ser de interesse dentro do seu mercado de atuação. Embora na generalidade das empresas reconhecerem a importância da adoção de estratégia de Gestão de Conhecimento (GC) negligenciam as suas práticas valorizando apenas a implementação de tecnologias para essa finalidade, os conhecidos sistemas de GC, abandonando o princípio base que para o total aproveitamento da tecnologia é necessário a adoção de práticas constantes que mantenham o sistema. Esta realidade vai melhorar o seu processo de DS, tornando-o mais rápido e eficaz garantindo a qualidade dos seus produtos e serviços, aumentar a consciencialização para a necessidade de partilha e reutilização de conhecimento e, para além disso, garantir uma melhor relação de comunicação interna e externa, respetivamente com os colaboradores e clientes.

1.1.1 Descrição do problema

A Skrey, como já referido anteriormente, não se dedica apenas aos projetos onde se comprometem a desenvolver software à medida dos seus clientes, dedica-se também à construção de componentes de software independentes como, por exemplo, módulos específicos para áreas como a gestão de campanhas, gestão de recursos ou clientes, ou plugins que podem ser incorporados noutros programas e sistemas de maior dimensão. Com o seu desenvolvimento, quer sejam as soluções desenhadas exclusivamente para os seus clientes, quer sejam componentes independentes, é criada uma série de documentação técnica que acompanha todo o processo de DM, terminando em manuais de utilizadores a serem disponibilizados aos seus clientes e colaboradores facilitando assim a sua compreensão e perceção sobre os produtos descritos.

Contudo, o processo de criação e organização desses documentos, não se encontra devidamente estruturado, fazendo com que práticas de Gestão Documental (GD) sejam muitas

vezes adiadas. Também em questões quanto à reutilização e atualização dos documentos o cenário é equivalente. Quanto à sua disponibilização aos clientes e colaboradores podemos dividir a questão em dois polos opostos, quando se trata de produtos concebidos e operacionais para clientes o manual de utilizador é enviado ao cliente aquando colocação online da solução tecnológica desenhada, no que diz respeito aos componentes autónomos os manuais de utilizadores foram apenas criados após a sua conceção.

O principal problema encontra-se na quantidade de conteúdo gerado ao longo do desenvolvimento dos componentes de software que necessita ser devidamente documentado, não só no final por manuais de utilizador. Contudo, a prática de documentar o processo ao longo do caminho é ainda descurada, por vezes por falta de tempo ou falta de alguém dedicado a essa tarefa, mas por outras vezes não saber quando o fazer. O trabalho a realizar irá debruçar-se sobre a documentação dos módulos onde cada componente de software deverá encontrar-se apropriadamente descrito, o que está em desacordo com a realidade da empresa que a conceção da documentação não acompanha o processo de DM.

Posto isto, o projeto a desenvolver tem como objetivo principal a conceção de um processo para a criação de documentação de software, nomeadamente documentação técnica e de utilizador de componentes de software, tendo também em conta questões como o seu armazenamento e disponibilização, com forma de facilitar o acesso aos mesmos por partes de colaboradores, clientes atuais e futuros, para isso a empresa conta com uma maior valorização dos seus processos internos adequando a elaboração da documentação ao processo já existente para o DM. A Skrey reconhece que o progresso do seu negócio se encontra numa reformulação dos seus processos, como a conceção dos manuais que deverá ser alinhada com o processo de construção dos módulos com a finalidade de serem redigidos em simultâneo, para posteriormente facilitar a organização e disponibilização da documentação online como forma de promover o seu trabalho e o seu conteúdo organizacional.

1.1.2 Objetivos e resultados esperados

Um passo fundamental para dar início a este trabalho foi realizar um estudo sobre as metodologias de DS com a finalidade de entender o seu funcionamento e qual o seu procedimento para a criação de documentação. Com este intuito foi definido o objetivo “Caracterização do estado-da-arte na gestão de conhecimento em empresas de desenvolvimento de software”, e através do reconhecimento das práticas de Gestão de Conhecimento e de Conteúdo (GCC) será possível definir quais as melhores práticas a serem implementadas em empresas de DS. Durante a realização da pesquisa inicial foi notório que

existe em grande escala uma sensibilização para as boas práticas de criação de documentação, explicando os seus benefícios e como deve ser feita. Os principais pontos a retirar são a necessidade de documentar tudo de forma sucinta e clara, que deve ser feita ao longo do desenvolvimento do projeto para que todos os pormenores sejam referidos sendo um trabalho colaborativo entre todos os integrantes da equipa.

Se por um lado é necessário realizar um reconhecimento do domínio e identificar quais as melhores práticas atuais para a concretização de um processo de tal envergadura, por outro lado, é também fundamental fazer um reconhecimento aprofundado da realidade da empresa. Inicialmente é fundamental discernir todo o processo de criação de componentes de software da empresa, dando especial atenção à documentação criada e, posteriormente, realizar um levantamento de requisitos para que seja desenhado um processo que vá de encontro a todas as necessidades apresentadas pela empresa, adequando-o aos recursos que a mesma dispõe. Para completar este reconhecimento deve ser feita a caracterização de toda a informação documentada existente pelas suas tipologias e pelo valor que lhes é agregado pela empresa, com o objetivo de identificar os próximos passos a serem seguidos e a seleção da metainformação essencial para a descrição dos componentes facilitando a sua organização e posterior recuperação. O cumprimento destas tarefas irá cumprir o objetivo “Classificação dos processos de desenvolvimento de software do ponto de vista informacional”.

Por conseguinte, foram destacados dois grandes objetivos que podem ser vistos na árvore de objetivos abaixo. Eles contribuíram para um maior entendimento sobre o tema e para definir quais os requisitos para o desenvolvimento do sistema de informação.

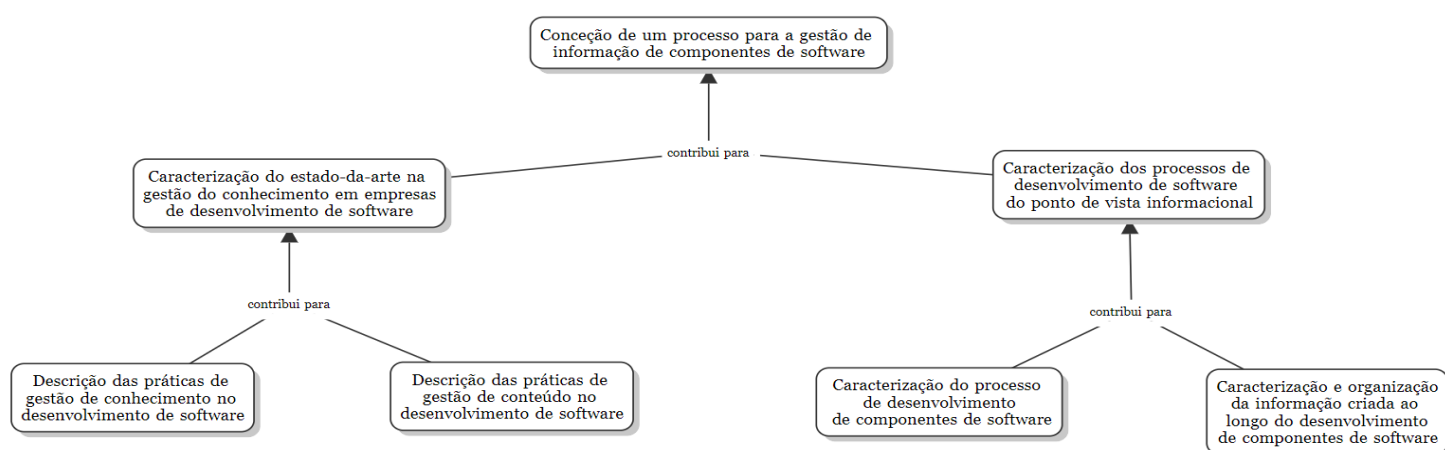


Figura 1 – Árvore de objetivos

Fonte: Autoria própria

É esperado que através da concretização dos objetivos haja uma maior percepção sobre a área de DS e, após a realização de um reconhecimento interno da empresa que colmatará num levantamento de requisitos, seja possível identificar quais as práticas de GCC que devem ser aplicadas na Skrey. A sua meta encontra-se no desenho do processo para a elaboração de documentação aquando do DM de software e um plano de Gestão de Informação (GI) para esses mesmos documentos que ajudará a empresa na sua organização e recuperação de forma eficiente.

Com o desenvolvimento desta dissertação de mestrado espera-se contribuir para o envolvimento da Ciência da Informação (CI) na área de Engenharia de Software (ES), mostrando como esta pode ser inserida em todo o processo de DS e, consequentemente, os benefícios alocados. Mais especificamente, aprimorar os processos internos de documentação dos componentes de software desenvolvidos pela Skrey, nomeadamente as práticas de GC alicerçadas numa gestão de conteúdos, impulsionando uma maior organização dos seus produtos em plataformas adequadas e descritos corretamente. Consequentemente, em primeira instância a empresa beneficiará de um processo para a elaboração de documentos de software de forma correta e nos tempos apropriados, e em segundo plano e através da sua melhoria a empresa conseguirá retirar vantagens uma vez que a documentação de encontrará devidamente estruturada pronta para ser aproveitada, atualizada e reutilizada, facilitando a sua recuperação por partes dos clientes e colaboradores, poupando tempo e esforços. Para além destes pontos, espera-se também que seja benéfico para a comunicação com os clientes e com os colaboradores atuais assim como na angariação de novos, dado que parte dos documentos elaborados irão servir de ferramenta para uma estratégia de propaganda da empresa facilitando a interação com o mercado.

A nível pessoal este projeto trará um maior conhecimento sobre a área em questão de DS, onde a CI tem dado cartas e provando o seu valor destacando certos aspetos que por vezes são descorados, mas que podem trazer valor inigualável para as empresas que se dedicam ao DS. Na prática, espera-se conseguir provar os benefícios que uma GD adequada pode trazer, não só através de boas práticas para a criação de documentação, mas também garantindo que esses documentos estão devidamente estruturados e que contém todos os conteúdos fundamentais. A par desta realidade é também esperado sensibilizar que a questão referida anteriormente é essencial para aumentar o CO e individual. Para além do já referido, é de destacar que se trata de uma organização que tem o seu método de trabalho baseado em Gestão de Projetos (GP), uma realidade que abrange cada vez mais o mercado de trabalho, o que será benéfico para alargar horizontes sobre o funcionamento desta metodologia, não só em termos teóricos, mas também a nível prático.

1.2 Estrutura da dissertação

A presente dissertação encontra-se dividida em sete capítulos principais que serão abordados de seguida, contando ainda com duas secções complementares destinadas às referências bibliográficas e aos anexos, respetivamente.

O presente capítulo “Introdução” destina-se à contextualização do projeto, apresentando a descrição do problema, quais os objetivos com a realização deste trabalho e os resultados esperados. Por fim, é apresentada toda a dissertação.

No segundo capítulo podemos encontrar a revisão de literatura que contém a revisão do estado da arte sobre o tema da dissertação, tornando-se fundamental para o desenvolvimento da mesma. Para isso o capítulo encontra-se dividido em “Gestão do Conhecimento em Organizações de Desenvolvimento de Software”, estudo fundamental para entender como organizações equivalentes ao caso de estudo gerem o seu conhecimento; “Gestão de Conteúdos no Desenvolvimento de Software”, enquadrando o funcionamento das organizações de software no seu quotidiano com o conteúdo e documentos criados, assim como a sua relação com os clientes; por fim o último tópico destina-se à identificação das ferramentas mais usadas.

Seguidamente temos um terceiro capítulo onde é apresentada a metodologia empregue para realizar os objetivos anteriormente definidos, exibindo as suas diretrizes e realizando a ponte entre estas e a sua aplicabilidade. Como os objetivos definidos para o desenvolvimento da presente dissertação têm uma só finalidade, sendo ela a especificação de requisitos que permitam desenhar um processo para a GI de componentes de software, foi escolhida a metodologia de Klaus Pohl de Engenharia de Requisitos.

No capítulo seguinte, o quarto, é apresentado o caso de estudo, onde podemos encontrar o problema exposto detalhadamente. Neste capítulo é possível identificar as necessidades da empresa, através da caracterização do seu processo de desenvolvimento dos produtos, do reconhecimento de todos os documentos produzidos, assim como a sua estrutura, e das ferramentas utilizadas.

Após este capítulo onde foi realizado um profundo reconhecimento da empresa em estudo, segue-se o quinto capítulo onde podemos encontrar uma proposta que responde às necessidades identificadas anteriormente devidamente fundamentadas. Conta também com um ponto dedicado à validação dessa mesma proposta.

O penúltimo e sexto capítulo podemos encontrar uma análise dos resultados alcançados e perspectivas futuras para o trabalho realizado. O sétimo capítulo dedica-se às principais conclusões retiradas ao longo do desenvolvimento da presente dissertação.

2 Revisão de Literatura

2.1 Gestão do Conhecimento em Organizações de Desenvolvimento de Software

Vários autores referem Nonaka (1994), que identifica duas dimensões do conhecimento, conhecimento explícito e conhecimento tácito. O primeiro refere-se ao conhecimento que pode ser representado em documentação formal, como por exemplo manuais, e posteriormente transferido através dos mesmos. Já o segundo tipo de conhecimento, apresentado pelo autor é um saber implícito, por outras palavras é o conhecimento que cada indivíduo vai adquirindo e retendo ao longo da sua vida sendo mais complexa a sua representação e, como consequência, mais difícil a sua transmissão. Ainda na mesma publicação, Nonaka afirma que o conhecimento pode ser individual, criado e existente numa só pessoa, ou coletivo, criado socialmente através de ações coletivas de várias pessoas.

Ainda não existe um consenso nem provas irrefutáveis sobre qual conhecimento será mais importante para a GC, contudo é reconhecido pelos autores que ambos são essenciais. Weinreich e Groher (2016), vão mais além, *"If not explicitly managed, critical knowledge remains tacit knowledge, which erodes over time."* tal como referido na obra de Babar et al., "Software Architecture Knowledge Management: Theory and Practice" (citado em Weinreich e Groher 2016), retratando a necessidade de capturar o conhecimento que está implícito dentro de cada pessoa para que este não desapareça com o passar do tempo ou para quando a pessoa que o contém abandone a organização, o seu conhecimento permaneça como CO.

Para Maciel et al. (2018) a GC surge como uma disciplina que tem como objetivo adquirir, organizar e partilhar o conhecimento para que possa ser reutilizado. Dentro de uma organização este processo torna o trabalho de cada um mais eficaz e eficiente, tornando todo o CO acessível promovendo a criação de novos conhecimentos. Para os autores, é através da GC que as organizações conseguem retirar valor dos seus ativos, nomeadamente com a reutilização de conhecimento de atividades anteriores e da experiência dos membros da organização, *"... the experience from each activity conducted can be collected and packaged and stored in an experience base to be easily reused, documented, and thus it can be accessed by multiple organization members."* (Maciel et al. 2018, 2), que a longo prazo trazem benefícios para a organização. Na sua pesquisa Priya e Selwyn (2016), provam que ao usufruir do conhecimento dos ativos da organização, assim como reutilizar conhecimento de projetos anteriores, evitando repetição de tarefas tornando-se possível conservar esforços, tempo e reduzir os custos.

Bjornson e Dingsoyr (2008) realizaram uma revisão sistemática da literatura existente sobre a aplicação da GC em ES e, consequentemente, dos conceitos e práticas da GC no DS. Quando abordam o conceito de GC afirmam que é "... *a method that simplifies the process of sharing, distributing, creating, capturing and understanding of company's knowledge*" tal como referido na obra de Davenport e Prusak "Working Knowledge: How Organizations Manage What They Know" (citado em Bjørnson e Dingsøyr 2008), seguidamente de Hanssen et al. que apresentam duas estratégias envoltas na GC, a codificação e a personalização. A primeira sintetiza e armazena a informação que constitui o CO disponibilizando-o a todos os membros da empresa, a segunda, personalização, apoia o fluxo dessa mesma informação que, por sua vez, deve estar centralizada e conter as fontes do conhecimento. É neste segundo ponto que, segundo os autores, se encontra um dos maiores desafios da GC, para que facilite o fluxo de troca de conhecimento entre os indivíduos para que haja a máxima transparência, e onde pode surgir um dos maiores problemas, "... *too little can result in expensive mistakes, while too much can lead to unwanted accountability.*" tal como referido na obra de Schultze e Leidner "Studying Knowledge management in information systems research: discourse and theoretical assumptions" (citado em Bjørnson and Dingsøyr 2008).

Quando a GC é incorporada no DS tem como objetivo primordial melhorar a produtividade de cada organização específica, os seus processos e produtos assim como a rastreabilidade de soluções eficazes, trazendo vantagem competitiva para a mesma. Na visão de Montoni et al. (2004) a identificação, manutenção e disseminação de processos de DS, de projeto em projeto, é fulcral para o DS com alta qualidade, este conhecimento é intrínseco a cada organização e, como tal, vai variando de empresa para empresa consoante os seus membros. Indumini e Vasanthapriyan (2018) defendem que o CO é o ativo mais valioso que uma organização pode possuir e, para complementar essa ideia, citam Dingsoyr e Smite quando estes afirmam que a GC pode ser definida com uma gestão sistemática dos mais valiosos ativos de uma organização que criam valor e definem os requisitos para o desenho da estratégia organizacional. Para provarem a teoria dos autores citados, realizaram um estudo qualitativo e comprovaram que o problema nas organizações é a baixa taxa de reutilização do conhecimento adquirido em projetos anteriores, e a implementação de uma estratégia baseada na GC numa organização de DS pode trazer benefício como o aumento da eficácia, do uso adequados das técnicas e tecnologias e a longo prazo trazer vantagem competitiva. Contudo, e para além das vantagens da GC, os autores apresentam duas desvantagens, como o uso de sistemas de GC que não são apropriados para a organização o que pode levar a uma maior carga de trabalho para os seus funcionários. Também Montoni et al. (2004) apresentam benefícios da GC retirados do estudo que realizaram e estes focam-se em: reutilização eficiente de experiências documentadas;

maior facilidade de encontrar soluções para problemas organizacionais; identificar e armazenar o valor de cada experiência; facilidade de propor medidas para melhorar os processos internos; aumentar a qualidades dos seus produtos/serviços.

O processo de DS está dividido em atividades variadas e, em cada uma delas, podemos identificar práticas de GC. As atividades estão divididas nas fases de: especificação e análise de requisitos; planeamento do projeto; programação; fase de testes; correção de erros e manutenção. É de referir que as ODS trabalham essencialmente por projetos, para cada projeto é selecionado um conjunto de pessoas, com variadas formações necessárias ao DS, para formar uma equipa multifuncional e dentro de cada equipa o trabalho é dividido pelas pessoas consoante as suas áreas de atuação, Dorairaj et al.(Dorairaj, Noble, and Malik 2012) (2012) definem equipas multifuncionais como "*Cross-functionoal team just means that the team as a whole as all skills needed to build the software, and that every team member is willing to coordinate, communicate and collaborate to get the job done.*" (Dorairaj, Noble, e Malik 2012, 1). No decorrer de um projeto a informação sobre o mesmo tem de passar de pessoa em pessoa, de forma intacta para que cada fase seja concluída com sucesso. Para além disso, o diferente conhecimento que cada integrante da equipa retém deve ser aproveitado tornando cada membro valioso para a organização.

Existem muitas abordagens de como o software deve ser desenvolvido, o que também afeta a GC. Até ao final dos anos 90 as metodologias seguidas para o DS enfatizavam a necessidade de uma extensa documentação do conhecimento explícito, codificado em manuais e relatórios pormenorizados e consistentes. Com o passar do tempo, a prática de documentar intensivamente todo o processo é vista como um desperdício de tempo e ativos de uma organização. Neste sentido assistiu-se ao surgimento de métodos ágeis, que baseiam os seus princípios na comunicação, partilha de conhecimento e de feedback frequente, o que incentiva a existência de uma estreita colaboração entre todos os membros de uma organização.

Khalil e Khalil (2020) citam vários autores e chegam à conclusão que as equipas ágeis - equipas formadas para o DS tendo como bases metodologias ágeis - comparecem em reuniões diárias onde discutem o avanço do projeto e têm como objetivo intrínseco promover a comunicação e colaboração entre os membros das equipas para resolver problemas que surgem, realizar testes de unidade e recolher feedback sobre o código, garantindo um melhor controlo do projeto. Assim, "*Thus, agile engineering and management practices provide a context where team members are predisposed to combining, creating and sharing their knowledge.*" (Khalil e Khalil 2020, 5), é através destas atividades que grande parte do conhecimento relacionado com o software é criado tornando-se imprescindível a sua captura.

Bjornson e Dingsoyr (2008), explicam que as empresas de DS carecem de um melhoramento contínuo dos seus serviços e produtos para se manterem na concorrência. Como referido anteriormente, à luz da GC é possível criar uma estratégia que foque a melhoria dos processos de DS e, em sequência, a melhoria dos seus serviços e produtos. Rabelo et al. (2015) concordam com esta visão sobre a GC, completando que a cultura organizacional é fundamental nesse processo. Em ES a reutilização de conhecimento adquirido através de experiências anteriores e com outras pessoas é frequentemente referido como "Experience Factory", onde assistimos à recolha e armazenamento dos projetos anteriores, chamada "Experience base", seguidamente de uma adaptação e formalização do conhecimento capturado para facilitar a sua reutilização, denominada de "Bypacking", o que resulta no que podemos qualificar de CO.

Aurum, Daneshgar e Ward (2008) realizaram um estudo tendo como objetivo relacionar as práticas e as atividades de GC usadas em ES, e por fim quais os impactos que uma estratégia de GC pode provocar. No que diz respeito às práticas reconhecem que se baseiam em quatro elementos: o tipo de conhecimento (tácito ou explícito); a motivação para a partilha de conhecimento tornando-o numa prática regular; incentivar a autoaprendizagem e a procura por novas fontes de conhecimento; e a instalação de um sistema próprio para a GC que vá de entre as especificidades de cada organização.

Em suma, podemos dividir o conhecimento em conhecimento explícito e tácito que devem ser igualmente tratados pela GC, que tem como objetivo a captura, organização, armazenamento e partilhar o conhecimento.



Figura 2 – Componentes da Gestão do Conhecimento

Fonte: Autoria própria

Quando aplicada à ES tem como objetivo primordial melhorar o DS encontrando-se presente durante todo o processo, no qual é criado conhecimento diversificado, comumente denominado de CO, que tem a necessidade de ser codificado, quer seja explícito ou tácito, para que possa ser reutilizado. Através da revisão de literatura realizada sobre o tema, podemos afirmar que ele pode trazer diversos benefícios para a organização, sendo reconhecida a sua importância pelas ODS, contudo, muitas focam-se apenas em tecnologias para ajudar na sua prática menosprezando a adoção de boas práticas. O DS é realizado por etapas, divididas em várias tarefas, na atualidade a maioria das organizações adotam metodologias ágeis como suporte à sua realização. Os métodos ágeis enfatizam a comunicação, a partilha de conhecimento e a valorização do feedback, desvalorizando a extensa e pormenorizada documentação, mas não descartando a obrigação de documentar todas as etapas do DS. A GC desempenha um papel importante dentro das metodologias ágeis, uma vez que o conhecimento tácito é produzido em grande escala e este necessita de ser registado para que não se perca com o passar do tempo ou com a saída das pessoas que o possuem da organização. Além disso, garante que o conhecimento que já se encontra codificado seja tratado e organizado de forma adequada.

2.2 Gestão de Conteúdos no Desenvolvimento de Software

2.2.1 Processo de Desenvolvimento de Software

Os métodos tradicionais usados no DS, como por exemplo o modelo em cascata, estão mais ligados aos processos e às ferramentas, exigiam documentação detalhada, sendo vistos como uma simples contratação onde o plano era definido no início do projeto e seguido até ao fim sem que seja submetido a alterações. Um conjunto de métodos veio contradizer essa realidade, esse conjunto é designado de metodologias ágeis, "*Methods for agile software development constitute a set of practices for software development (...) These methods can be seen as a reaction to plan-based or traditional methods (...) in which it is claimed that problems are fully specifiable and that optimal and predictable solution exist for every problem.*" (Dybå and Dingsøyr 2008), com intenção de completar a sua definição de métodos ágeis é essencial abordar o conceito de agilidade, "*agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like.*" tal como referido na obra de Ericksson et al. "Agile Modeling, Agile software development, and extreme programming: the state of research" (citado em Dybå e Dingsøyr 2008). Para colmatar esta ideia, o conceito de ágil é definido como

"the continual readiness of an ISD (information systems development) method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment." tal como referido na obra de Conboy "Agility from first principles: reconstructing the concept of agility in information systems development" (citado em Yu e Petter 2014).

Na figura abaixo podemos analisar a diferença entre os métodos tradicionais, mais concretamente o modelo cascata, e as metodologias ágeis.

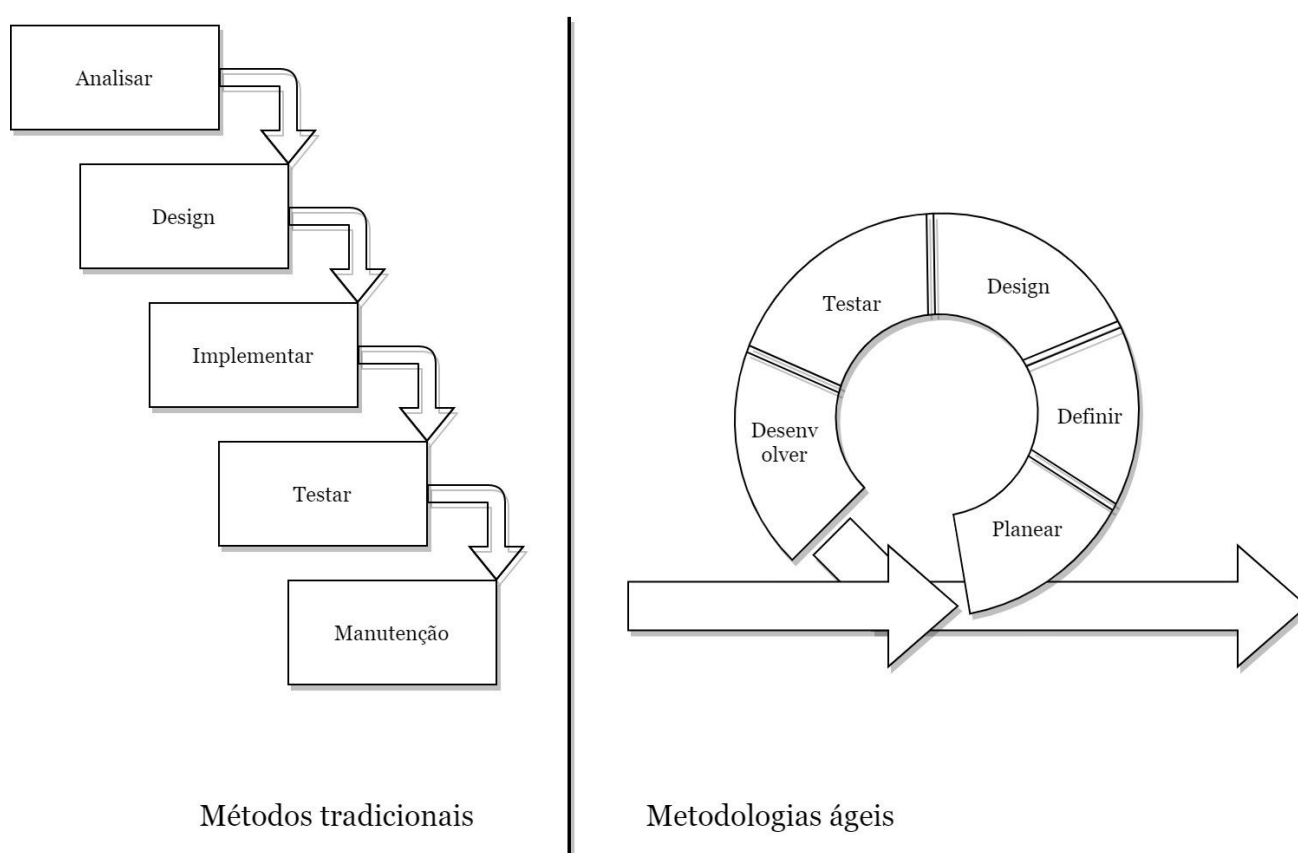


Figura 3 – Métodos tradicionais VS Metodologias ágeis para o desenvolvimento de software

Fonte: Autoria própria adaptada de (Saigon Technology 2021).

O manifesto agile (Cunningham 2001) oficializou estes métodos em 2001 quando um grupo de 17 desenvolvedores de DS se reuniram e discutiram soluções alternativas para a GP, o manifesto não impõem nenhuma prática em específica afirmando que os métodos ágeis foram criados para indivíduos e iterações, para criar software funcional focado na colaboração com

o cliente e para responder rapidamente à mudança. Para alcançar esta mudança foram criados uma série de princípios que devem ser seguidos: satisfação do cliente através de entregas rápidas e contínuas; alteração de requisitos; fornecer frequentemente software funcional; trabalhar em conjunto com o cliente; apoiar a equipa de desenvolvimento; incentivar a comunicação e a partilha; desenvolvimento sustentável; excelência técnica; simplicidade de trabalho; equipas auto-organizadas; reuniões e reflexões constantes. Em suma, podemos traduzir os seus princípios em: flexibilidade; divisão do trabalho em pequenos ciclos; valorização da equipa; reavaliação constante do trabalho; cooperação com o cliente.

As metodologias ágeis estão desenhadas para conseguir responder às mudanças no mercado ou ao feedback dos clientes sem que haja um prejuízo em grande escala, Yu e Petter (2014) consideram que o desenvolvimento ágil de software é leve e incremental, constituído de práticas específicas que enfatizam a interação próxima com os clientes. A sua gestão é considerada uma abordagem iterativa, denominada de gestão ágil de projetos de DS que, por sua vez, diminui o tempo necessário para o desenvolvimento do software através do aumento de colaboração. Porém, o nível de gestão requerido não é idêntico ao que estamos acostumados a assistir, as equipas são multifuncionais e auto-organizadas que se focam no DS e são capacitadas para o abordar por conta própria. Posto isto, é necessário clarificar o papel do gestor, este é responsável por garantir que as equipas têm as habilidades e recursos necessários para realizar o seu trabalho, garantindo que os prazos são cumpridos dando liberdade à equipa para tomar decisões.

Como já referido anteriormente, as metodologias ágeis são um conjunto de métodos com a sua própria estrutura para o DS, mas que seguem os princípios do manifesto agile. Apesar de existirem em grande escala alguns métodos são mais usados e referenciados do que outros, nessa lista encontramos: Cryst, Dynamic Software Development Method, Lean, Scrum e XP. De seguida serão apenas abordados os métodos scrum e XP uma vez que será fundamental um maior entendimento dos mesmos para o trabalho futuro.

Scrum e XP são bastante diferentes no que toca à análise e desenvolvimento do projeto, o primeiro é focado na mudança, caracterizando-se pelas etapas de desenvolvimento conhecidas como sprints, Jeff Sutherland afirma que "*Scrum embraces uncertainty and creativity. It places a structure around the learning process, enabling teams to assess both what they've created and, just as important, how they created it. The Scrum framework harnesses how teams actually work and gives them the tools to self-organize and rapidly improve both speed and quality of work.*" (Tracy, Frog, e Power 2014, 14). O segundo procura satisfazer as necessidade do cliente, enfatizando os valores da comunicação, simplicidade e feedback, na visão de Kent Beck "*XP is a style of software development focusing on excellent application*

of programming techniques, clear communication, and teamwork which allows us to accomplish things we previously could not even imagine. (...) XP is a path of improvement to excellence for people coming together to develop software." (Beck 2004, 25).

As práticas de Scrum resumem-se em curtas interações denominadas de sprints. Após o cliente especificar o produto é realizado um sprint de planeamento do projeto, de seguida são realizados vários sprints - com duração entre uma a quatro semanas -, e todos os dias existe uma pequena reunião onde os membros da equipa analisam o que foi feito definindo as tarefas para o dia seguinte, estas reuniões apelidam-se de daily scrum. No fim de cada sprint existe uma revisão do mesmo que é analisado com o cliente para que este dê o seu feedback, seguindo-se de uma sprint de retrospectiva sobre o decorrido até então. Na figura 1 podemos ver o framework scrum.

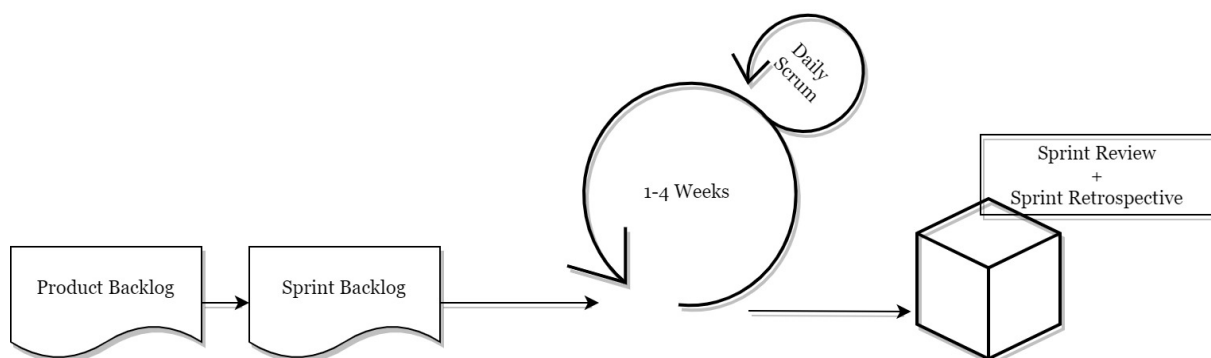


Figura 4 – Framework Scrum

Fonte: Autoria própria adaptada de

<https://www.scrum.org/resources/scrum-framework-poster>

O método XP sugere doze práticas recorrentes agrupadas em quatro grupos subdivididos. Feedback – desenvolvimento orientado para testes de unidade; planeamento através de uma reunião que ocorre no início de cada ciclo de iteração; o cliente deve estar presente durante todo o processo de desenvolvimento; programação a pares, enquanto que um se programador se concentra na escrita do código, o outro supervisiona -, processo contínuo – através da comunicação constante existem elevados nível de partilha e reutilização do código; refatoração do código foca-se na remoção da redundância do código; pequenos lançamentos -, entendimento do código – design simples; código coletivo onde toda a equipa pode

supervisionar e atualizar o código; seguir-se por padrões de codificação -, condições de trabalho – 40 horas semanais. Na figura 2 mostra-se o framework do método XP.

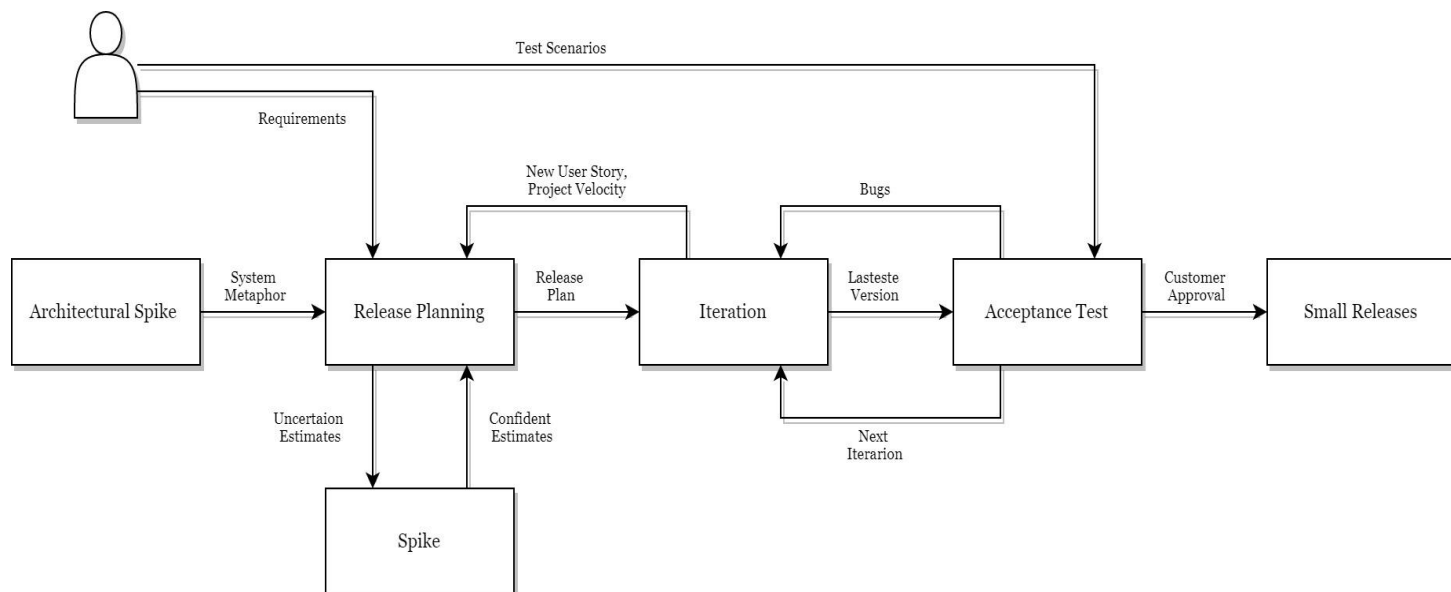


Figura 5 – Framework XP

Fonte: Autoria própria adaptada de

<http://www.extremeprogramming.org/map/project.html>

Em comparação, os dois métodos funcionam iterativamente, porém com tempos e atividades diferentes e, conseqüentemente, são destinados a diferentes tipos de projetos. Scrum é mais indicado para equipes multifuncionais dentro de um ambiente de desenvolvimento de produto sem grande quantidade de trabalho, para Almseidin et al. (2015) o método Scrum é preferencialmente utilizado em grandes organizações que requerem mais tempo e recursos, enquanto que o XP é mais adequado a organizações e projetos curtos que precisem de ser realizados rapidamente, contudo, antes de escolher XP, deve considerar-se uma série de fatores: os requisitos mudam dinamicamente, projetos arriscados, equipes pequenas, tecnologias usadas devem suportar testes automatizados e a participação constante do cliente.

Independentemente do framework escolhido, as metodologias ágeis têm como grande foco a comunicação com o cliente com o objetivo de alcançar a satisfação do cliente. A maioria das definições de satisfação do cliente em DS concentra-se em fatores como: requisitos do cliente, custos, design e qualidade. Contudo, durante o DS o cliente e a empresa criam um relacionamento que pode ser definido com o nível de satisfação do cliente em relação ao

processo adotado pela empresa durante todo o projeto. Principalmente em ODS que adotam metodologias ágeis a comunicação com os clientes é feita regularmente, influenciando diretamente o desenvolvimento do produto e o grau de expectativa por parte do cliente para o mesmo, através da identificação e validação constante dos requisitos para o software, estimar os seus custos, monitorar progressos, cumprir datas e fornecer garantias de qualidade, uma vez que o principal objetivo é entregar um produto que satisfaça as necessidades apresentadas pelos clientes, dentro dos prazos estipulados e dentro do orçamento, sem que o produto apresente qualquer defeito.

Segundo Grisham e Perry (2005), os defensores destas metodologias reconhecem a dificuldade de construir uma solução complexa, mas afirmam que um processo onde são entregues constantemente pequenas unidades garante uma maior qualidade no produto e na satisfação dos clientes. O autor reconhece os benefícios identificados, porém afirma que esta prática pode enfraquecer a relação com os clientes, apresentando lacunas de satisfação que devem ser geridas, como: gerir as expectativas do cliente com as experiências do cliente; gerir as expectativas do cliente e a percepção da empresa em relação às expectativas dos clientes; gerir as experiências do cliente e a percepção da empresa sobre as experiências dos clientes.

Para Flora e Chande (2014), a comunicação surge aqui como um fator chave para o sucesso de qualquer projeto, quando é realizado um trabalho colaborativo, onde a empresa envolve os clientes em cada fase através de pequenas apresentações, demonstrações de testes ou reuniões, descartando a ideia da entrega de documentação exaustiva e massuda. Apesar desta última ideia apresentada da não entrega de documentação, o autor refere que a falta de documentação é um problema constante em projetos de DS, apresentando-a como “... a big disadvantage for the developer. Since internal design changes frequently based upon users changing requirements after every iteration, it is not possible to maintain the detail documentation of design and implementation because of project deadline.” (Flora and Chande 2014, 11). A qualidade da relação entre o cliente e a empresa vai ditar o sucesso ou insucesso do projeto, esta realidade fez aumentar a procura de melhoria contínua por relações mais estáveis por parte das empresas para que pudessem proporcionar aos seus clientes experiências de excelência em todas as iterações.

A relação com o cliente descreve as maneiras como uma empresa se envolve com os mesmos, incluindo respostas imediatas a questões colocadas pelos clientes e o acompanhamento a longo prazo, mesmo após a entrega do produto, criando um relacionamento benéfico entre ambos.

2.2.2 Processo de Gestão Documental no Desenvolvimento de Software

Um documento, em ES, é algo que descreve o produto que está a ser desenvolvido, fornecendo as instruções necessárias para a sua utilização, não implicando apenas documentos e manuais relacionados com o produto, mas sim todo o material relacionado com o seu desenvolvimento. A prática de documentar ainda não se tornou comum nem regular na área uma vez que ainda não há provas dos seus benefícios, e com o argumento das metodologias ágeis de interação, cooperação e colaboração durante o DS em vez da comunicação por documentos formais, esta prática acaba muitas vezes por ser esquecida por não ser reconhecido o seu valor. Para refutar esta ideia, Clear (2009) apresenta duas razões principais: para comunicar e para entender. Os documentos produzidos devem apoiar o pensamento associado a cada fase do projeto, descrevendo e justificando as decisões tomadas que servirá de apoio a um maior entendimento do projeto. Desta forma a documentação não deverá ser algo isolado do DS, mas sim um resultado natural de cada fase do processo, tornando-se mesmo um requisito no DS como afirma Chomal e Saini (2015) na sua obra.

Ainda na sua obra, Chomal Saini (2015) afirmam que os documentos associados a um projeto de software devem ter uma série de funções, como: atuar como meio de comunicação entre os membros do projeto; ser olhados como um repositório de informações; ajudar a gestão a tomar decisões; ajudar os futuros utilizadores a usar o software. Van Loggem (2014) apresenta uma série de funções que a documentação deve desempenhar, como: facilitar a aprendizagem de colaboradores internos; consolidar a memória institucional; verificar a conformidade do planeado com o desenvolvido.

Porém, o seu propósito passa por tornar-se útil a todos os interessados, tanto durante o decorrer do projeto, futuros utilizadores do produto ou a outros desenvolvedores e/ou investigadores, como esclarece Hadar et al. (2013) "*One of its main purpose is to provide instruction about how the system is structured, how it works, and the design rationale that led to it.*" (Hadar et al. 2013, 1). Neste sentido a proposta de documentação em DS passa pela criação de diversos documentos, são:

Documentos do Processo: representa todos os documentos produzidos durante o desenvolvimento e manutenção do produto, geralmente são específicos para um determinado momento ou fase do processo

- Planos / Estimativas / Cronogramas geralmente criados antes do início de cada projeto para que os desenvolvedores se sigam pelos mesmos, podendo sofrer alterações
- Relatórios com a descrição de uso de recursos durante o projeto
- Papéis de trabalho onde são registradas as ideias

Documentos do Sistema: refletem-se em documentos que descrevem o próprio sistema

• Documentos de requisitos onde são fornecidas informações sobre as funcionalidades que o sistema	Informações sobre os participantes no projeto, assim como as suas funções e responsabilidades durante o mesmo
	Metas e objetivos que se pretende alcançar de uma forma resumida
	Quais as necessidades que levaram à construção do produto em questão e como essas ações se alinham com os objetivos da empresa
	Breve descrição das ações do cliente e dos resultados que pretendem alcançar com cada ação
	Crterios de aceitação por parte do utilizador para que seja alcança o sucesso com o produto
	Design da interação do utilizador
• Documentos com a descrição do design e experiência do utilizador	Personas do utilizador: criar possíveis utilizadores do produto, com diferentes perfis, tentando prever o seu comportamento, padrões de pensamento e motivação
	Cenários do utilizador: descrever as etapas que as diferentes personas executam ao realizar as diferentes tarefas específicas
	Mapa dos cenários: para compilar os diversos cenários criados

	Mapa da história do utilizador: para organizar todas as histórias do utilizador	
	Guias de estilo UX: incluir os padrões de design utilizados, assim como a justificação para essas escolhas, descrevendo todos os elementos de interface de utilizador possíveis e tipos de conteúdos usados, definindo regras para a sua utilização	
	Prototipagem:	Mapa do site: esquema visual que representa a conexão entre todas as páginas do produto
		Wireframes: esquemas que mostram como organizar a interface do futuro utilizador
		Mock-ups: fase seguinte à criação dos wireframes, onde é mostrada a aparência real do produto
		Protótipos ou maquete: criado para que as pessoas possam interagir sem ser necessária qualquer instalação
		Relatório de testes de usabilidade: documento onde são comunicados os resultados dos testes de usabilidade
Arquitetura do software é o documento com as especificações técnicas e principais decisões tomadas relacionadas com a arquitetura:	Incluir a visão geral do produto onde, resumidamente, se descreve os objetivos principais do projeto, os problemas que podem ocorrer, os resultados esperados e os princípios que foram seguidos	
	Detalhar a solução desenhada listando as funcionalidades que terá	
	Representar diagramaticamente a solução através de diagramas e/ou outros materiais gráficos que ajudam a compreender a sua estrutura	
Código-fonte:	Pode resumir-se a um documento que explica como o código funciona, embora não seja necessário e útil a futuros	

	utilizadores do código, podendo ser feito através de comentários ao longo do mesmo
	Se for criado um documento este pode incluir a estrutura do código, o tipo de ligações dos dados, medidas de segurança, padrões e princípios
Planos efetuados para garantir a qualidades assim como os testes realizados	Plano de gestão de qualidade definindo os padrões exigidos para garantir a qualidade do produto e descrever os métodos para atingir esse nível;
	Descrever uma abordagem para a realização de testes com as abordagens que serão realizadas para testar o software, incluindo recursos necessário para o feito;
	O plano de testes consiste em descrever o que deve ser testado e em que determinado momento listando os testes que devem ser efetuados para cada funcionalidade;
	Especificar os casos de testes onde cada funcionalidade é testada.
Guia de manutenção de ajuda: deve descrever quais os problemas encontrados e as soluções adaptadas, assim como as dependências de cada parte do projeto;	
Documentação de API's: tutoriais e guias a listas e especificar todas as API's usadas.	
Documentos do utilizador: são criados para o utilizador final se informar sobre o produto, podem ser destinados ao usuário final ou a outros desenvolvedores	
Documentos para o utilizador final deve explicar de maneira simples como o software funciona:	Tutoriais de imagens ou vídeos a explicar como o produto funcional
	Guias de informações sobre como localizar e resolver problemas
	Manuais de instruções de como instalar o produto
	Guia de funcionalidades onde explica os diferentes comportamentos do sistema

	Manual completo incluindo todas as informações e instruções sobre o produto
Documentação dos administrados do sistema que normalmente cobre a instalação e necessidades de atualizações que ajuda o administrador a realizar a manutenção do seu produto, por norma a documentação usada por outros desenvolvedores	Descrição funcional de forma técnica
	Explicar os diferentes comportamentos nos diferentes ambientes

Tabela 1 - Documentos propostos para o desenvolvimento de software

Com o aparecimento das metodologias ágeis as práticas de documentar foram sendo deixadas para trás, alguns desenvolvedores defendem que o código-fonte é suficiente para documentar. Contudo, a sua prática ainda é defendida por muitos autores, não como tradicionalmente onde resultava documentação exaustiva, mas a suficiente, para Maric et al. (2016), os documentos não devem ser longos e complexos mas sim concisos e compreensíveis, facilitando o seu uso para fins de comunicação, sendo uma prática ainda de elevado valor para as metodologias ágeis *"Documentation on agile projects is often seen as secondary to code and as a means to support communication"* tal como referido na obra de Kajko-Mattsson "Problems in agile trenches" (citado em Hoda, Noble, e Marshall 2012). Para colmatar a visão dos desenvolvedores de software com a necessidade de documentar foi adotado o lema "Documentar tanto quanto o necessário, o mínimo possível" ora fica dependente da necessidade de documentação de cada projeto e que cada cliente exigir, por norma, na maioria das vezes, são apenas criados documentos do utilizador.

No método Scrum a documentação baseia-se em backlogs do produto onde encontramos o produto descrito em pormenor, assim como a identificação de requisitos para o seu desenvolvimento, backlogs dos sprints para que fique registado todas as atividades realizadas em cada sprint, assim como os problemas e soluções encontradas, um relatório final detalhando o que foi feito e, quando requerido pelo cliente, um manual de utilizador.

Em XP os documentos dividem-se em documentos de requisitos realizado no início do projeto, um documento onde explicita o design do produto, assim como a justificações das escolhas realizadas para o mesmo, também o código fonte surge como um documento essencial nesta metodologia, no entanto ele deve estar devidamente estruturado e possuir anotações que simplifiquem a sua leitura, e no final um manual de utilizador. Todavia, para Cakir et al. (2018) independente do método usado para o DS a documentação tem de ser compreensível, bem estruturada e atualizada. Apesar das metodologias ágeis não exigirem uma documentação extensiva do projeto, a prática de documentar encontra-se presente durante todo o processo concedendo ao projeto documentação onde já se encontra todo o desenvolvimento do projeto explicado, assim como as decisões tomadas, o que irá facilitar a criação final de manuais de utilizador, ou outros documentos pedidos pelo cliente. A realidade que tem vindo a ser notória é a fraca GI, nomeadamente na falta de práticas de documentar ao longo do desenvolvimento do produto, e na gestão do conteúdo que está ainda aquém das suas possibilidades o que levam à perda de informação que pode ser benéfica para as empresas através da sua reutilização.

2.3 Ferramentas para a Gestão de Conteúdos no Desenvolvimento de Software

2.3.1 Ferramentas de Gestão de Projetos

A GP baseia-se na aplicação de conhecimentos, aptidões, ferramentas e técnicas às atividades de um projeto, otimizando-as e, em consequência, aumentar a eficiência do seu projeto. Juntamente com a ES, as suas práticas estão presentes em todo o processo de DS e o seu controlo. Mishra e Mishra (2013) defendem que gerir um projeto de DS pode ser extremamente complexo onde são usados diversos recursos e pessoas, dependendo da equipa e das organizações, os autores acrescentam ainda que a qualidade do produto irá depender da gestão do projeto ao longo do desenvolvimento do mesmo, "*The quality of a software product is dependent on the manner in which the project is completed, and if the software is designed in a hasty manner this will, in due course, create chaos in maintenance and future extensions.*" (Mishra e Mishra 2013, 1). Na atualidade existem imensas ferramentas que atuam na área de GP facilitando a sua prática, no entanto é necessário escolher a que melhor responde às necessidades apresentadas por cada empresa/organização uma vez que cada uma delas apresenta um conjunto de funcionalidades abrangentes.

No caso de ODS é necessário que apoie a partilha constante de conhecimento, principalmente quando são aplicadas metodologias ágeis, para que todos os membros das equipas consigam realizar o seu trabalho com o máximo de apoio. Porém, segundo Shen e Yu (2019), a partilha do conhecimento é ainda um desafio para as equipas de DS afetando a colaboração durante o projeto, citando vários autores que apresentam a mesma visão ao longo dos anos.

Neste sentido, e para uma GP devidamente apoiada surgem ferramentas, principalmente colaborativas, com as mais diversas funcionalidades para dar resposta a todas as práticas de GP aplicadas ao DS, que apoiam todas as atividades dos projetos. Ao longo do estudo sobre a utilidade das ferramentas específicas para realizar uma boa GP, foram identificadas diversas plataformas, contudo algumas foram mais mencionadas do que outras, isto acontece por dois motivos, ou por serem as mais utilizadas, ou por se destacarem em relação às restantes. Para além das plataformas mais mencionadas pelos autores foi também realizada uma pesquisa sobre as plataformas que possam ser consideradas concorrentes às identificadas, posto as plataformas recomendadas para a GP são as seguintes:

- JIRA (Atlassian 2021): software comercial desenvolvido pela empresa Atlassian, é uma ferramenta que permite acompanhar o desenrolar de tarefas apoiando todas as fases do projeto dedicado a equipas ágeis onde cada membro pode acompanhar o desenvolvimento do software num só lugar, e onde podem planear livremente o seu trabalho. É também possível que cada empresa escolha um fluxo de trabalho, ou que criem o seu próprio fluxo, desde o início do DS até ao seu envio, tendo ainda o benefício de ser possível integrar as ferramentas que a empresa já utilize, facilitando todo o trabalho das equipas.
- Redmine (Redmine 2014): um software de livre acesso podendo ser utilizado através da web dedicado à gestão de erros permitindo e uma gestão flexível dos projetos. Para além disso contém componentes que permitem representar visualmente os projetos na sua totalidade através do uso gráficos de Gantt, apresentando os prazos e atividades realizadas. É considerada uma multiplataforma uma vez que permite a integração de bases de dados e de diversos repositórios. Apesar de ser mais adequada para realizar a gestão de problemas e erros, do que para gestão de todo o tipo de projetos, é adequada para a realização de projetos relacionados com o DS, uma vez que a identificação de problemas e erros é uma prática comum,
- Trello (Trello 2021): através da utilização do paradigma Kanban, estratégia que permite otimizar o fluxo de trabalho, permite a organização do projeto através da representação dos projetos em quadros, que contém listas das várias tarefas a serem realizadas, e

cartões que representam cada tarefa. É indicada para equipas ágeis, uma vez que foi criada para permitir trabalhar com maior colaboração e, consequentemente, maior produtividade.

- Microsoft Project (Microsoft 2021a): um software de GP produzido pela Microsoft para que os seus utilizadores possam facilmente planear os seus projetos, quer sejam de grandes ou pequenas dimensões, e conseguir ter uma melhor visão sobre os resultados através da criação de quadros interativos que permitem ver o estado geral e detalhado dos projetos com a visualização do Power BI. As principais funcionalidades desta plataforma permitem o agendamento de todas as tarefas individuais e em grupo necessárias para a conclusão do projeto através da inserção de todas as informações sobre o projeto, assim como os recursos imprescindíveis para a sua conclusão atribuindo os recursos às pessoas e as tarefas específicas. Além disso permite um controlo dos custos realizando estimativas orçamentais e definir padrões de qualidade para garantir a qualidade do produto final.
- Monday (Monday 2021): uma plataforma colaborativa baseada na nuvem de diversas funcionalidades para que consiga dar resposta a todas as necessidades dos seus utilizadores indicada para a GP, através da personalização de fluxos de trabalho para executar projetos, processo e trabalho diário adequados a cada empresa. Apesar de ser conhecida como uma plataforma de gestão de tarefas os recursos que esta oferece, como automação, painéis de dados e integração com as principais ferramentas do mercado, permitem um planeamento flexível de qualquer projeto, independente da área em que estes se enquadrem.
- Asana (Asana 2021): desenhada para acompanhar todo o processo de desenvolvimento do projeto, focando na sua gestão, organização e controlo. A plataforma compromete-se a ajudar as empresas a gerir os seus projetos e tarefas de cada colaborador, tudo numa só ferramenta, adequada para as empresas que se dividem por projetos destacando uma equipa para cada um, dado que permite a gestão de equipas. As equipas podem criar os projetos, atribuir tarefas a colegas da equipa, especificar prazos e comunicarem dentro da plataforma. Para além disso, são disponibilizadas ferramentas de relatório, arquivo, mapeamento e calendário, entre outras.

Em suma, todas as plataformas identificadas podem ser utilizadas para a GP, porém as mais adequadas para cada empresa depende dos seus objetivos, uma vez que cada uma delas tem as suas especificidades, mas todas elas garantem o acompanhamento dos projetos.

2.3.2 Ferramentas para a Gestão de Conhecimento

Em ODS usa-se em grande escala ferramentas de colaboração "collaboration tools such as story-boards and virtual whiteboard enable a strong level of information exchange between team members." tal como referido na obra de Bjorson "Knowledge management in software process improvement" (citado em Khalil e Khalil 2020), no entanto o autor destaca a importância de ferramentas de TI "knowledge management initiatives necessitate a good IT Infrastructure." (Khalil e Khalil 2020, 12) na promoção de comunicação informacional e partilha de conhecimento codificado, tipicamente conhecidos como ferramentas de GC. Priya e Selwyn (2016) reconhecem a sua importância sendo utilizadas na captura de dados, análise da informação e disseminação do conhecimento, porém reconhecem que pequenas e médias empresas não têm capital para manter um sistema de GC, devido à relação custo/benefício que estes apresentam. Em contrapartida surgem abordagens e outros métodos de transferência de conhecimento, Razzak e Ahmed seguem a mesma linha de pensamento e apresentam também ferramentas colaborativas que são usadas em equipas ágeis, *"agile team members share knowledge through effective use of knowledge management tools like wiki, pairs-programming and video-conferencing."* (Razzak e Ahmed 2014, 2) para completar os autores explicam que em DS os membros das equipas partilham na sua maioria ideias, conhecimento tácito, em storyboards, série de ilustrações com um propósito, e whiteboards, tal como o próprio nome indica um quadro branco onde são expostas ideias.

À semelhança do que foi feito para identificar as ferramentas adequadas para a GP, inicialmente procedeu-se a um levantamento das ferramentas mais mencionadas pelos autores, seguindo-se de uma análise do mercado para identificar possíveis concorrentes. Posto isto, as ferramentas usadas para a GC em ODS com maior destaque baseiam-se em:

- Plataformas wiki (Wikipédia 2021) permitem aos utilizadores modificarem colaborativamente o seu conteúdo, não sendo necessária revisão para que essas modificações sejam aceites, com recursos para a gc, de websites de comunidades e adequado para intranets. A estrutura das wikis não são muito complexas, o que permite aos seus utilizadores melhorarem a mesma de acordo com as suas necessidades, através da adoção de certas funcionalidades como controlos de acesso, adição de páginas e edição colaborativa. Estas funcionalidade são gerais para todas as wikis, contudo estas podem ser usadas autonomamente ou como partes de outros sistemas, podendo ainda ser públicas ou privadas, o que permite complementar as wikis com outras características. As wikis são conhecidas por serem usadas para encontrar informações ou para contribuir com informações, favorecendo a partilha de conteúdo, que pode ser bastante útil dentro de um contexto empresarial para: integração de novos funcionário

para que estes encontrei o material necessário mais facilmente; para acompanhar todos os processos dentro de uma organização; para serem consultados sempre que necessário fornecendo ajuda; ser utilizado como um arquivo interno e por fim consolidar todo o CO.

- Microsoft SharePoint (Microsoft 2021b) é uma plataforma de aplicações web disponibilizada pela Microsoft para a GCC que pode ser acessível através de portais e intranets empresariais. O SharePoint está desenhado para incentivar o trabalho em equipa de forma dinâmica e produtiva, através da personalização dos sites para simplificar a partilha de ficheiros, dados, notícias e recursos, sem esforço e de forma segura entre os membros de uma empresa. A plataforma é também conhecida por possuir formas inteligentes de descobrir informações e experiências que ajudam na tomada de decisões, aumentar a produtividade ao transformar os processos internos desde as mais simples tarefas até fluxos de trabalho completos, uma vez que, apesar de ser associada à GCC, abrange diversas áreas de prestação de serviços.
- KnowledgeBase Manager Pro (KnowledgeBase Manager Pro 2017) utilizada para a GC e de conhecimento empresarial de forma automatizada. Com o uso desta ferramenta é possível: melhorar o atendimento ao cliente, que ao receber informações sobre os clientes e o seu feedback toma as medidas adequadas; reduzir os custos associados ao suporte, uma vez que facilita à equipa a tarefa de dar suporte; maior rapidez ao encontrar as informações necessárias; tomar decisões de forma informada e objetiva; aumento da eficiência individual de cada funcionário evitando a redundância no seu trabalho; retenção do conhecimento individual transformando-o em CO. Esta plataforma simplifica toda a documentação e o processo de criação e partilha da mesma, apoiada no aprimoramento dos processos organizacionais de criação, armazenamento, recuperação e partilha do conhecimento. Comumente é utilizada para complementar um help desk ou para partilhar informações entre os colaboradores de uma organização.
- O GitHub (GitHub 2021) é usado como plataforma de hospedagem de código-fonte e arquivo para o controlo de versões de software, em DS é usada para criar e manter os seus trabalhos. Ela permite que todos os seus utilizadores contribuam para qualquer projeto, rastreamento de problemas e fácil comunicação entre todos os intervenientes. Apesar de ser uma plataforma dedicada à escrita de código possui também funcionalidades que permitem oferecer suporte ao DS, garantir a organização do mesmo e, quando terminado, o seu armazenamento. Para além destas características permite automatizar os fluxos de trabalho de DS e integração de documentos que acompanham esse mesmo desenvolvimento.

Porém, muito do conhecimento é compartilhado no dia-a-dia através de programação a pares, reuniões diárias, storyboards para organizações gráficas com sequências de ilustrações, virtual whiteboards para organização das tarefas de forma simplificada e visual, salas de chats comuns, fóruns e conferências online fonte de conhecimento para os desenvolvedores de software, e devem ser feitos todos os esforços para que esse conhecimento tácito se torne em conhecimento explícito. Muitas ferramentas mencionadas no ponto anterior, de GP, são apoiadas por mecanismos que favorecem a GCC.

2.3.3 Ferramentas de Comunicação

Como visto anteriormente a comunicação num ambiente de DS é fulcral para o bom desenrolar do projeto, alguns estudos identificam a dificuldade de partilhar o conhecimento em projetos que seguem metodologias ágeis devido à falta de comunicação entre os membros das equipas. Estas empresas são munidas de práticas que incentivam a comunicação, como as reuniões diárias (comum no método Scrum) e na programação a pares (comum no método XP), para além destas que se encontram identificadas anteriormente existem outros meios como videoconferências e fóruns online de discussão, que para além de serem um grande meio para a partilha de conhecimento são também um meio de comunicação informal eficaz. Razzak e Ahmed (2014) enfatizam esta prática *"In the agile collocated development, informal communication is the key enabler for knowledge sharing but when an agile project is distributed, informal communication and knowledge sharing is a challenge due to low communication bandwidth as well a social and culture distance."* (Razzak e Ahmed 2014, 2), concluindo o seu estudo com as chaves para promover a partilha do conhecimento entre os membros das empresas sendo elas a comunicação, a coordenação e a colaboração.

Para satisfazer esta necessidade surgem ferramentas que garante suporte à comunicação durante todo o desenvolvimento do projeto como o Slack, que inclui salas de chat privados e em grupos, recursos de arquivo, e permitindo ainda a sua integração noutros softwares mais indicados para as ferramentas de GP e GC. Neste contexto surgem as redes sociais como aplicações favoráveis para a comunicação, para a transferência informal do conhecimento, Priya e Selwyn (2016) afirmam que facilitam os comportamentos humanos e da sua interconexão, uma vez que vivemos numa realidade em que as redes sociais estão cada vez mais presentes nos nossos dias controlando as nossas interações. Noutro ponto oposto temos a comunicação com o cliente, que embora não possa exibir o grau de informalidade da comunicação interna, carece de abertura e facilidade para que tanto os clientes como a empresa

possam expressar as suas ideias, debater o desenrolar do projeto, e no apoio posterior que será dado.

Tendo em conta o baixo orçamento disponibilizado por pequenas e médias empresas surgem ferramentas que podem ser integradas em sistemas de GP que apoiam a comunicação com o cliente. Um exemplo bastante usado e referido é o sistema de tickets que se predispõem a auxiliar na solução de problemas de atendimento ao cliente fornecendo-lhes um serviço proativo e imediato, e à empresa garante uma gestão dos contactos dos clientes facilitada o que irá facilitar o desempenho dos seus membros na resposta às questões com os clientes. É também de referir que o webmail é uma solução usada em grande escala, Sourì, Asghari e Rezaei (2017) refere por diversas vezes a sua integração nos mais variados sistemas existentes para o apoio da comunicação com os clientes, independentemente da extensão do webmail, que permitem uma organização minimalista dos contactos e do registo de cada interação com os mesmos.

A comunicação eficaz com os clientes e colaboradores irá beneficiar a GI. Através de cada interação é transferida informação, como as necessidades que os clientes apresentam que, por sua vez, será essencial para o desenrolar dos projetos, os entregáveis intermédios ao qual o cliente dá o seu feedback, e é requisitado e fornecido o apoio de manutenção que as empresas oferecem após a entrega do produto final. A adoção de práticas e ferramentas adequadas para a gestão de toda essa informação é essencial, não só para estabelecer uma boa relação com os clientes e colaboradores, mas também para melhorar a sua prática internamente para que as empresas tenham as informações devidamente estruturadas e organizadas.

3 Metodologia

Os objetivos definidos para o desenvolvimento da presente dissertação têm uma só finalidade, a especificação de requisitos para o desenho do processo de GI para componentes de software. A Engenharia de Requisitos (ER) é um processo colaborativo e interativo que pretende garantir a explicitação de requisitos relevantes com o detalhe exigido através de acordos entre as partes interessadas, nomeadamente o seu desenvolvedor e o futuro utilizador, por fim a validação dos requisitos identificados assim como a devida documentação.

Neste sentido, foi escolhida a metodologias de ER concebida por Klaus Pohl¹, em 2010. O autor defende que o processo de ER começa quando existe a necessidade de mudar a realidade, acrescentando que essa mudança deve ser definida de forma breve e precisa. Para isso é necessário entender o contexto da criação de um sistema e as necessidades do futuro usuário do mesmo. Esta abordagem metodológica, apesar de ser dedicada ao desenvolvimento de um sistema, irá permitir desenvolver o processo em questão com base nas necessidades identificadas com uma validação e gestão dos mesmos através do contacto constante com a empresa, vai mais além, pois os seus fundamentos garantem que todos os passos sejam devidamente documentados. Os elementos da metodologia de Pohl baseiam-se no: contexto do sistema, nas suas atividades principais, nos artefactos dos requisitos e em duas atividades transversais de apoio que influenciam em grande escala o processo de ER garantindo os resultados esperados, são elas a validação e a gestão dos artefactos dos requisitos, das atividades principais e do contexto do sistema.

De seguida será analisado elemento a elemento comparando a parte teórica com a parte prática aplicada durante o desenvolvimento da presente dissertação. A metodologia completa encontra-se no [anexo A](#).

Para contextualizar a criação do sistema na sua plenitude existem várias partes relevantes que devem ser consideradas, como o porquê do seu desenvolvimento, a sua usabilidade e quem irá usufruir do mesmo, o hardware e software que será necessário e por fim desenhar o seu processo de desenvolvimento. Formando assim as quatro facetas do contexto apresentadas na figura abaixo, assim como a sua explicação, abaixo da figura podemos ver a aplicabilidade de cada faceta do contexto.

¹ Pohl, Klaus. 2010. “Requirements Engineering by Klaus Pohl.” Springer.

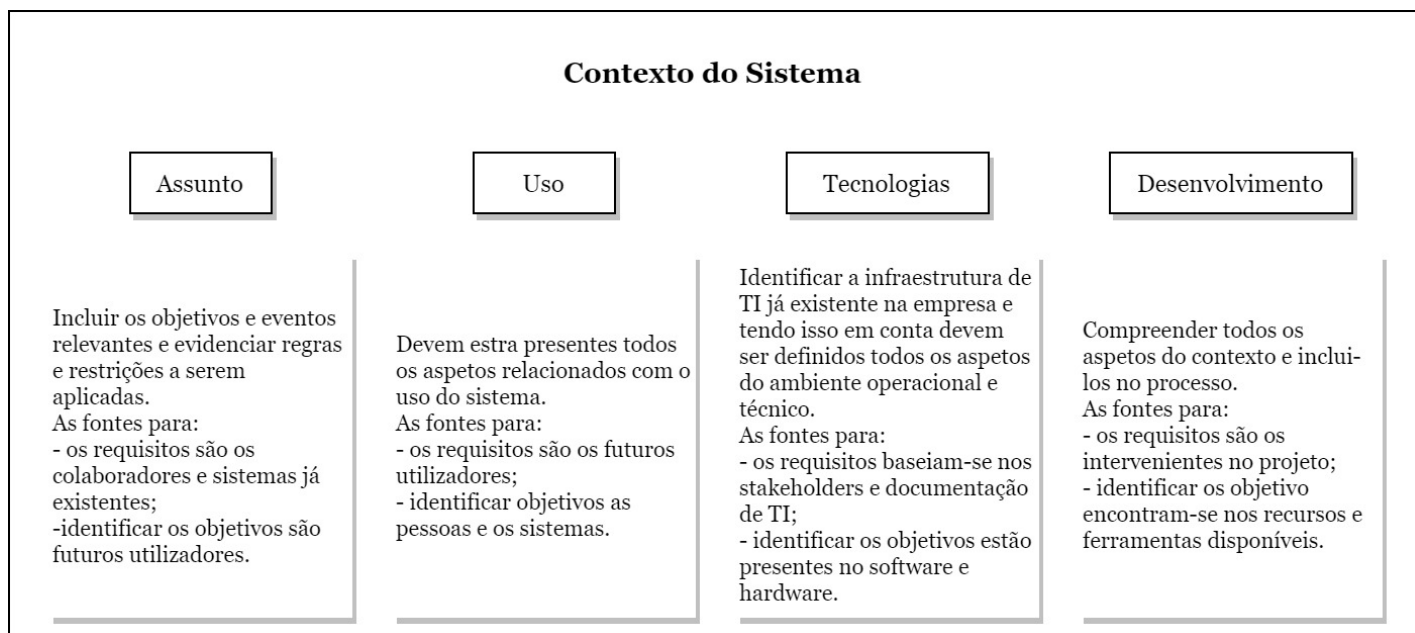


Figura 6 – Engenharia de Requisitos: Contexto do Sistema

- Faceta do assunto: apresentação da necessidade que leva à criação do processo, quais os objetivos intermédios e finais, assim como os eventos relevantes durante o projeto, como por exemplo quais e quando os documentos que descrevem o desenvolvimento do software devem ser criados. Para além disso, devem ser evidenciadas regras que regulam o registo e atualizações desses mesmos documentos, assim como restrições de acesso. Os documentos já existentes foram fonte para identificar os requisitos, já os colaboradores da empresa contribuíram como fonte tanto para identificar os requisitos como os objetivos do processo.
- Faceta do uso: distinguir os intervenientes que ficaram responsáveis por cada parte do processo e as atividades a serem abrangidas por cada fase. Tanto os requisitos como os objetivos foram identificados através dos futuros utilizadores do sistema.
- Faceta das tecnologias: foi identificada a infraestrutura existente na empresa, os sistemas e plataformas usadas, redes de comunicação, entre outros, com a finalidade de perceber se seria possível desenvolver um processo onde apenas seriam utilizadas essas ferramentas. Tendo isso em conta, o processo desenhado conta com a utilização da maioria das TI disponíveis, sendo recomendadas alterações quando estritamente necessário. As fontes para identificar os requisitos tecnológicos basearam-se nos stakeholders e na documentação sobre a infraestrutura e arquitetura de TI existente, já os objetivos foram identificados através do hardware e software.

- **Faceta do desenvolvimento:** após identificar as necessidades, quem irá usufruir da solução e as tecnologias farão parte da mesma, a solução foi pensada e desenhada tendo todas essas variáveis em consideração. A definição dos requisitos na presente faceta foi gerida pelos intervenientes no processo enquanto os objetivos foram encontrados nos recursos e ferramentas disponibilizadas para o desenvolvimento do processo.

Apesar de representarem diferentes pontos necessários para a contextualização do sistema, as quatro facetas não funcionam independentemente umas das outras, elas interligam-se. A faceta do assunto liga-se às tecnologias para representar a informação real, um exemplo concreto é a transposição do nome do cliente para o formato digital, consequentemente a faceta das tecnologias liga-se à usabilidade uma vez que representa a informação consoante a necessidades dos seus utilizadores, por fim, na faceta do uso, o utilizador interpreta a informação associando-a à realidade ligando a presente faceta à primeira referida, a faceta do assunto. Não afastando a faceta do desenvolvimento uma vez que se relaciona com as outras três facetas relacionando-se com todas durante o seu desenvolvimento.

Seguidamente, na figura 7, podemos ver que se encontram as três atividades principais de documentação, de negociação e de elicitação, e o que é esperado de cada uma delas na teoria, abaixo da mesma é descrita a sua serventia na realidade.

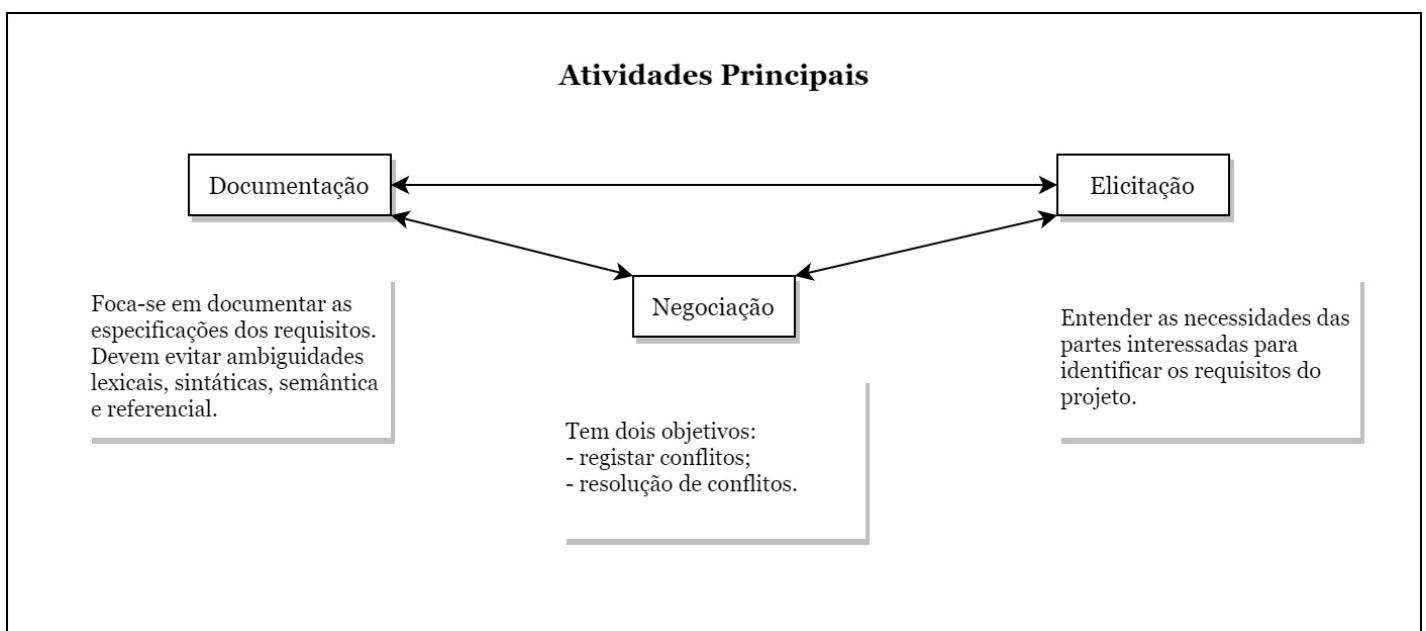


Figura 7 – Engenharia de Requisitos: Atividades Principais

- Atividade de documentar: apresentar os requisitos especificados, evitando a ambiguidade lexical que pode surgir pelo uso inadequado de sinónimos e homónimos, evitar a ambiguidade sintática referida quando existe uma ambiguidade estrutural ou quando existem duas árvores de sintaxe válidas que possam ser atribuídas à mesmas frase, evitar a ambiguidade semântica que ocorre quando uma frase pode ter mais do que uma interpretação e, por fim, evitar a ambiguidade referencial que acontece quando uma palavra numa frase pode gerar diferentes interpretações.
- Atividade de negociar: identificar problemas que surgiram e como foram resolvidos.
- Atividade de elicitação: depois de identificar os requisitos, estes necessitam de ser analisados e entendidos, no início apresentaram-se vagos, mas através de reuniões informais foram sendo aprimorados.

Para completar, a metodologia requer a criação de artefactos de requisitos, todos eles interligados, onde os requisitos identificados serão devidamente documentados, incluindo quais os objetivos dos mesmos, a descrição dos cenários de aplicação e as soluções encontradas. À semelhança dos elementos anteriores na figura abaixo podemos ver a descrição da metodologia na teoria, seguindo-se de como foi aplicada no presente contexto.

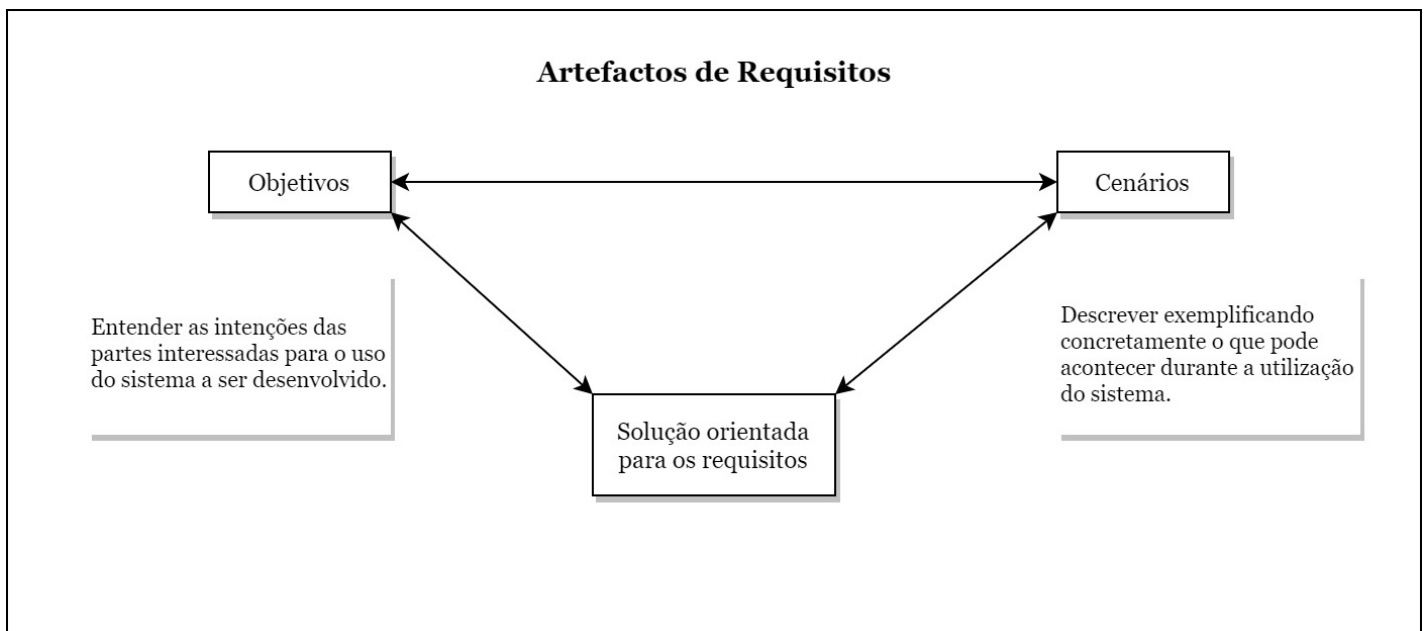


Figura 8 – Engenharia de Requisitos – Artefactos de Requisitos

- Artefacto de objetivo: que objetivos são pretendidos alcançar com a criação de um processo para a criação de documentação durante o DS.
- Artefacto de cenários: representação gráfica do processo, para que esteja devidamente descrito e exemplificado.
- Solução orientada para os requisitos: os dois artefactos descritos anteriormente fornecem uma base para a definição e elaboração de requisitos orientados para a solução, uma vez que incorpora soluções concretas num contexto de utilização que pode levar à identificação, revisão e/ou refinamento de objetivos e cenários. As soluções documentadas devem satisfazer os objetivos e os cenários.

Por fim, esta metodologia conta também com duas atividades transversais, a validação e a gestão. A validação liga-se com o contexto do sistema verificando se o contexto é considerado durante o processo de ER, com as atividades principais verificando a conformidade entre as atividades realizadas e as especificações da atividade previamente definidas, com os artefactos dos requisitos tendo o objetivo de identificar os defeitos dos requisitos. A gestão tem objetivo observar o contexto do sistema e identificar mudanças relevantes que ocorram e possam influenciar o mesmo, com o planeamento das atividades principais controlando-as durante a ER com a finalidade de assegurar um processo eficiente e eficaz, e identificar os artefactos de requisitos necessários ao longo de todo o ciclo de vida do sistema incluindo a priorização do registo persistente dos requisitos, a gestão da configuração e a gestão de mudança dos requisitos, assim como a rastreabilidade e manutenção.

Em suma, podemos afirmar que na faceta do contexto foram identificados os principais documentos a serem criados – documentos do sistema, processo e de utilizador-, quais as plataformas a sere usadas -continuação da utilização da plataforma redmine -, quais os intervenientes e todo o processo para a sua elaboração. A faceta das atividades permitiu identificar as principais necessidades da empresa através de contactos informais com os seus colaboradores. A última faceta que diz respeito aos artefactos baseiam-se no presente documentos. As atividades transversais permitiram uma gestão dos objetivos e expectativas, assim como a validação constante dos mesmos.

4 A Gestão de informação no desenvolvimento de módulos de software

4.1 Caso de Estudo: Skrey Software

A Skrey marca presença no mercado nacional e internacional contando com um vasto leque de clientes, que vão desde empresas a particulares, como tal, os seus produtos são concebidos tanto para empresas como para particulares, Business to Business (B2B) e Business to Consumer (B2C) respetivamente. No primeiro, como o próprio nome indica, os produtos e/ou serviços são desenhados para outras empresas, é um tipo de negócio que tem vindo a crescer e, consequentemente, faz crescer a criação de estratégias específicas que otimizam a sua adequação ao mercado. Quanto ao segundo tipo de negócio referido, nomeadamente B2C, este tem como público-alvo o consumidor final do produto e/ou serviço, o que fará que em qualquer negociação varie o cliente e os objetivos do mesmo. Independentemente do tipo de cliente, sejam empresas ou particulares, para quem sejam desenhadas as soluções, elas são idealizadas e projetadas para que seja possível a sua interligação com outros sistemas, sendo as mais comuns plataformas de e-commerce, de gestão empresarial e de gestão de clientes, porém não descora de outras opções. Para além disso, a Skrey, fornece hosting especificamente projetado e otimizado para sites de comércio digital, contando com datacenters em sete países diferentes garantindo uma cobertura global, através de servidores de alto desempenho a nível empresarial.

Durante o decorrer dos projetos a empresa segue um processo bem definido onde inclui uma fase de análise, seguindo-se de uma fase dedicada ao desenvolvimento e por fim uma fase de operações. A primeira fase de análise é fundamental para conhecer o cliente e as suas necessidades com o objetivo de ser possível dar continuidade ao trabalho, ainda durante a primeira fase e após o primeiro ponto realizado é desenhada uma solução à medida do cliente. Na fase de desenvolvimento é implementada a solução tecnológica previamente desenhada, usando práticas e metodologias de DS que garantam um bom funcionamento dos mesmos, e garantindo a sua compatibilidade com outros sistemas, ainda nesta fase é desenvolvida uma fase de testes finais para além dos testes unitários ao longo do projeto. Por fim, a solução é colocada online de forma planeada e controlada, onde a empresa garante o apoio total durante o processo, e uma melhoria contínua da plataforma e do modelo de negócio.

Um software de grande complexidade e dimensão, como são categorizadas as soluções criadas à medida dos clientes, é dividido em diversos componentes que garantem que este contém múltiplas funcionalidades. Na Skrey, módulos também são construídos durante a conceção de

um software de maior dimensão em contexto de um projeto específico com um cliente. Como tal, o termo módulo, em ES, é comumente usado para descrever um elemento integrado num software que segrega uma série de funcionalidades, contudo é desenvolvido como uma unidade independente que pode ser reutilizada em diversos sistemas mais complexos. O propósito desta divisão está diretamente relacionado com a facilidade de DS, sendo mais rentável em tempo, e com a possibilidade de reutilização dos módulos. O módulo de um sistema é visto como parte do mesmo que tem responsabilidades sobre determinadas tarefas pelo qual foi desenhado para projetar, podendo ser empregue em diversos sistemas desde que a arquitetura tecnológica seja equivalente, não havendo consequência nem na sua inclusão nem exclusão. O objetivo de cada módulo é fornecer recursos específicos para um produto, implementando novas funcionalidades ampliando os recursos disponíveis para o negócio em questão para que sejam desenvolvidos ou integrados.

Ao desenharem a solução ideal para o cliente esta é dividida pelos diversos componentes que lhe serão agregados para que possua todas as funcionalidades requeridas pelos clientes, sendo desenvolvidos em separado. Cada um passa pelo mesmo caminho, existe uma especificação de requisitos, o seu desenvolvimento sujeito a testes constantes garantindo que todas as funcionalidades estão operacionais e que está apto para integração, e depois de ser aprovado é criado um manual de utilizador para que os clientes e futuros clientes possam consultar quando desejarem utilizá-lo. Este trabalho não é descartável visto que os módulos podem ser reutilizados em diversos projetos, tal como são concebidos no início ou sofrendo alterações.

Em primeiro lugar são identificados os requisitos funcionais e não funcionais após uma pesquisa aprofundada sobre o que já existe, que funcionalidades são indispensáveis e o que podem ter de novo, resultando num documento inicial, onde é incluída uma pequena descrição do módulo e os requisitos identificados, não sendo mandatária a sua atualização caso os requisitos mudem. Após esta tarefa concluída segue-se a fase de escrita do código onde não é elaborado qualquer documento. No final da escrita do código o módulo é submetido a testes iniciando-se a fase de testes que até então não incluía nenhuma prática para que os mesmos fossem documentados, porém está a ser iniciada através de sensibilização para a necessidade da sua prática. Por fim, é elaborado um manual de utilizador onde um colaborador da empresa se dedica à sua redação. Na prática, estes componentes são criados e armazenados e apenas disponibilizados quando requerido pelo cliente, fazendo com que o processo de organização e recuperação dos mesmos seja mais cansativo e demorado, levando à perda de tempo por parte dos membros da empresa.

Após a sua conceção e devido armazenamento dos componentes, a Skrey tem em vista dá-los a conhecer a atuais e potenciais clientes e colaboradores de forma prática e eficaz através da

utilização da documentação criada para que estejam sempre a par de novos produtos e serviços que sejam desenvolvidos assim como das atualizações feitas aos já existentes. Para isso, procura criar toda a documentação necessária, que esta esteja devidamente completa e que seja redigida a tempo adequados, assim como mantê-la atualizada a parte das atualizações dos módulos através da adoção das melhores soluções que garantam que todas estas questões sejam respeitadas e que se adequem à sua realidade, não olhando apenas para a vertente tecnológica, mas também tendo em conta os recursos disponíveis. Não descorado que procura comunicar com futuros clientes e colaboradores através da disponibilização de alguma documentação mostrando assim o trabalho que é realizado e os produtos e soluções que disponibilizam, com essa finalidade debruçam-se sobre a qualidade da informação que é disponibilizada, tendo esta de ser legível e de fácil compreensão para todos os que procurem saber mais sobre os produtos e serviços que disponibilizem, sem que seja necessário terem elevada literacia sobre a área.

4.2 Caracterização do processo de desenvolvimento dos módulos de software

Antes de analisar o processo de desenvolvimento dos módulos é necessário ter em conta que estes são idealizados e concebidos durante cada projeto como referido anteriormente. O processo completo pode ser consultado no [anexo B](#), contudo apresenta algumas modificações relativamente aos processos apresentados abaixo, uma vez que apresenta dos o processo e todas as interações entre as diversas fases de DM e os que são de seguida apresentados são específicos de cada fase.

O primeiro passo dado para a criação dos módulos é a especificação dos requisitos que, numa primeira instância, e se for o caso, são fornecidos pelos clientes seguindo-se de uma pesquisa exploratória que tem como objetivo identificar os produtos já existentes no mercado com a finalidade de constatar que inovações podem incluir no seu produto trazendo vantagens para o mesmo, e também quais as funcionalidades base imprescindíveis para quem procura cada produto específico. Com isto, pretendem garantir que o produto final contenha todas as funções desejadas pelos clientes, acrescentando-lhes valor por ser considerado inovador. Os resultados desta tarefa são visíveis num documento, estando ao encargo de uma pessoa destacada para o fazer, onde podemos encontrar uma pequena e introdutória descrição do módulo, os requisitos funcionais e não funcionais anteriormente identificados, por norma encontra-se na página wiki do Redmine, plataforma usada para gerir os projetos.

O passo seguinte é a validação. Em primeiro é sujeito a uma validação interna onde os requisitos são discutidos e defendidos por quem os apresenta, podendo surgir alterações para que o módulo se adeque melhor à realidade do mercado. Quando o módulo é incluído num projeto específico com um cliente, o documento de requisitos é enviado aos clientes para que este o analise e indique se está de acordo com o seu objetivo inicial ou se é necessário sofrer alterações, quando o resultado desta validação é o segundo referido (sujeitar o documento a alterações), é necessário que se volte à fase de identificação de requisitos. Quando está tudo em conformidade e o cliente dá o seu aval passam então à fase de desenvolvimento, contudo, os requisitos podem ainda sofrer alterações durante a próxima fase. Na figura abaixo podemos ver o processo de identificação de requisitos.

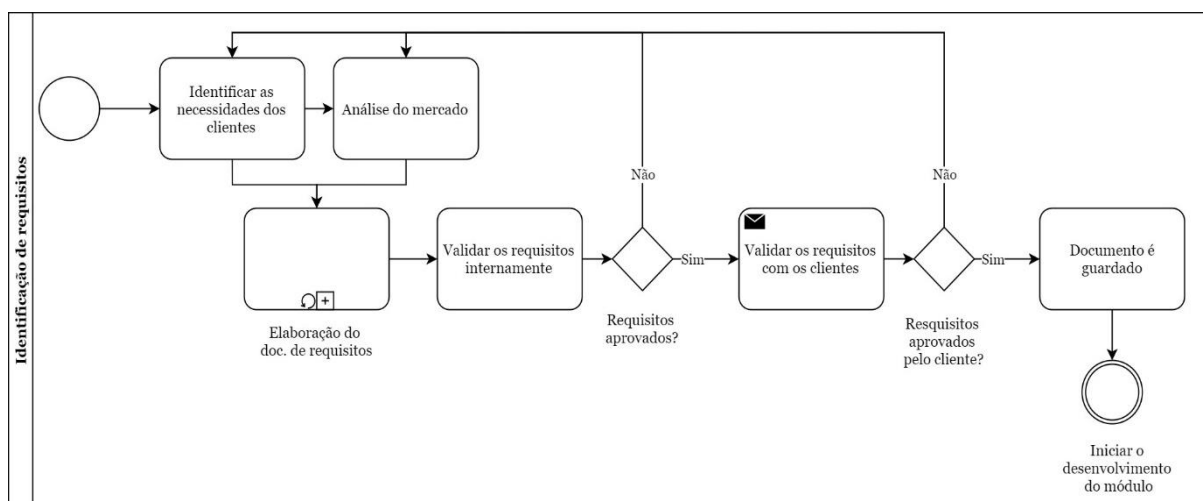


Figura 9 – Processo de identificação de requisitos

A prática recorrente é passar da especificação de requisitos para a escrita do código num editor de código-fonte, onde os módulos são baseados no requisitos identificados anteriormente mas também a pensar na plataforma onde serão inseridos, neste caso a plataforma Magento uma plataforma de e-commerce, destinada a esse feito, querendo isto dizer que existem certas funcionalidades e certos padrões já exigidos pelo Magento que devem ser seguidos para que o módulo possa ser comercializado nessa plataforma. De modo a organizar este procedimento o módulo é dividido por funcionalidades e cada uma é constantemente submetida a testes funcionais e de integração durante a sua evolução, garantindo que todas as funcionalidades operam de forma esperada e que é possível a sua integração. Contudo, são frequentemente usadas API's (Application Programming Interface), por definição um conjunto de padrões e

protocolos usados para a integração de aplicações de software, que permite que a solução desenhada comunique com outros produtos sem precisar que estes façam parte da solução. Quando recorrem ao uso de uma API é necessário que a mesma seja documentada e com essa finalidade recorrem ao Swagger, uma linguagem própria para a descrição de interfaces que com recurso a um conjunto de ferramentas de código aberto que ajudam a projetar e a documentar API's através dos comentários inseridos ao longo do código do módulo. No final esse documento gerado automaticamente é disponibilizado ao cliente através da plataforma Magento.

É também importante referir que é usado o GitLab como repositório de software onde é guardado o código-fonte do módulo, e o Nexus para as versões preparadas que efetivamente seguem para os clientes, e onde é feita a gestão de todas as versões de cada módulo. É esperado que o output se traduza numa primeira versão do módulo para que possam seguir para as próximas etapas, sendo a seguinte a fase de testes. Este processo está visível na figura abaixo.

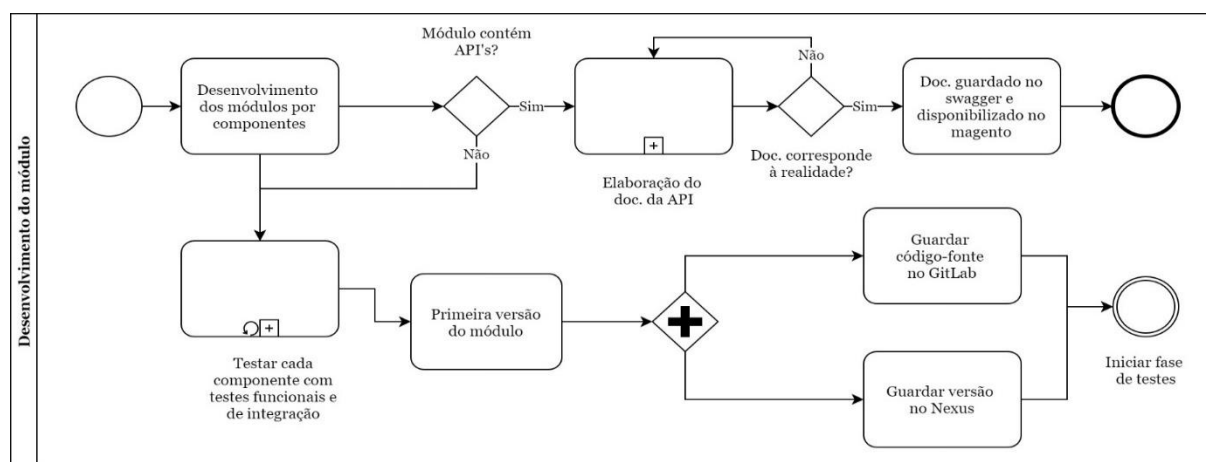


Figura 10 – Processo de desenvolvimento dos módulos

Após uma versão pronta para ser testada esta é em primeiro plano testada internamente no ambiente de testes destinado a esse efeito, cada teste resulta num documento chamado de “Caso de teste” onde são apresentados os resultados esperados e os resultados obtidos, caso sejam identificados erros é necessário voltar à parte do DM, caso esteja tudo operacional é disponibilizado para os clientes no ambiente staging, um ambiente de testes apenas para cliente onde eles podem testar a solução desenhada para si e dar o seu feedback. O processo de testagem com os clientes é equiparado ao processo de testagem interna, porém, existem dois tipos de problemas que podem ser levantados pelos clientes, estes podem encontrar erro, neste

caso retrocede-se até à escrita do código onde esses erros são corrigidos, ou podem sugerir algumas alterações e nestes casos o mais acertado é voltar à identificação de requisitos e objetivamente identificar as modificações pretendidas pelos clientes. Contudo, o que acontece no seu quotidiano é apenas voltar à escrita do código e realizar as alterações.

Dá-se por terminado o processo de DS quando estes estão em conformidade com as necessidades dos clientes, todos são agregados à solução desenhada para os clientes e esta é colocada no ambiente de produto, comumente referido como colocar a solução online. Se algum cliente desejar apenas obter um módulo é fundamental que a mesma versão seja validada pelo mesmo, iniciando-se o processo de realização de testes já descrito acima é representado na figura abaixo.

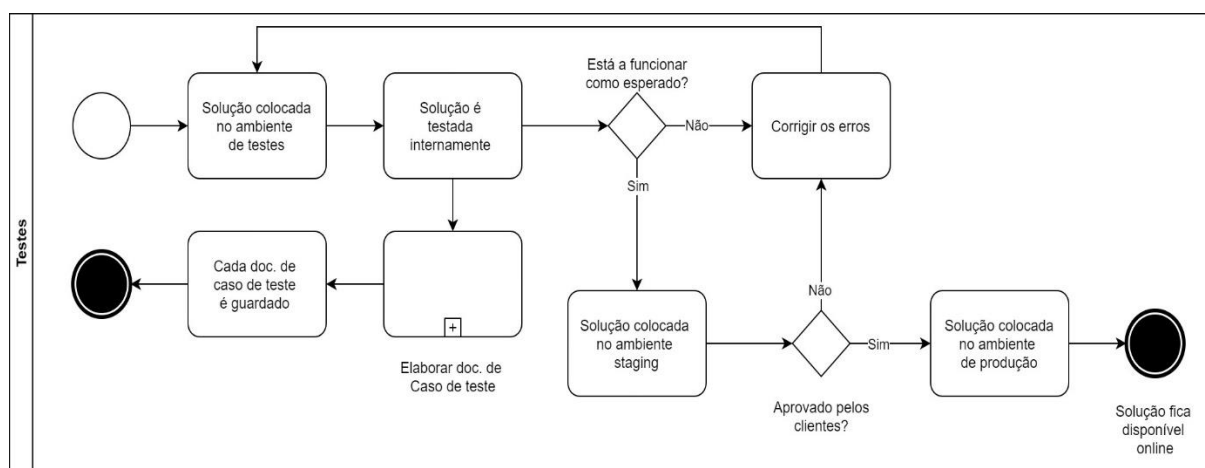


Figura 11 – Processo de realização de testes

A última etapa passa pela elaboração do manual de utilizador, onde é destacada uma pessoa para ficar responsável pela sua redação, procedendo a um levantamento de todas as informações necessárias, previamente identificadas, a conter no manual, seguindo-se da escrita do manual, aprovação do mesmo, a sua formatação e por fim a sua difusão para os clientes e armazenamento. Todo este processo de elaboração do manual é suportado pelo google drive. Na figura abaixo pode ser consultado o processo de elaboração do manual.

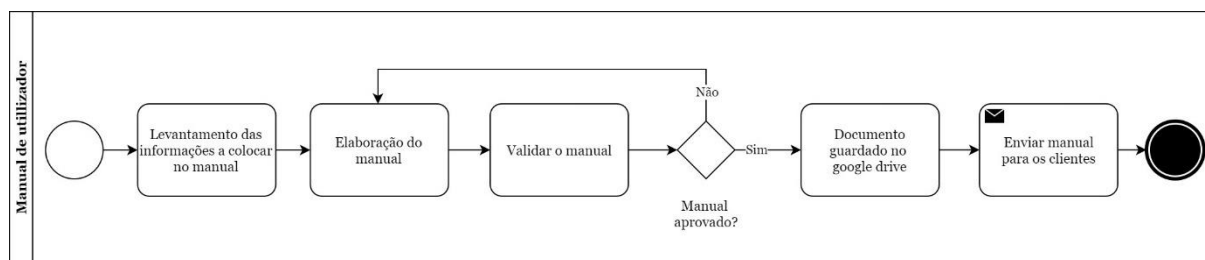


Figura 12 – Processo de elaboração do manual de utilizador

Os processos acima descritos e ilustrados apenas demonstram o processo de criação de documentação do sistema e do utilizador. Contudo, é também realizada documentação do processo através de relatórios diários, onde cada membro da empresa registar as suas atividades justificando as decisões tomadas. Existem também os diagramas de gantt criados através do registo de issues, na prática cada vez que é apresentado um problema este é aberto e inicia-se a sua resolução, o tempo dispensado e as tarefas realizadas são registadas no diagrama. Com os diagramas de gantt garantem que todo o processo de DS fica registado e nos relatórios diários podemos encontrar a justificação das decisões tomadas.

4.2.1 Caracterização dos documentos

Atualmente o processo de documentação ainda não se aproxima do ideal, são elaborados numa pequena escala e em tempos indevidos. Contudo, o processo de desenvolvimento de cada módulo encontra-se descrito através dos relatórios diários e dos registos feitos no Redmine através do levantamento de problemas, dos tempos dispensados na sua resolução e na representação gráfica destes dois elementos num diagrama de gantt. É de referir que estes últimos indicados não são vistos apenas como registos na cultura da presente empresa e não como documentos do processo. De seguida serão referidos os documentos do sistema e do utilizador.

Inicialmente é apenas elaborado um documento de especificação de requisitos onde estes são identificados através da análise das necessidades dos clientes e com uma pesquisa que tem a finalidade de discriminar o que já existe no mercado e, na maioria dos casos e ao contrário do recomendado, este documento não é atualizado com o decorrer do projeto, uma vez serve de apoio à escrita do código. No caso de conter uma API é gerado automaticamente um documento com a sua estrutura. Durante a escrita do código não é elaborada nenhuma

documentação, apenas inseridos comentários ao longo do código. Quando uma versão está pronta esta é submetida ao ambiente de testes onde, recentemente, se começou a adotar a prática de documentar esses mesmos testes através de casos de teste. Recentemente foi também iniciada a elaboração de um documento nomeado como plano de testes, estando o mesmo à responsabilidade de uma pessoa, que terá a finalidade de apoiar esses mesmo testes. Contudo, ainda não existe nenhuma estratégia de testagem bem delineada e com a devida documentação pensada. Por fim, são elaborados manuais de utilizadores, sem qualquer seguimento do processo anterior de desenvolvimento do módulo.

Quanto à documentação criada, foi sendo referido alguns dos documentos ao longo da explicação do DS, contudo é importante destacar quais documentos são criados em pormenor encaixando-os dentro da sua categoria, assim como a informação contida em cada um deles:

Documentos do Processo	Relatórios diários – são feitos todos os dias por todos os membros da empresa	○ O que fez no dia ○ O que vai fazer a seguir ○ Problemas que estejam a impedir de realizar o trabalho
Documentos do Sistema	Documento de requisitos – realizado em todos os projetos	○ Descrição do módulo ○ Requisitos funcionais ○ Requisitos não funcionais
	Documento de API - apenas quando o módulo contém uma API e o resultado final depende da estrutura do módulo e dos comentários inseridos ao longo do código	
	Documento de Testes - ainda não é uma prática recorrente, contudo começa a revelar-se necessário	Casos de testes: ○ Descrição do problema ○ Quantas vezes foi reproduzido ○ Passos realizados para reproduzir o erro ○ Resultados obtidos ○ Resultados esperados ○ Informação adicional

		Plano de testes: ○ Cenário ○ Modelo padrão ○ O que é suposto acontecer ○ O que acontece
Documentos do Utilizador	Manual de utilizador - criado apenas no fim por uma pessoa que não esteve presente durante o DS, então para criar o manual teve que falar com as diversas pessoas responsáveis pelos módulos	○ Características ○ Configurações ○ Como utilizar
	Manual de instalação - processo para a sua conceção é idêntica à do manual de utilizador.	

Tabela 2 – Documentos criados pela Skrey no desenvolvimento dos módulos

Em conclusão podemos referir que o processo de documentação dos módulos está ainda embrionário, tanto a nível de quantidade de documentos elaborados como na quantidade e qualidade de conteúdo que contém. Para além disso, foi também identificado que não acompanhando o processo de DS, muitas das vezes os documentos são apenas produzidos nas fases finais, ou quando os módulos são inseridos num projeto de maior dimensão com clientes, ou por outro lado, dá-se início à redação dos documentos, mas esta tarefa acaba por ser abandonada. Contudo, foi também identificada uma grande falta de recursos para que se torne uma prática recorrente e que esteja de acordo com as conformidades e padrões definidos, apesar do reconhecimento da sua importância e dos esforços para que venha a ser uma realidade da empresa.

4.2.2 Ferramentas utilizadas

De seguida serão apresentadas as ferramentas utilizadas para o DS, identificando as suas principais características e que função desempenham.

- Visual Studio Code (VSCode) (Microsoft 2021c)

O VSCode é um editor de código fonte desenvolvido pela Microsoft para Windows, Linux e MacOS, e apesar de ser um software livre de código aberto disponível no GitHub o download oficial está protegido sob uma licença privada. Suporta diversas linguagens de programação e um conjunto de recursos que podem ou não estar disponíveis, dependendo também de cada linguagem, esses recursos podem ser resumidos em: depuração - processo para encontrar e reduzir erros no software ou hardware -; controlo de GIT incorporado - sistema de controlo de versões -; refatoração - processo para modificar um sistema de software com o objetivo de melhorar a estrutura do código -; snippets - pequenas partes do código que facilitam a inserção de padrões de código repetidos -; realce de sintaxe - formatação diferente para cada categoria de termos dentro do código -; e um complemento para código inteligente. Apesar de já possuir uma série de recursos, é ainda possível entendê-los através de plugins disponíveis no seu repositório central.

O VSCode é utilizado para a escrita do código podendo utilizar a linguagem PHP ou JavaScript, dependendo das características e funcionalidades pensadas para o módulo. Não é mandatária a sua utilização para desenvolver esta tarefa, porém é a plataforma adotada pela maioria dos colaboradores da empresa.

- GitLab (GitLab 2021)

GitLab é usado como gestor de repositório de software baseado em GIT, usado principalmente no DS, acrescentando ainda as funcionalidades de histórico de edições de qualquer tipo de arquivo ou diretório, acompanhando as revisões não dependendo do acesso de rede ou ser um servidor central, suporte de wiki, gestão de tarefas, e permite ainda que o código seja escrito e armazenado nos seus próprios servidores. Além disso, é de referir que o GitLab é uma plataforma de software livre distribuído por uma licença MIT, garantindo uma maior facilidade de colaboração, visibilidade e velocidade de desenvolvimento.

Esta plataforma é usada para armazenar o código-fonte de cada código e oferece uma série de recursos que podem ser benéficos para a gestão do código-fonte de cada módulo, como:

- GP: criar projetos para hospedar o código-fonte;
- Repositório: garantindo a possibilidade de armazenar o código numa plataforma totalmente integrada;
- Gestão de problemas: rastreamento e discussão de problemas;

- Gestão de pedidos: possibilidade de pedidos de alterações, controlo das suas atividades e tempo dedicado;
- CI/CD: implementação automática integração contínua ou entrega contínua e uso de pipelines para sua configuração e visualização;
- Outras funcionalidades:
 - Wiki: para documentar o projeto;
 - Snippets: armazenar e trabalhar colaborativamente em snippets;
 - Analytics: analisar o ciclo de vida do desenvolvimento;
 - Insights: configurações importantes para o projeto;
 - Sintaxe: personalizar e destacar blocos de código;
 - Badges: adicionar imagens à página de informações do projeto;
 - Versões: controlo de versões;
 - Proprietários do código: especificar os proprietários do código;
 - Licença: aprovação das licenças do projeto;
 - Requisitos: criar critérios para verificação dos produtos;
 - Editor estático: editar o conteúdo de sites estáticos;
 - Code intelligence: navegar pelo código de forma inteligente.

- Nexus (Sonatype 2021)

O Nexus é um repositório de código aberto que oferece uma gestão dos componentes desde o início do seu desenvolvimento até à sua entrega ao cliente fornecendo suporte na gestão de binário – o que permite ao computador ler o software, palavra comumente utilizada para se referir a arquivo de compilações de vários componentes -, construção de artefactos,- associado a tudo o que é produzido durante o desenvolvimento do software, podendo ir desde diagramas de classes, casos de uso, planos de projeto, manuais, entre outros exemplos -, containers – uma unidade padronizada de software que permitem empacota-lo como um todo, tornando possível o seu transporte e execução noutros ambientes -, e montagem dos diversos componentes até ao produto acabado.

Com a integração do Nexus é possível publicar e recuperar todas as versões independentemente do ambiente em que se encontre. As ferramentas principais baseiam-se na identificação da versão, na navegação, premissas da administração, atualização global do repositório e visualização de recursos. Contudo, a sua principal função é o armazenamento do código de forma que a sua recuperação seja rápida e eficaz, para conseguir atingir esse objetivo foram definidos uma série de metadados: nome ou formato do repositório, grupo ou nome do componente e versão.

Posto isto, é de fácil precessão que o Nexus é usado como repositório interno para as versões compiladas dos componentes, ora após a construção dos diversos componentes, estão são agregados e enviados para os clientes. Todas as versões de todos os componentes são armazenadas e geridas nesta plataforma.

- Redmine (Lesyuk 2013)

O Redmine é um software livre baseado na web destinado à GP, sendo possível trabalhar em vários projetos e subprojetos, ser usada como base de dados, uma vez que aceita a integração com a maioria das bases de dados, e ainda como ferramenta de gestão de erros. O framework da plataforma é escrito em Ruby on Rails caracterizado por ter como objetivo aumentar a velocidade e facilidade no desenvolvimento de sites orientados a bases de dados. Decompondo o nome do framework, Ruby diz respeito à linguagem de programação dinâmica e forte com gestão automática de memória e de leitura facilitada. Rails é a ferramenta de desenvolvimento usada pelos desenvolvedores fornecendo uma estrutura a ser usada em todos os códigos, o seu framework ajuda a construir sites e aplicações de forma mais automática.

O Redmine disponibiliza uma série de funcionalidades garantindo um melhor fluxo na gestão dos projetos lá hospedados. Como um controlo de acessos para diferentes utilizadores, um sistema flexível para rastrear problemas onde podem definir o estado e o tipo do problema, ferramentas para controlo de tempo como os diagramas gantt e um calendário, módulos para a gestão de atualizações e de documentos, sistema de notificações através do feed ou via e-mail, fóruns e notícias, repositórios e wikis por projetos, integração de SCM (supply chain management) e de várias bases de dados.

Quando falamos em wiki é necessário ter em conta que o Redmine disponibiliza dois recursos diferenciados. Temos a formatação wiki, um recurso utilizado pelos seus componentes, encontrando-se sempre disponível para formulação de questões, descrição de projetos e documentos, mesmo com o módulo wiki desativado. O módulo wiki é o segundo recurso apresentado pelo Redmine, que implementa uma página wiki associada a cada projeto.

Com uma página wiki é possível realizar a GI e de conteúdo de forma simplificada, através de:

- Criação de novas páginas;
- Adição de ficheiros ou ligação a outros documentos já inseridos no Redmine;
- Ligação a páginas internas e externas;
- Adição de anexos e fazer uma gestão do repositório interno;
- Realizar um controlo de versões;

- Fazer referências a problemas;
- Ligar com diversos projetos e utilizadores.

Internamente o Redmine é utilizado para realizar a gestão dos projetos, já a wiki do Redmine ainda não é devidamente aproveitada, em alguns casos podemos encontrar a especificação de requisitos do projeto.

- Swagger (SmartBear Software 2021)

Swagger é uma linguagem de descrição de interfaces para descrever API's expressas em JSON, usado juntamente com um conjunto de ferramentas de código aberto para projetar, construir, documentar automaticamente, gerando códigos e casos de testes, usando servidores web. O uso de ferramentas do Swagger pode ser dividido em diferentes casos:

- Desenvolvimento de API's: através do Editor Swagger é possível criar um conjunto de API's pode ser gerado de forma automática um documento de especificação Open API baseado no próprio código;
- Interação de API's: o Swagger Codegen permite a construção simplificada de SDKs (Kits de desenvolvimento de software) para cada cliente para que estes possam instalar diretamente do documento de Open API;
- Documentação de API's: quando o API's é descrito por um OpenAPI é possível criar a documentação que descreva o mesmo de forma automática através do IU do Swagger que permite que qualquer pessoa consiga visualizar e interagir com os recursos finais que a API fornece sem ter que ser previamente, contendo no documento a forma de implementar o backend e o frontend.

A integração destas funcionalidades tem como objetivo simplificar o desenvolvimento de API's para os utilizadores, para as equipas e para as empresas, sendo usada com essa finalidade.

-WebHost Manager Complete Solution (WHMCS) (WHMCS 2021)

O WHMCS é um sistema de gestão de clientes, administração financeira e prestação de suporte com diversas funcionalidades de controlo para esse fim, sendo utilizado e recomendado para empresas de web hosting. No quotidiano, ele ajuda na otimização das tarefas de cada colaborador de uma organização, através dos diversos recursos disponíveis como:

- Cobrança automática;
- Área do cliente associado ao sistema de tickets;

- Relatório estatísticos e contabilidade;
- Integração com órgãos de registos de domínios;
- Sistema de tickets;
- Administração financeira automática;
- Integração com várias empresas de pagamento online;
- Suporte para faturas, orçamentos e estimativas.

Em suma, a ferramenta é usada para economizar tempo e a faturação, integração em domínios web e fornecer suporte aos clientes, sendo estes os motivos que levam a empresa em estudo a utilizar esta ferramenta. O principal objetivo com o seu uso está diretamente relacionado com a comunicação com o cliente, uma vez que através do sistema de tickets do WHMCS mantêm uma comunicação bidirecional durante todo o projeto garantindo também suporte e manutenção após o seu término.

4.3 Diagnostico da Gestão de Informação na empresa

Durante a análise do processo de DS foi notória uma falta constante de referência a documentos que acompanhassem essa prática, à semelhança do identificado na revisão de literatura, é da opinião de muito que o código-fonte, juntamente com os comentários nele inserido, muitas das vezes é suficiente para descrever o produto. Não obstante, existem documentos que foram sempre mencionado, sendo eles o documento de requisitos, aqui foi identificado um certo descuido, tanto na sua elaboração como na sua manutenção, e o documento de API. Independentemente desta realidade, foi notória uma preocupação crescente em dedicar esforços à elaboração de toda a documentação necessária, sendo mais perceptível esta preocupação na descrição dos testes realizados. Outro documento elaborado é o manual de utilizador, contudo este não foi referido associado ao DS, apenas é redigido quando há essa necessidade, esta tarefa fica ao encargo de um integrante da empresa, que procede à recolha de informações sobre os módulos previamente definidas como necessárias a constar nos manuais, após a conclusão desta fase dedica-se à elaboração do manual. Neste campo a principal adversidade encontra-se na qualidade do manual, se nestes constam toda a informação necessária e se a mesma está clara e precisa. É reconhecida a importância dos manuais tanto como fonte de CO e também como ferramenta de comunicação com o cliente.

Ao longo da análise da empresa tornando-se evidente a falta de preocupação por parte dos colaboradores da empresa às práticas de GD recomendadas para um bom desenrolar de cada projeto, quer seja em maior escala ou em menor escala.

Com estes pontos foram identificadas as duas principais necessidades da empresa, tendo como fonte vários colaboradores da empresa:

1. Criar todos os documentos propostos para o DS: não só documentos que acompanhem todo o processo de DS, mas também aqueles que dão suporte a essa prática e os que são elaborados com a finalidade de serem partilhados com atuais e futuros clientes;
2. Criar os documentos em tempos devidos: é importante que os documentos sejam elaborados nas fases mais adequadas para que não haja perda de conhecimento garantindo que o conteúdo dos mesmos será o mais fidedigno possível, ademais é crucial que estes sofram alterações sempre que necessário.

Para ser possível responder a estas duas necessidades é fundamental ter em conta alguns aspetos. Em primeiro lugar é preciso referir que este problema se enquadra no DM que, como visto anteriormente, são unidades independentes que segregam uma série de funcionalidade à solução tecnológica que está a ser criada e, em consequência dessa realidade, os documentos a serem criados serão pensados nessa perspetiva, não de um software completo. Ainda tendo em conta a realidade acima referida, os documentos serão pensados para que seja possível a sua reutilização. Numa primeira instância reutilizar o conteúdo de documento para documentos dentro do mesmo projeto, ou reutilizar o documento na totalidade quando o módulo é integrado num software de maior dimensão tornando-se imprescindível que o documento seja também anexo à restante documentação.

Para concretizar estes pontos existem aspetos ligados à GC, como já identificado no ponto 2.1 da revisão de literatura, que não podem ser ignorados:

1. Captura: as informações a conter nos documentos devem ser pensadas de forma que este seja o mais completo possível sem que se torne demasiado consistente, sendo possível a reutilização dessas informações;
2. Organização: deve estar alinhada com a realidade da empresa, ora deve ser exercida na adequada com os documentos devidamente descritos favorecendo a sua recuperação;
3. Armazenamento: no local apropriados durante o tempo necessário;
4. Partilha: associada a questões de recuperação dos documentos que deve ser fácil e eficaz facilitando a partilha dos documentos interna e externamente.

Desta forma, encontramos as diretrizes para a criação de uma proposta que irá de encontro às necessidades referidas.

5 Proposta para processo de Gestão de Informação para componentes de software

5.1 Documentos a serem criados

Quando pensamos em criar documentos independentemente do seu objetivo existe uma série de considerações que devem ser tidas em conta, segundo a Norma Portuguesa 4438-1:2005 – “Informação e documentação. Gestão de documentos de arquivo. Parte 1: Princípios directores” (IPQ 2005) os documentos devem ser criados, organizados e arquivados de forma pertinente, quer seja por parte de organismos públicos ou privados, e independentemente do seu formato ou suporte, podendo ser produzido ou recebido pelo organismo em questão ou por uma pessoa individual. Posto isto, a presente norma apresenta-nos uma normalização dos procedimentos para a GD definindo as responsabilidades dos organismos face aos seus documentos em todo o seu ciclo de vida. Contudo, para levar a cabo uma boa GD existem uma série de aspetos sobre os documentos que as organizações devem ponderar:

- Quais devem ser produzidos para cada processo de negócio assim como a informação que estes devem conter;
- Decidir qual a sua forma, estrutura e que tecnologias devem ser utilizadas;
- Determinar a metainformação que deve ser associada aos documentos de modo que favoreça a gestão dos mesmos;
- Como será feita a recuperação, utilização e transmissão dos documentos entre todos os intervenientes no processo de negócio, assim como os prazos adequados para cada fase;
- Preservar os documentos para que sejam acessíveis a longo prazo, no entanto apenas durante o tempo estritamente necessário;
- Garantir que os documentos são mantidos num ambiente protegido e seguro.

Não descurando a necessidade de uma avaliação contínua da qualidade dos processos internos, apelando à criticidade para identificar aspetos que possam ser melhorados, principalmente na eficácia e eficiência dos mesmos. Quanto aos documentos especificamente, é preciso realizar periodicamente ações de monitorização para assegurar que estão em conformidade com os processos e procedimentos referidos, e também que os resultados esperados estão a ser os obtidos.

Os documentos podem tornar-se uma fonte de informação notável para as organizações, e quando bem manuseada pode ser considerada um dos recursos mais valiosos, contudo, antes

de entender o processo de gestão de documentos é necessário assimilar os benefícios que esta prática fornece. Com uma gestão adequada da documentação de uma organização esta pode levar a uma condução das atividades de maneira ordenada, mais eficiente e responsável, garantir que os serviços prestados são consistentes e prudente, suportar as tomadas de decisão, facilita o desempenho das atividades organizacionais, assegurar continuidade, satisfazer requisitos legais para a organização esteja em conformidade com os mesmos, fornecer proteção e suporte na gestão de riscos, ter em atenção dos interesses da organização assim como de todas as partes interessadas e por fim estabelecer uma entidade organizacional preservando a sua memória.

Por conseguinte as fases do processo para o controlo e GD baseiam-se nos seguintes:

- Determinação dos documentos a serem produzidos: através de uma análise aprofundada da empresa e dos seus processos de negócio com o objetivo de identificar as suas necessidades e consequentemente os documentos que se adequam à sua realidade;
- Determinar os prazos de retenção: dependendo da avaliação das exigências que forem determinadas na fase anterior, tendo em conta que é fulcral responder não só as exigências presentes da organização e de intervenientes externos, mas também às futuras, assumir responsabilidades legais e das necessidades de gestão e de avaliação de riscos;
- Integração dos documentos: numa primeira instância é imprescindível estabelecer um enquadramento do contexto de criação do documento estabelecendo ligações com quem o produziu e com a entidade organizacional
- Registo: provar que o documento foi criado ou integrado tornando possível a sua recuperação
- Classificação: das funções e atividades ligadas aos documentos
- Armazenamento e manutenção: devem ser armazenados em suportes que facilitam a sua utilização, fidedignidade, autenticidade e preservação enquanto necessário, estando presentes ao longo de toda a vida do documento
- Acesso: definir formalmente as autorização e condições de acesso aos documentos

Apesar de ser reconhecida a importância da documentação durante o processo de DS e que há um processo de documentação ideal, é de referir que a solução ideal muitas das vezes não é de possível implementação por falta de recursos. Neste caso, foi pensada uma solução adequada aos recursos disponíveis, sendo que os documentos propostos são os seguintes – relatórios

diários, diagramas de gantt, documentos de requisitos, documento de arquitetura, documentos API, documentos de testes e manuais de utilizador. Para definir a estrutura de cada documento proposto foi usada a norma ISO/IEC/IEEE 15289 – “Systems and software Engineering – Content of life-cycle information items (documentation) (IEEE 2019) Systems and software engineering – content of life-cycle information items (documentation), que num modo geral específica o conteúdo dos itens de informação, os documentos, mapeando o conteúdo, templates e modelos de informação.

A norma referida anteriormente não apresenta nenhuma descrição para os documentos de processo definidos, nem para os documentos de API, como tal não serão apresentados de seguida. As especificações dos requisitos do sistema/software podem ser divididas em requisitos específicos apresentados pelas partes interessadas, neste caso os clientes que apresentam as suas necessidades, vontades, desejos e expectativas, e em requisitos do sistema, ora as funcionalidades base que o sistema ou software têm obrigatoriamente que apresentar e requisitos de negócio ou organizacionais. Cada requisito deve ser identificado de forma única. A arquitetura do módulo é descrita através da apresentação do conceito do sistema em termos de elementos a incluir e do modelo do domínio, fundamental para identificar a sua finalidade, qualidade, restrições e decisões a serem tomadas. Posto isto, um documento de arquitetura deverá – justificar as decisões tomadas, estabelecer os princípios dos elementos do sistema, registar propriedades importantes e relações entre os elementos.

Os documentos de testes devem ser divididos em plano de testes onde são apresentados os testes que são realizados, definindo as ações a serem tomadas e os procedimentos para a resolução de problemas, e os casos de testes onde são indicados os resultados dos testes realizados, estes indicam a qualidade do sistema a responder a cada requisito. Por fim, o documento de utilizador fornece aos mesmos os procedimentos para efetuar tarefas específicas do sistema/software, podendo incluir procedimentos de implementação, integração, instalação, desinstalação, operação ou descarte. A sua estrutura deve basear-se na descrição do uso pretendido do sistema, avisos, cuidados e possíveis soluções, identificar os recursos necessários e o ambiente operacional.

Posto isto, os documentos a serem produzidos podem ser traduzidos nos apresentados na tabela abaixo, assim como uma descrição geral das informações que devem conter. Nos anexos referidos na tabela dizem respeito aos templates criados para cada documento.

Documentos do sistema	Documento de requisitos – Anexo C	○ Qual a necessidade de criação dos módulos, a sua finalidade e que valor pode agregar ○ Descrever quais os objetivos do módulo, as suas características e as decisões tomadas sobre as funcionalidades a inserir. ○ Visão geral do módulo, funcionalidades e restrições que podem levar ao não funcionamento das ações pensadas. Requisitos funcionais e não funcionais ○ Critérios para aceitação por parte do cliente
	Arquitetura do módulo - Anexo C	○ Descrever a solução que será proposta ○ Detalhar todas as funcionalidades do módulo ○ Usar diagramas e/ou outros materiais gráficos para representar a estrutura
	Documentos de testes - Anexo C	○ Um plano de testes onde num possível cenário é referido o que é suposto acontecer e quando, as ações necessárias e os resultados esperados ○ Descrever o teste feito, apresentar os resultados esperados e os obtido
	Documento API - Anexo C	○ Documentação API que deverá conter as informações necessárias sobre o API, e fazendo a ligação à documentação própria do API
Documentos do processo	Diagramas gantt	○ Atividades ○ Problemas ○ Tempo despendido
	Relatórios diários	○ Atividades realizadas ○ Decisões tomadas ○ Justificações ○ Problemas encontrados ○ O que vai fazer a seguir

Documentos do utilizador	Manual de utilizador completo, mas dividido nas seguintes secções - Anexo C	○ Descrição do módulo (introdução) ○ Características e benefícios ○ Requisitos para adquirir o módulo ○ Guia de funcionalidade ○ Configuração do módulo ○ Processo de instalação detalhado ○ Possíveis problemas que podem surgir e como resolver ○ Processo de atualizações ○ Questões frequentes
---------------------------------	--	--

Tabela 3 – Proposta de documentos a serem criados durante o desenvolvimento dos módulos

Foi também identificada a possibilidade de reutilizar o conteúdo de documentos elaborados ao longo do projeto, sendo eles o documento de requisitos, de arquitetura e de API's, para ajudar a elaborar o manual de utilizador, uma vez que muitas das informações podem ser encontradas nesses documentos, poupando esforços e tempo dos membros da empresa que pode ser aproveitado para outros projetos. Podem também ser consultados os documentos de processo com o objetivo de responder a alguma questão que surja ao longo da elaboração do manual, e em última instância contactar a(s) pessoa(s) destacada(s) para o desenvolvimento do módulo.

5.1.1 Ciclo de vida dos documentos

Os documentos a serem criados terão tipologias e conteúdos diferenciados e, para que acompanhem o DM, serão elaborados ao longo do seu progresso. Uma das principais razões para a elaboração da documentação aquando do desenvolvimento do módulo baseia-se na impossibilidade de identificar todo o conteúdo diversificado que terão de uma só vez, e tendo como segunda razão o facto de não ser uma boa prática a sua elaboração apenas quando já identificaram todas as informações necessárias a constar nos documentos, ora apenas quando o módulo se encontra pronto a ser comercializado, tornando-se fundamental este acompanhamento. Posto isto, é importante referir que os documentos passaram por diversos estados até estarem completos, e que sofreram alterações ao longo do processo.

A documentação do processo não interfere diretamente com o DS, quer isto dizer que não existe uma documentação própria do processo apenas dedicada aos módulos, mas sim, quando os colaboradores trabalham num módulo as decisões que tomam, as atividades que fazer

relacionadas com o seu desenvolvimento e o tempo gasta em cada uma fica registado. Assim, garantem que as decisões sobre resolução de problemas, adoção de certas medidas e decisões ficam guardadas e que, para além disso, todo o processo de DS é acompanhado pela documentação do processo desde o seu início até estar completamente finalizado. Levando esta realidade em consideração, nos parágrafos seguintes serão apenas abordados os documentos do sistema e do utilizador.

Primeiramente é fundamental identificar qual o tempo ideal para dar início à redação de cada documento, podendo esta prolongar-se com o desencadear no âmbito dos projetos em que cada módulo está a ser desenvolvido, podendo ser um projeto específico para um cliente ou um projeto interno. Abaixo serão apresentados os momentos em que os documentos devem ser criados tendo em conta o processo de DS, apenas aplicável à documentação do processo e do utilizador:

- Antes do início do desenvolvimento do módulo onde são apresentadas as diretrizes para dar continuidade ao trabalho;
- Durante o desenvolvimento do módulo, pois a sua redação acompanha o desenvolvimento do módulo sendo, em alguns casos, fundamentais;
- Final do desenvolvimento quando já existem todas as informações necessárias que devem ser incluídas nos documentos.

Documentos

Doc. de requisitos	Antes /durante do desenvolvimento do módulo
Doc. de arquitetura	Antes /durante do desenvolvimento do módulo
Testes	Durante o desenvolvimento do módulo
API's	Durante o desenvolvimento do módulo
Manual de utilizador	No fim do desenvolvimento do módulo

Tabela 4 – Fase para a criação dos documentos

O documento de requisitos e o documento de arquitetura devem ter o seu início antes do desenvolvimento do módulo para que sejam especificados os requisitos que levam à criação do

módulo, ora o problema, e as funcionalidades que este terá, ora a solução que será desenhada. Contudo, poderão sofrer alterações durante o processo uma vez que poderão surgir novos requisitos e novas funcionalidades fulcrais para o sucesso do módulo. Com estes dois documentos podemos assumir que temos um esboço do que será o módulo e dá-se início ao processo de desenvolvimento do módulo.

O documento de API é apenas criado quando o módulo contém API's, e como já identificado anteriormente é gerado automaticamente através dos comentários inseridos no código, tornando obrigatória a sua fase de produção durante o desenvolvimento do código.

Os testes aqui referidos são os testes finais, não os testes funcionais e de integração uma vez que estes são feitos durante a escrita e validação do código e não necessitam de documentação, já os testes finais são feitos no ambiente staging quando já existe uma primeira versão do módulo que segue para comercialização, caso o resultado seja positivo, ou voltar à escrita do código onde serão corrigidos os erros identificados pelos testes, ora isto acontece quando o resultado dos testes é negativo. Para além dos testes realizados internamente quando o módulo surge num contexto de projetos com clientes o módulo segue para os mesmo mas antes de entrar no ambiente de produção é também submetido a testes por parte dos clientes, se estes não validarem o módulo deverão apresentar as suas questões e dependendo do seu feedback os documentos até então produzidos podem sofrer alterações, nomeadamente o de requisitos e de arquitetura, os restantes permaneceram iguais, podendo aumentar em quantidade, por exemplo se forem realizados mais testes o número de documentos de testes também irá aumentar. Se o cliente aprovar o módulo este seguirá para produção, ora ficará disponível online, finalizando-se assim o processo de desenvolvimento do módulo. O mesmo processo acontece se o cliente requer um módulo específico que já se encontre em comercialização e propuser alterações.

Por fim é elaborado o manual de utilizador que será também entregue ao cliente, a sua redação será apenas no final pois apenas nesta fase é possível ter todo o conteúdo necessário a abordar no manual, porém, e como já referido anteriormente, grande parte da informação pode ser extraída de documentos elaborados anteriormente.

Podemos concluir que o ciclo de vida de cada documento será diferenciado, contudo a Norma Portuguesa 82045-1:2015 – “Gestão de documentos. Parte 1: Princípios e métodos)” (IPQ 2015) foca-se na gestão de documentos especificando quais os princípios e métodos a serem usados nessa prática abrangendo todo o ciclo de vida dos documentos. Este ciclo vai,

idealmente, desde a ideia conceptual de um documento até à sua supressão. Os princípios da gestão de documentos apresentados pela presente norma baseiam-se na utilização de metadados que identifiquem e/ou descrevem o documento, fornecendo um valor acrescido pois permitem a sua gestão, pesquisa, recuperação e identificação, garantindo apoio na gestão do seu fluxo de trabalho e de relações entre documentos e outros associados, entidades, pessoas, entre outros.

Para além da utilização dos metadados para a identificação e descrição dos documentos, estes devem também estar associados ao ciclo de vida do documento, ao processo de negócio no qual o documento está a ser utilizado, vinculando informação entre atividades à sua estrutura, podendo ser resultado de uma atividade processual. Neste caso, o processo de negócio em que os documentos serão produzidos é o desenvolvimento de módulo de software, aqui podemos identificar que a elaboração dos documentos está associada à conceção de um produto. Um documento associado ao ciclo de vida um produto, de frisar que apesar de ser associado é independente do mesmo, está encarregado de representar a informação sobre o mesmo e sobre o seu processo de construção, contudo as diferentes fases específicas do ciclo de vida do produto geram diversas versões distintas dos documento, e num ambiente onde todas estas questões se encontrem bem definidas é necessário ter também em conta quando é necessário elaborar uma nova versão de um documento que pode ocorrer por dois tipos de necessidades de modificações - modificação da informação ou modificação da apresentação visual da informação.

O estabelecimento de regras e procedimentos para o manuseamento de documento irá depender do seu ciclo de vida que, segundo a norma NP 82045 -1 pode dividir-se nas seguintes fases:

1. Iniciação: fornecer e adquirir os dados que levam à preparação do documento
2. Preparação: inclui a produção de um documento até que este esteja pronto para a próxima fase
3. Estabelecimento: ou verificação e aprovação, necessários antes que o documento seja utilizado para a finalidade que foi pensado
4. Utilização: manter os documentos disponíveis para utilização sempre que for necessário
5. Revisão: quando são necessárias modificações, podendo ser uma modificação do conteúdo ou da finalidade de publicação do mesmo
6. Arquivo: armazenamento de versões anteriores do mesmo documento
7. Supressão: normalmente quando cessou o tempo de responsabilidade do produto

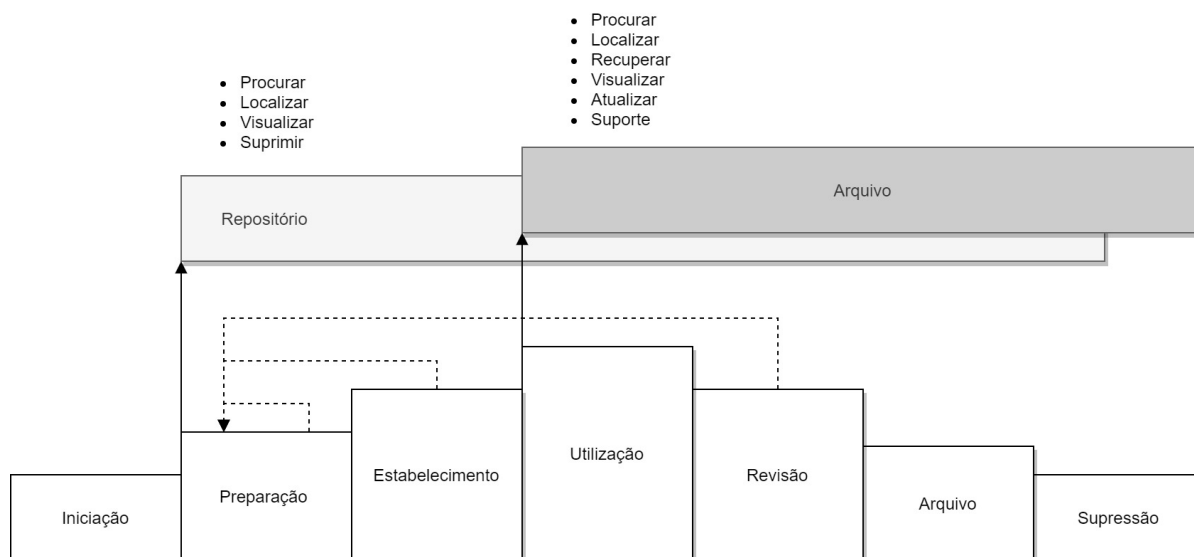


Figura 13 – Ciclo de vida dos documentos

Fonte: NP 82045 -1:2015

Segundo a descrição apresentada na norma para o fluxo das fases anteriormente apresentadas, temos a fase de iniciação onde o documento obtém uma identificação e é classificado de acordo com o conteúdo que o mesmo irá conter. Na fase de preparação o conteúdo é desenvolvido até que esteja pronto para passar para a fase de verificação e aprovação por parte de membros superiores dentro da organização e, caso seja necessário, entidades externas envolvidas. Uma vez iniciada esta 3ª fase deve-se proceder a todas as modificações que surjam garantindo que são rastreáveis, tendo como seu término a publicação do documento, entenda-se por publicação o documento estar disponível para os fins pretendidos. A fase seguinte dedica-se a alterações que possam surgir, contudo após algum tempo o documento acaba por perder a sua utilidade sendo anulado, mas mantido em arquivo, acabando por ser eliminado aquando do fim das obrigações contratuais sobre o mesmo. A fase de supressão dedica-se à eliminação dos documentos do arquivo onde estes são definitivamente apagados assim como os metadados que lhes tenham sido associados deixando de ser rastreável.

De seguida iremos ver documento por documento em que altura passam por cada estado tendo também em conta a fase de desenvolvimento do módulo – antes de iniciar o desenvolvimento do módulo, durante o desenvolvimento do módulo, depois do desenvolvimento do módulo.

Estados	Antes	Durante	Depois
Iniciação	Doc. Requisitos Doc. Arquitetura	Testes API's	Manual de utilizador
Preparação	Doc. Requisitos Doc. Arquitetura	Doc. Requisitos Doc. Arquitetura API's Testes	Testes Manual de utilizador
Estabelecimento	Doc. Requisitos Doc. Arquitetura	Doc. Requisitos Doc. Arquitetura API's Testes	Testes Manual de utilizador
Utilização		Doc. Requisitos Doc. Arquitetura API's Testes	Todos
Revisão	Doc. Requisitos Doc. Arquitetura	Doc. Requisitos Doc. Arquitetura	Manual de utilizador
Arquivo			Todos
Supressão			Todos

Tabela 5 – Estado dos documentos

- Documento de requisitos e documento de arquitetura

O documento de requisitos e o documento da arquitetura passam por todos os estados sempre em conjunto, uma vez que o primeiro é a apresentação das necessidades para a construção do módulo e a arquitetura é a resposta a essas necessidades. Antes de dar início ao

desenvolvimento do módulo estes dois documentos passam por três estados, o de iniciação, preparação e de estabelecimento, ora dá-se início à sua elaboração onde são identificados os requisitos e elaborada a arquitetura do módulo sendo já esta a fase de preparação, e posteriormente é necessário que sejam validados internamente antes de passar à fase de conceção do módulo. Contudo passam também pela fase de preparação e de estabelecimento durante o desenvolvimento do módulo uma vez que podem surgir novas necessidades e consequentemente são identificados os requisitos e as funcionalidades adicionais que serão acrescentadas, sendo sempre necessária a verificação e aprovação para que se continue com a escrita do código. Quando são aprovados entram no estado de utilização, onde numa primeira instância são consultados para a escrita do código e, posteriormente, sempre que necessário. Já o estado de revisão pode ocorrer durante duas fases, antes de dar início ao desenvolvimento do código e durante o mesmo, devido a sugestões de alterações que podem vir de colaboradores da empresa, quer sejam intervenientes no projeto ou não, ou então, e caso o módulo esteja a ser desenvolvido no âmbito de um projeto com clientes, serem alterações e sugestões propostas pelos mesmos, quando acontece antes do desenvolvimento do módulo unicamente para validação por partes dos clientes para que se possa dar início ao desenvolvimento, já quando acontece durante o desenvolvimento é devido a sugestões de alterações que os mesmos sugeriram. O estado de arquivo acontece quando os módulos sofrem alterações dando origem a novas versões sendo imprescindível adaptar os documentos para estarem de acordo com essas alterações tendo também como resultado novas versões dos documentos, sendo que as versões antigas passam ao estado de arquivo, ficam armazenadas, mas facilmente recuperáveis. Por fim, o último está de supressão refere-se à eliminação dos documentos, isto acontece quando os módulos são eliminados.

No [anexo D](#) é possível ver o processo de desenvolvimento dos dois documentos juntamente com os estados em que estes se encontram em cada fase. Na figura abaixo encontra-se representada graficamente a descrição apresentada no paragrafo anterior.



Iniciação	Preparação	Estabelecimento	Utilização	Revisão	Arquivo	Supressão
Levantamento de requisitos	Elaboração do doc. de requisitos	Suporte ao desenvolvimento do módulo	Disponíveis para consulta e reutilização	Criada nova versão do módulo	Documento da versão antiga é arquivado	Módulo é eliminado
	Validação interna e externa					

Figura 14 –Ciclo de vida do documento de requisitos

Fonte: Autoria própria adaptada da NP 82045 -1:2015



Iniciação	Preparação	Estabelecimento	Utilização	Revisão	Arquivo	Supressão
Desenhar arquitetura do módulo	Elaboração do doc. de arquitetura	Suporte ao desenvolvimento do módulo	Disponíveis para consulta e reutilização	Criada nova versão do módulo	Documento da versão antiga é arquivado	Módulo é eliminado
	Validação interna e externa					

Figura 15 – Ciclo de vida do documento de arquitetura

Fonte: Autoria própria adaptada da NP 82045 -1:2015

- Documento de API's

O documento com a descrição da API's tem quase todo o seu processo de elaboração, passando por quase todos os estados, durante o desenvolvimento do módulo, mas concretamente pelo estado de iniciação, preparação, estabelecimento e utilização, sendo que apenas passa pelo estado de arquivo e de supressão no final no desenvolvimento não passando pelo estado de revisão uma vez as API's não sofrem alterações e consequentemente o documento também não, podendo apenas ser usadas diferentes API's e nesses casos são criados novos documentos que passam por todo o processo acima descrito.

À semelhança dos documentos anteriores, na figura seguinte podemos ver os estados da documentação de API's, e no [anexo D](#) o seu processo.

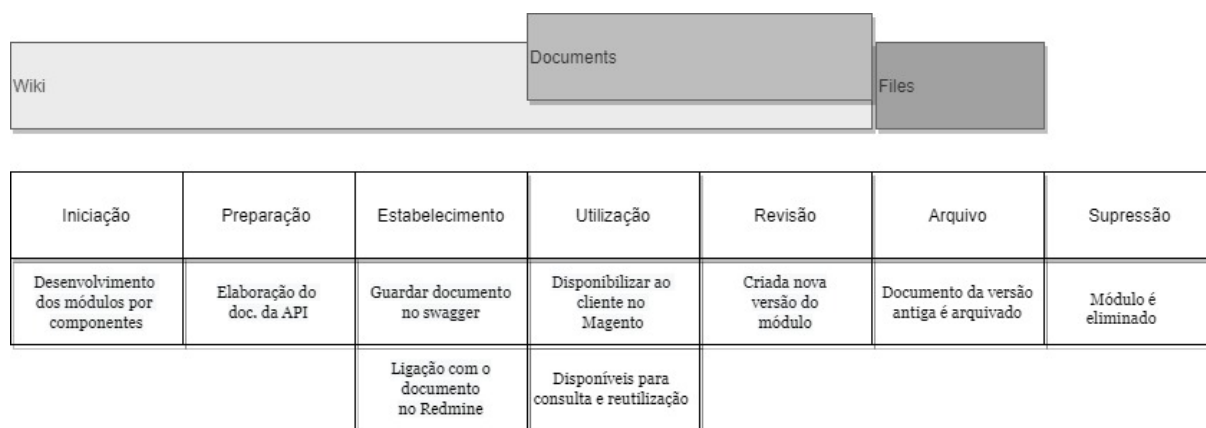


Figura 16 – Ciclo de vida do documento de API

Fonte: Autoria própria adaptada da NP 82045 -1:2015

- Documentos de testes

Os documentos de testes aqui referidos dizem respeito aos testes realizados no ambiente de testes e não aos testes funcionais e de integração realizados ao longo da escrita do código. Primeiramente é necessário lembrar que no âmbito da documentação de testes temos dois documentos, o plano de testes e o caso de testes, o primeiro documento deverá ser elaborado anteriormente dos casos de testes, uma vez que, como o próprio nome indica, serão um plano de testes que terá o papel de guiar os testes a serem feitos. Com isto, é de frisar que os testes são realizados durante o desenvolvimento do módulo uma vez que no processo de DS tem uma fase destinada apenas à realização dos testes, e em consequência os documentos a serem criados nesse âmbito veem o seu ciclo de vida a ocorrer apenas durante o desenvolvimento do módulo e após a sua finalização. A sua fase de iniciação dá-se durante o desenvolvimento, contudo o estado de preparação, estabelecimento e utilização podem ocorrer tanto durante ou depois do desenvolvimento do módulo, ou iniciar-se numa fase e acabar na seguinte. Quando se dá a fase a aprovação na fase de estabelecimento, ora quando os testes realizados não apresentam nenhum erro e os documentos estão de acordo com o sucedido é considerado um marco para dar como terminado o processo de desenvolvimento de um módulo, de seguida o documento passa à fase de utilização, onde este se encontra disponível para consulta interna sempre que necessário por parte de qualquer colaborador. Os documentos de testes não passam pela fase de revisão, uma vez que não existem novas versões a serem criadas dos mesmos, sendo apenas efetuados novos testes e documentos a descrever os mesmos. Os

presentes documentos passam ao estado de arquivo apenas quando o módulo é aprovado e finalizado, ora os testes correrem todos como o esperado e, à semelhança dos anteriores, a fase de supressão acontece quando o módulo é apagado.

Na figura abaixo encontramos representado os estados dos documentos de testes, nomeadamente o plano de testes e os casos de testes. Acompanhado do [anexo D](#).

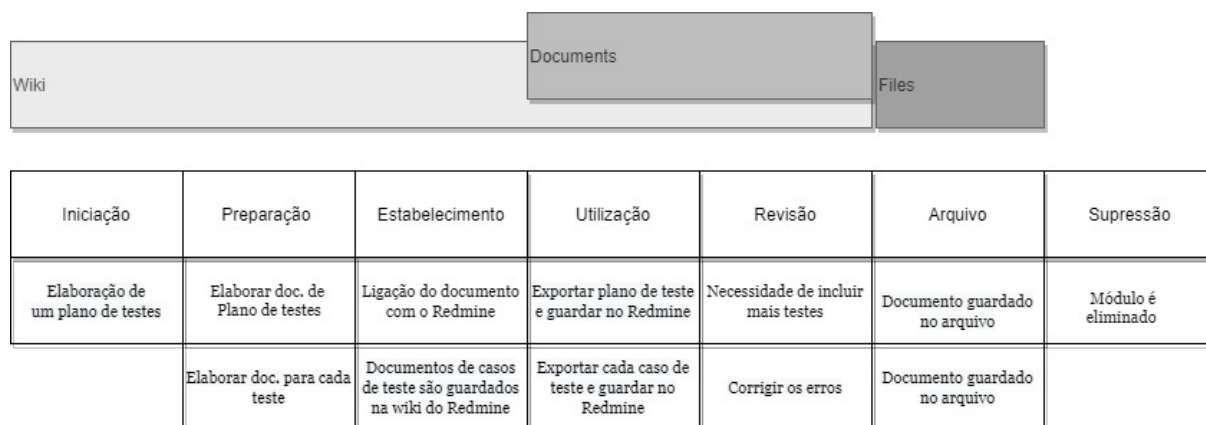


Figura 17 – Ciclo de vida dos documentos de testes

Fonte: Autoria própria adaptada da NP 82045 -1:2015

- Manual de utilizador

Por fim, o manual de utilizador passa por todos os estados já na última fase identificada, ora apenas após o final do desenvolvimento do módulo. Apenas nesta fase é possível recolher todas as informações que constarão no manual, quer sejam retiradas dos documentos redigidos anteriormente quer seja por consulta das pessoas que os desenvolveram. Estas duas práticas referidas anteriormente para a elaboração dos manuais devem ser seguidas para garantir que estes contêm todo o conteúdo necessário e que este está correto.

Num modo geral os documentos são preparados adquirindo todos os dados necessários na fase de iniciação que pode ocorrer em alturas diferentes para cada documento, a partir daí as próximas fases variam também de documento para documento, mas acompanham o processo de desenvolvimento do produto, neste caso o desenvolvimento do módulo. Todos os documentos podem intercalar entre os estados de preparação e de estabelecimento, uma vez que é necessário que sejam revistos até serem aprovados. Após a difusão dos componentes de software, nomeadamente os módulos, os documentos permanecem no estado de utilização, onde se encontram devidamente guardados no arquivo, permitindo que estes sejam mantidos, recuperados e consultados por todos os colaboradores ou distribuídos para os seus clientes ou

interessados, avançando para a fase de revisão quando o módulo é sujeito a modificações sendo também essencial alterar o conteúdo dos documentos, o que resultará numa nova versão dos documentos que ficará no estado de utilização e o documento anterior passará à fase de arquivo. Todos os documentos permanecem nesse estado até que o componente de software seja eliminado, não havendo mais a necessidade justificada de manter os documentos que se tornaram obsoletos.

Por fim, podemos ver na figura seguinte a representação do último documento proposto, assim como o seu processo no [anexo D](#).

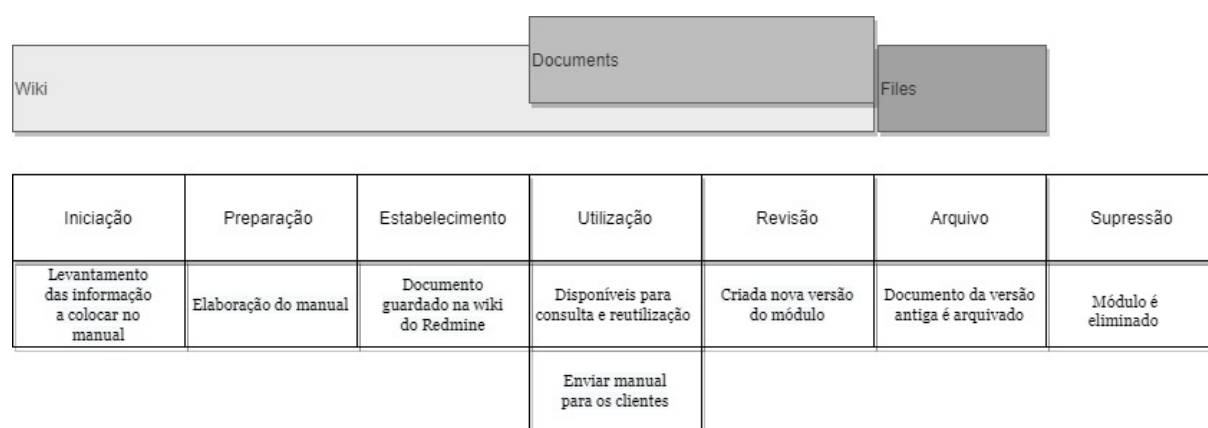


Figura 18 – Ciclo de vida do manual de utilizador

Fonte: Autoria própria adaptada da NP 82045 -1:2015

É de destacar que a documentação deverá acompanhar o DS, numa primeira instância servir de apoio para a escrita do código, e posteriormente guardada para que seja consultada, reutilizada ou alterada sempre que necessário. Acompanhando também quando o produto é distribuído, pois quando o módulo é entregue ao cliente é também entregue a documentação apropriada, especificamente os manuais de utilizador. Por fim, quando o módulo é sujeito a alterações surgindo novas versões o mesmo acontece com os documentos, como visto anteriormente.

5.2 Processo para a Gestão dos Documentos

Como já destacado anteriormente, podemos dividir os documentos de software em documentação do processo, sistema ou do utilizador. Cada uma delas tem diferentes objetivos, mas todos os documentos, independentemente da sua tipologia devem acompanhar o

desenvolvimento da solução, neste caso específico foca-se apenas em módulos de software. Os documentos do processo não requerem que seja dispensada grande atenção, uma vez que foram apenas selecionados aqueles que acompanham as práticas da empresa, os relatórios diários - que, com o próprio nome indica, é um relatório diário de cada colaborador da empresa, abrangendo as suas atividades ao longo do dia, as decisões tomadas, lições aprendidas, problemas que enfrenta e as atividades planeadas para fazer a seguir -, e um diagrama de gantt - que ao longo que o módulo é desenvolvido são apontadas todas as tarefas e atividades relacionadas com o mesmo, chegando ao final com um diagrama devidamente descritivo da dedicação a cada componente específico.

Os documentos do sistema podem ser divididos naqueles que são fundamentais para dar início ao DS, sendo criados antes desse marco. De referir que apesar de serem criados no início dos projetos podem sofrer alterações com o decorrer do mesmo, uma vez que podem ocorrer alterações no seu âmbito de produção ou de necessidades apresentadas. São eles o documento de requisitos que, para além dos requisitos funcionais e não funcionais identificados para o módulo, deverá também conter uma pequena descrição do módulo assim como os seus objetivos, e o documento da arquitetura do módulo onde é necessário descrever a arquitetura do módulo, mas também acrescentar representações gráficas. Estes dois documentos devem ser redigidos e armazenados na wiki do Redmine para que haja uma maior facilidade na sua gestão, como por exemplo quando for necessário proceder a alterações ou realizar o download dessas informações num ficheiro PDF, HTML ou TXT. Contudo, não é possível realizar a representação gráfica da arquitetura no Redmine, para contornar essa realidade é sugerido o uso do Draw.io, ou plataformas semelhantes, que permitam representar graficamente a solução.

Os outros dois documentos incluídos na documentação do processo surgem apenas durante o desenvolvimento do módulo, sendo eles a documentação API e de testes. A primeira é gerada através dos comentários presentes no código sendo inseridos ao longo da sua escrita, fazendo com que a sua altura de criação seja durante o desenvolvimento do módulo, para isso deverá ser usada a plataforma Swagger, à semelhança do que já acontece na atualidade, para criar e guardar documentação específica de API, que quando necessário partilhar com os clientes é gerado um link inserido na área do Magento destina ao cliente específico, esta prática deverá ser mantida, contudo deverá também ser criada uma ligação para esses documentos através do Redmine. A documentação de testes deverá ser redigida sempre que um teste seja efetuado, seja da primeira versão ou qualquer outra versão seguinte à mesma, usufruindo da wiki do Redmine, e à semelhança do documento de requisitos e da arquitetura, para descrever os testes e guardar os documentos que daí surgirem.

Os documentos de utilizadores resumem-se apenas ao manual de utilizador. Contudo, os manuais a serem criados devem ser de maior complexidade, sendo divididos em diferentes seções de informações necessárias que devem ser incluídas, já referidas acima. É de destacar que, apesar do ponto de elaboração dos manuais ser no final do DS, maior parte da informação a ser abrangida pelo manual deverá ser retirada de documentos criados anteriormente, podendo sofrer algumas alterações, e ainda novo conteúdo que será pertinente dar a conhecer aos clientes. Para isso, o seu caminho deve seguir os passos dos documentos anteriores, deverá ser utilizada a plataforma Redmine para facilitar a gestão do documento.

Para realizar uma gestão adequada de todos os documentos acima referidos é necessário ter em conta as suas especificidades e o modelo de negócio da empresa. O MoReq (DML 2010) é uma especificação de gestão de registos que descreve os requisitos modulares para um sistema de GD, todavia começou a ser utilizado como suporte à gestão de documentos, registos eletrónicos e para a gestão de conteúdo empresarial, em virtude da sua especificação ser do interesse de todas as pessoas dentro de uma organização que precisem de manter informações e registo atualizados, acessíveis a qualquer altura e por todos os intervenientes.

Este visa fornecer um conjunto de requisitos abrangente, mas de fácil compreensão que se destina a ser adaptável e aplicável a todo o tipo de informação e atividades, dentro de qualquer organização, e tendo esse princípio em vista a arquitetura tradicional foi originalmente baseada num único modelo de repositório central onde um sistema de registos autónomo de uma organização captura os registos tanto de fontes internas como externas, posto isto, inclui a captura de registos realizados noutro sistema de negócio e realiza a sua gestão centralizada num repositório controlado pelo sistema em causa. A arquitetura alternativa permite que o sistema de registos controle os registos já existentes dentro do sistema de negócio que eles são criados, ora em vez de duplicar os registos o sistema de arquivo partilha os registos. Por fim, a terceira arquitetura possível garante que os registos são incorporados no sistema de negócio, fazendo com que o sistema se torne simultaneamente um sistema de registos.

A arquitetura mais viável a implementar dentro da empresa em questão baseia-se na terceira apresentada, uma vez que os documentos são incorporados dentro do Redmine, permitindo realizar a sua captura e gestão. Na figura abaixo podemos ver uma representação dessa mesma arquitetura baseada na apresentada pelo MoReq.

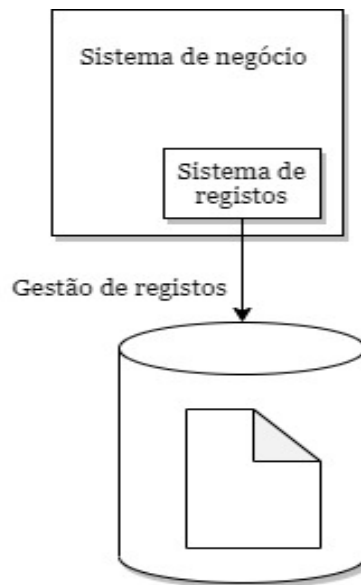


Figura 19 – Arquitetura do sistema de registros

Fonte: MoReq 2010

É de destacar que foi sugerida a utilização do Redmine para fazer a gestão de todos os documentos, em alguns casos os documentos serão elaborados diretamente na wiki da plataforma, porém existem casos em que é necessário o uso de diferentes plataformas para conceber a documentação, nestes casos deverá ser criada uma ligação entre essas plataformas com a wiki do Redmine. Para além do objetivo primordial de elaborar a documentação, o uso da wiki do Redmine tem também como foco realizar a gestão da mesma e para que isso seja possível é fundamental tirar proveito de todas as funcionalidades que a wiki oferece, como: criar novas páginas, adicionar novos ficheiros ou fazer ligação a ficheiros já existentes, ligação a páginas internas e externas, controlo de versões e problemas, adicionar anexos, gestão de repositório interno e possibilidade de ligar a diversos projetos e utilizadores, realizar um controlo de problemas, atividades e horas gastas relativamente a cada módulo que irá colmatar um diagrama de gantt.

Mas para que esta realidade seja possível é necessário ter um processo devidamente desenhado desde a criação dos documentos até ao momento em que são armazenados permanentemente ou eliminados. É possível ver a estrutura da utilização do Redmine que está a ser descrita no [anexo E](#).

Posto isto, é esperado que seja destacado um projeto na plataforma para cada módulo, assim torna-se possível inserir toda a documentação - do processo, do sistema e do utilizador - relativamente a cada módulo, sendo possível adicionar todos os intervenientes no seu

desenvolvimento, uma ligação a todos os projetos em que cada módulo está a ser utilizado, e como uma ligação bidirecional irá facilitar, por um lado aceder a todos os projetos que contêm o módulo, como através dos projetos aceder à documentação sempre que necessário. Para além disso, cada documento deverá receber metadados para que a sua recuperação seja facilitada, sendo eles: o nome, a versão do módulo, a data de criação, e uma pequena descrição.

O Redmine fornece uma série de funcionalidades através dos módulos que disponibiliza permitindo realizar um controlo dos seus projetos de uma forma abrangente. Mais concretamente realizar um controlo dos problemas que possam surgir através da indicação de “Issues” que podem ser levantados tanto por colaboradores como por clientes, controlar o tempo que é dispensado tanto a cada problema como a cada atividade destacada através de “Activity” e, por fim, elabora um diagrama gantt baseado nas informações registadas nos módulos anteriores, no campo destinado unicamente a essa funcionalidade com o nome “Gantt”, também irão facilitar a realização dos relatórios diários de cada colaborador uma vez que todas as suas atividades ficam registadas, assim como o tempo que dedicaram a cada uma. Através destas práticas é possível registar a documentação do processo, uma vez que todas as atividades que envolvem o desenvolvimento de cada módulo ficam registadas. Posto isto serão usados os módulos “Activity”, “Issues” e “Gantt” para a documentação do processo.

Para a documentação do sistema e do utilizador foi idealizado o uso dos módulos “Wiki”, “Documents” e “Files”, marcando a diferença entre os estados dos documentos referidos anteriormente. O primeiro será essencialmente utilizado para a elaboração dos documentos e para a realização de atualizações, abrangendo os estados de iniciação, preparação e estabelecimento –[anexo F](#)–, após serem aprovados passam pela fase utilização e revisão no módulo “Documents” onde serão divididos “User Documentation” e “Technical Documentation”, já por defeito da plataforma mas que vai de encontro ao tipo de documentos existentes –[anexo F](#)–, por fim será usado o módulo “Files” para questões de armazenamento dos documentos, destacado apenas para o estado de arquivo –[anexo F](#).

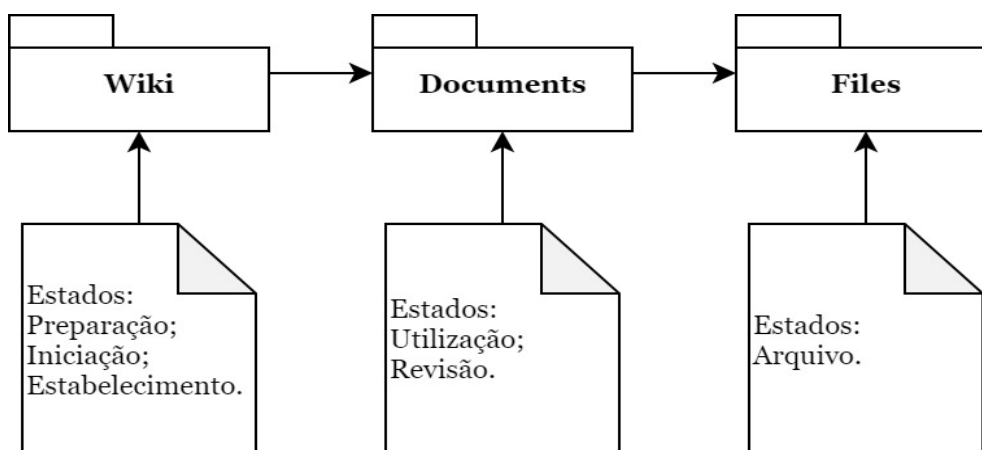


Figura 20 – Integração dos documentos no Redmine

Fonte: Autoria própria

Quando os documentos estão na fase de iniciação, preparação e estabelecimento são utilizados como via de comunicação entre os clientes e a empresa, uma vez que durante o projeto é significativo entregar documentos aos clientes, ainda que possam estar numa fase embrionária, para que se consiga identificar com mais especificidades os requisitos apresentados pelos clientes e controlar as expectativas dos mesmo em relação ao produto final. Já quando passam à fase utilização e revisão passam também a desempenhar um papel de suporte ao desenvolvimento do projeto, contudo as funções de comunicação que podem desempenhar sofrem ligeiras alterações. Nesta fase os documentos, mais concretamente o manual de utilizador, podem ser utilizados para angariar novos clientes através da sua divulgação, ou parte dele, para mostrar a possíveis futuros clientes os produtos e serviços que a empresa dispõe, como trabalham e como podem fazer o seu negócio mais inovador com os componentes que têm ou podem criar.

5.3 Validação

Idealmente a validação do processo passaria por duas fases, primeiro uma aprovação interna e, posteriormente, seria utilizado durante um desenvolvimento de um módulo com o objetivo de comprovar que a solução se enquadra na realidade da empresa. Contudo, não foi possível realizar estas duas fases, para contornar essa realidade foi efetuada a primeira fase já referida, e posteriormente foram criados exemplos dos documentos propostos para garantir que estão de acordo com o proposto. A validação interna foi sendo realizada ao longo do desenvolvimento

do trabalho através de conversas informais com os membros da organização. No final a solução foi apresentada onde foram sugeridas algumas alterações até à aprovação final.

Para dar seguimento à validação foram utilizadas as páginas criadas na wiki do Redmine como templates para cada documento que, juntamente com um módulo já acabado, serviram de exemplo dos documentos devidamente criados, tendo assim a possibilidade de comprovar que estes abrangem todo o conteúdo fundamental sobre o módulo. Posto isto, o módulo escolhi tem o nome de “Free Shipping Label”, esta escolha deve-se ao facto de já haver documentação sobre o módulo, tornando o processo de compreensão das suas funcionalidades mais rápido e eficaz, e pela possibilidade de realizar testes num ambiente de testes específico no âmbito de um projeto.

Inicialmente foi analisada a documentação já existente em relação ao módulo, sendo ela o documento de requisitos e um manual de utilizador em fase embrionária. Esta análise teve a finalidade de perceber a necessidade da sua criação, cada componente e as suas funcionalidades. Seguidamente procedeu-se à redação dos documentos propostos, a informação que já existente foi reutilizada e sempre que se revelou essencial foi acrescentada novo conteúdo, para conseguir obter novas informações foi consultada a pessoa responsável pelo desenvolvimento do módulo. Posteriormente, e para dar resposta nos documentos de teste, foi consultado um documento de plano de testes de um projeto específico para um cliente onde o módulo está disponível, e foi realizado um caso de teste no ambiente de testes criado com a finalidade de testar todas as funcionalidades da solução criada para o cliente em questão, e foi apenas realizado um teste ao módulo que está a ser utilizado. Contudo, é de referir que o teste não foi realizado com o objetivo de testar o módulo em si, mas para conseguir preencher o respetivo documento.

Em primeiro lugar foi criada a página inicial da wiki, inexistente até à data para os restantes projetos, onde podemos ver uma pequena descrição do módulo, a versão do mesmo (a versão colocada no exemplo é meramente ilustrativa não correspondendo à realidade), os documentos listados, e uma seção para inserir os projetos em que este está a ser utilizado (à semelhança da versão introduzida foi usado o mesmo projeto como exemplo apenas para testar a funcionalidade). No [anexo G](#) podemos ver uma imagem ilustrativa desta primeira página. De seguida serão apresentados os documentos criados.

- Documento de Requisitos

O documento de requisitos elaborado pode ser consultado no [anexo G](#), e conta com vários campos fundamentais para compreender o módulo que está a ser criado além dos requisitos funcionais e não funcionais. O que acontece na maioria dos casos é a inserção de uma pequena descrição do que é esperado do módulo e os requisitos funcionais e não funcionais. Em discordância com o que foi identificado na revisão de literatura existem muitas informações que não foram incluídas, contudo, a decisão de retirar essas informações deve-se ao facto de o documento estar a ser elaborado no Redmine e já se encontram por defeito inseridas na página do projeto, como por exemplo os intervenientes no projeto.

- Documento de arquitetura

Até então a elaboração deste documento não tinha sido pensada, e uma vez que assim acontece o documento pensado foi puramente inspirado no que fora outrora identificado na revisão de literatura, onde podemos ver uma visão geral do módulo, uma lista das suas funcionalidades e a representação gráfica do módulo. No [anexo G](#) podemos ver o presente documento.

- Documento de API

Este documento não foi criado por não se verificar a existência de uma API.

- Documentos de testes

Segundo o identificado previamente os documentos de testes surgem como resposta à necessidade de construção de um plano que garanta a qualidade dos produtos e/ou serviços. Inicialmente cada empresa necessita de ter explícita a sua estratégia de garantias de qualidade, no caso em estudo esta abordagem está ainda a ser pensada e posta em prática, baseando-se na elaboração de um plano de teste que dará origem aos casos de testes. Posto isto, foi seguido este planeamento, e no plano de teste temos informações relacionadas com possíveis cenários onde o módulo poderá ser utilizado, referindo o que é suposto acontecer e quando, quais as ações necessárias para realizar o teste e os resultados esperados, é possível ver o plano de teste para o módulo que foi usado para exemplificar no [anexo G](#), contudo o plano completo não é inserido uma vez que se trata de um projeto com um cliente. Com a realização do teste surge o documento de caso de teste, onde é apresentada uma descrição do teste realizado, os resultados esperados, os obtidos e informações adicionais que sejam relevantes, é possível ver um exemplo no [anexo G](#).

- Manual de utilizador

Contrariamente ao identificado na revisão de literatura, os documentos do utilizador não se baseiam apenas num manual, podendo ser vídeos, tutoriais, entre outros, foi apenas sugerido a elaboração de um manual de utilizador, que pode ser visualizado no [anexo G](#), sendo o único possível tendo em conta os recursos da empresa, nomeadamente colaboradores que se dediquem a essa prática. Os conteúdos sugeridos a incluir no manual vão, em grande escala, ao encontro do que já era incluído, contudo foi sugerido novas informações que são relevantes para um manual de utilizador, segundo o previamente identificado.

Existem outros documentos do sistema que foram identificados na revisão de literatura que são essenciais no DS, porém não foram considerados uma vez que a sua realização tem como base software, e o caso de estudo é apenas referente a módulos, ora componentes que integram projetos de maiores dimensões. Sendo eles o documento de design, onde é descrito o design do produto e as decisões tomadas sobre o mesmo, e o documento de manutenção, uma vez que a empresa oferece suporte e manutenção constante aos seus clientes. Para além desses documentos, temos os documentos do administrador inseridos nos documentos do utilizador, que também não foram propostos uma vez que não se enquadram na realidade da empresa. Por fim, temos os documentos do processo que podemos sumariá-la em relatórios diários que já é uma prática recorrente e em planos / estimativas / cronogramas que acabam por estar presente no diagrama de gantt, sendo a principal diferença é que à luz do identificado na revisão de literatura estes documentos são elaborados antes, e na solução criada eles vão sendo compostos ao longo do desenvolvimento do módulo.

Os documentos do processo não foram capturados figurativamente uma vez que a condição base para que isso acontecesse era a sua elaboração durante o desenvolvimento do módulo e este método de validação não foi possível.

6 Análise dos Resultados e Perspetivas futuras

Uma boa GC é fundamental para o crescimento de qualquer organização, quer seja através do conhecimento individual ou coletivo, implícito ou explícito, e para isso é crucial capturá-lo, organizá-lo, armazená-lo e partilhá-lo. Foi identificada uma necessidade crescente de capturar o conhecimento, o primeiro passo para que seja realizada uma gestão do mesmo, com a finalidade que este não desapareça, mas, pelo contrário, se transforme em CO. Este processo torna o trabalho de cada colaborador de uma empresa mais eficaz e eficiente, uma vez que todo o CO se encontra disponível e acessível para qualquer um e em qualquer momento, promovendo a partilha e criação de novos conhecimentos. Dentro de uma organização de DS, principalmente as que seguem metodologias ágeis para o desenrolar do seu trabalho, a comunicação e partilha são fundamentais, não só entre os colaboradores internos, mas também entre os clientes.

Ao longo do trabalho desenvolvido foi notória que a comunicação e partilha de conhecimento entre os membros da empresa é constante, contudo não existe a sua captura, o que acaba por se revelar prejudicial para a empresa, visto que se trata de uma organização que se divide em projetos, e quando existe a partilha do conhecimento acontece no âmbito de um projeto e apenas entre os membros que estejam destacados para o mesmo. Ao existir uma captura desse conhecimento é possível guardá-lo e torná-lo disponível a todos, para consulta de qualquer pessoa sempre que o considere pertinente.

Foi também perceptível a constante comunicação com o cliente, não só nas fases iniciais de análise e identificação de requisitos, mas também ao longo do desenvolvimento da solução e no ato contínuo de fornecimento de manutenção e suporte. Ao envolver os clientes em todas as fases do projeto tem como objetivo estabelecer uma relação com os mesmos, apoiada numa resposta imediata a questões que possam surgir e, posteriormente, em dar resposta a problemas que possam surgir. Deste modo, asseguram a fidelização e satisfação do cliente. Indo de encontro aos princípios expostos no manifesto agile, onde o processo de criação da solução é focado na colaboração com o cliente, para conseguir responder às mudanças, não só exigidas pelos clientes, mas também do mercado, fornecendo constantemente software funcional ou documentos que descrevam o produto em desenvolvimento.

Outro aspeto importante identificado durante o DS está diretamente relacionado com a GC. Um documento em ES não se destina apenas a referir aos manuais de utilizador ou documentação técnica exaustiva, mas sim a todo o material relacionado com o seu desenvolvimento, podendo ser trocas de emails com os clientes, anotações de colaboradores ou, por exemplo, um documento de requisitos. Em concordância com o identificado na revisão

de literatura, a prática de documentar ao longo do projeto ainda não se tornou uma prática comum. Contudo, os documentos têm a função de dar resposta a duas questões já indicadas anteriormente, sendo elas a comunicação, interna e/ou externa, e captar conhecimento para promover a aprendizagem interna. Posto isto, o seu propósito resume-se em tornar-se útil para todos os interessados, seja no decorrer do projeto ou por futuros utilizadores.

Ao analisar a documentação existente dentro da empresa foi observada uma grande lacuna entre o que foi identificado na revisão de literatura com a realidade, tanto a nível de quantidade de documentos, quanto ao nível de informações que constam nos mesmos e em relação aos tempos em que estes são criados. Após esta análise a solução criada foca-se em melhorar este processo de elaboração de documentos, contudo é de referir que o identificado na revisão de literatura quanto à documentação se destina a softwares de grandes dimensões, o trabalho desenvolvido foca-se apenas nos módulos de software, componentes que podem ser comercializados em separado, ou podem ser agregados numa solução mais complexa, traduzindo-se numa proposta de documentação em menor escala.

A solução criada prende-se ao objetivo de documentar os módulos de forma eficaz e eficiente, querendo isto dizer que é fundamental que cada documento seja elaborado em tempos devidos para que não haja perdas de conhecimento, e com conteúdo apropriado, não se transformando em documentação exaustiva, mas contendo todas as informações imprescindíveis. Para cada documento, neste caso específico os documentos do sistema, o conteúdo a incluir foi pensado de modo a tornar-se possível a sua reutilização para o manual de utilizador, ora documentação do utilizador, de modo que no final do projeto se reduzam os esforços na sua elaboração. Foram também pensados para poderem ser reutilizados em projetos de maior complexidade, ora quando um módulo é integrado numa solução desenhada especificamente para um cliente essa solução terá de ser acompanhada por toda a documentação, com a de cada módulo já elaborada são também poupados esforços uma vez que só se revela essencial a sua integração no resto da documentação, procedendo apenas a pequenas alterações, caso seja necessário.

Existem dois fatores, já identificados anteriormente, que podem beneficiar com a documentação dos módulos. O primeiro passa pela captura do conhecimento, ora quando os documentos são produzidos no tempo certo, a perda de conhecimento será menor, tornando possível que haja uma gestão do conhecimento focada na partilha e aprendizagem contínua. O segundo foca-se na comunicação com o cliente, através dos documentos torna-se possível mostrar ao cliente as funcionalidades que estão a ser pensadas integrar, realizando um levantamento de requisitos mais preciso, e gerir as expectativas dos clientes em relação ao produto final,

Para além de desempenharem um papel de via de comunicação com clientes atuais, podem também chegar a futuros clientes. Através da definição de conteúdo que possa ser partilhado sobre os seus produtos, neste caso os módulos, e da sua publicação online, no website da empresa, mais concretamente, podem, efetivamente, mostrar o seu trabalho e como agregar valor e inovação cativando novos clientes. Contudo, esta é uma questão a ser explorada no futuro, após a adequação de todo o processo de DS em conjunto com os documentos a serem criados.

Outra proposta interessante a explorar no futuro será a inserção de um plugin no Redmine que permita realizar uma GD. Existem já alguns disponíveis como: “Redmine documents” que permite reorganizar a visualização dos documentos tornando-a mais perceptível; “knowledgebase” dedicado à gestão do conhecimento dentro da GP; “Document Management System Features - DMSF” que se destina à substituição do módulo atual “Documents” adicionando uma série de novas funcionalidades como auditoria de acesso a documentos, configuração do fluxo de trabalho para aprovação do documento, pesquisa por conteúdo no texto completo, campos personalizados do documentos, entre outros. Porém, para conseguir proceder a instalação de plugins é necessário numa primeira instância verificar a versão que está a ser utilizada pelo Redmine, e para quais versões os plugins estão disponíveis.

Quanto às tecnologias utilizadas na empresa foi identificado um vasto leque de plataforma usadas para as mais diversas tarefas, contudo existe sempre a liberdade de inserir uma nova plataforma, caso esta se revele mais eficaz no trabalho a que se propõe do que aquela que está a ser utilizada na atualidade. Porém, é preciso referir que esta realidade apenas acontece com plataformas que estejam a ser utilizadas para algumas fases do projeto, como por exemplo, a escrita do código ou o design dos sites, não sendo obrigatória a utilização de uma só ferramenta, o que não acontece quando falamos na realização da gestão do projeto que é sempre usado o Redmine, ou a utilização do WHMCS para comunicar com o cliente. Posto isto, foi notória uma grande preocupação no uso das tecnologias mais adequadas, optando sempre por plataformas que sejam colaborativas, tentando tirar o melhor proveito das ferramentas existentes no mercado e daquilo que têm para oferecer.

7 Conclusões

A presente dissertação insere-se na área do DS, tendo sido realizado um caso de estudo de uma organização que se dedica a esse fim. Numa primeira instância foi fundamental realizar uma revisão de literatura direcionada para a GCC do em ODS com dois objetivos fulcrais, o primeiro dedica-se à identificação das boas práticas, e o segundo a alargar o conhecimento sobre o funcionamento deste tipo de organizações e se estão consciencializados para a importância das práticas anteriormente referidas ou não.

Posto isto, as principais conclusões a retirar da revisão de literatura consistem no crescente reconhecimento da importância de realizar uma GI ao longo de todo o processo de DS. Foi retratada a necessidade da existência de uma boa gestão do conhecimento para que este, quer seja explícito ou tácito, seja capturado, organizado, armazenado e partilhado para que este não se perca e, pelo contrário, se torne em CO, estando acessível a todos os membros da equipa promovendo a colaboração e autoaprendizagem. As ODS dividem-se em projetos, ora as pessoas são destacadas para cada projeto consoante as suas qualificações e, ao incorporar práticas de GC dentro deste tipo de organizações, permite que o conhecimento passe de pessoas para pessoas, independentemente do projeto em que estão inseridos, e que este não se perca quando o mesmo acabar. Em contrapartida e apesar deste reconhecimento, as melhorias que fazem nesse sentido focam-se apenas na adoção de novas tecnologias da informação, acabando por menosprezar as boas práticas.

Foram também analisadas metodologias para o DS, onde foi constatado que estas se dividem em métodos tradicionais e metodologias ágeis. A pesquisa realizada focou-se nas metodologias ágeis, mais concretamente no framework Scrum e XP, uma vez que são os seguidos pela empresa em questão. As metodologias ágeis seguem uma série de princípios que se podem basear em flexibilidade, divisão do trabalho, valorização da equipa, reavaliação constante do trabalho e cooperação com o cliente, seguindo a máxima de “documentar tanto quanto o necessário, o mínimo possível”. Contudo, segundo o estudo realizado foi identificada uma série de documentação que deve acompanhar o processo de desenvolvimento do software, tendo ela a função de descrever o produto e fornecer um guia para a sua utilização, abandonando a ideia de documentação extensa.

Após este primeiro estudo procedeu-se uma análise da empresa com o objetivo de reconhecer o processo de DS e os documentos produzidos detetando lacunas entre o identificado na revisão de literatura e o que acontece na realidade. Em consequência disso foi desenhado um processo para a GI de componentes de software, explicitando os tempos em que os documentos devem ser produzidos, assim como os conteúdos que devem ser inseridos em cada um deles.

Para colmatar esta solução foram pensadas questões relacionadas com a GD apropriada para a realidade da empresa.

Os objetivos definidos no início da presente dissertação tinham como meta o desenho de um processo para a elaboração de documentação aquando do DS de software e um plano de GI para esses mesmos documentos que ajudará a empresa na sua organização e recuperação de forma eficiente. Tendo isso em conta, podemos afirmar que o primeiro marco referido foi concebido, através de uma análise dos processos internos da empresa e das suas necessidades, contudo o segundo marco constituiu um maior desafio ficando ainda a requerer atenção por parte da empresa. Espera-se que o trabalho desenvolvido contribua para a fomentação da utilização das boas práticas na GI tornando mais perceptível que caminho deve ser seguido nesse sentido.

Ao longo do tempo foram encontradas algumas limitações que em grande parte se encontram associadas ao facto de ter sido um trabalho realizado remotamente. Dificultando alguns contactos com a mesma e, conseqüentemente, a análise do seu funcionamento, acabando por ser despendido mais tempo a esta tarefa que o previsto, uma vez que todo o trabalho a realizar estava dependente de uma análise da sua realidade para identificar possíveis melhorias.

Em suma, podemos afirmar que a realidade da empresa em estudo vai de encontro a muitos dos pontos abordados na revisão de literatura, como o funcionamento das organizações que se dedicam ao DS e da importância dada por estas à GI, assim como as poucas práticas adotadas nesse sentido. É também de destacar que esta necessidade surge das chefias, muitos dos restantes colaboradores não estão familiarizados com a área. Finalmente, acredita-se que a presente dissertação contribuiu para o envolvimento da área da CI na área de ES, retratando aspetos benéficos que podem ser conjugados com o intuito de alcançar melhorias nos processos e aumentar a produtividade tendo em vista fortalecer a vantagem competitiva no mercado de empresas em que a sua área de negócio se situa no DS.

Referência bibliográficas

AgileModeling. 2020. “Agile/Lean Documentation: Strategies for Agile Software Development.” 2020. <https://www.agilemodeling.com/essays/agileDocumentation.htm>.

Almseidin, Mohammad, Khaled Alrfou, Nidal Alnidami, and Ahmed Tarawneh. 2015. “A Comparative Study of Agile Methods: XP versus SCRUM.” *International Journal of Computer Science and Software Engineering (IJSSE)* 4 (5): 126–29. <http://ijcsse.org/published/volume4/issue5/p3-V4I5.pdf>.

AltexSoft. 2020. “Technical Documentation in Software Development: Types and Tools.” 2020. <https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/>.

Asana. 2021. “Asana.” 2021. <https://asana.com/pt>.

Atlassian. 2021. “Jira.” 2021. <https://www.atlassian.com/software/jira>.

Aurum, Aybüke, Farhad Daneshgar, and James Ward. 2008. “Investigating Knowledge Management Practices in Software Development Organisations - An Australian Experience.” *Information and Software Technology* 50 (6): 511–33. <https://doi.org/10.1016/j.infsof.2007.05.005>.

Beck, Kent. 2004. *Extreme Programming Explained, Second Edition*. Addison-Wesley Professional.

Bjornson, Finn Olav. 2007. *Knowledge Management in Software Process Improvement. Knowledge Engineering for Software Development Life Cycles*. <https://doi.org/10.4018/978-1-60960-509-4.ch015>.

Bjørnson, Finn Olav, and Torgeir Dingsøy. 2008. “Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts, Findings and Research Methods Used.” *Information and Software Technology* 50 (11): 1055–68. <https://doi.org/10.1016/j.infsof.2008.03.006>.

Cakir, C, F Cetin, M Savasci, and O Findik. 2018. “Software Architecture Documentation in Agile.” *International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18)*, no. December: 0–3.

Chomal, Vikas S, and Jatinderkumar R. Saini. 2015. "Significance of Software Documentation in Software Development Process." *International Journal of Engineering Innovation & Research* 3 (September): 410–16.

Clear, Tony. 2009. "ThinkingIssues." *Auckland University of Technology*.

Cunningham, Ward. 2021. "Manifesto for Agile Software Development." 2021. <https://agilemanifesto.org/>.

Holanda Rabelo, Jacilane De, Edson Cesar Cunha De Oliveira, Davi Viana Dos Santos, Luís Carlos Da Silva Braga, Gleison Dos Santos Souza, Igor Fabio Steinmacher, and Tayana Uchoa Conte. 2015. "Knowledge Management and Organizational Culture in a Software Organization - A Case Study." *Proceedings - 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015*, no. October: 89–92. <https://doi.org/10.1109/CHASE.2015.27>.

diagramas.net. 2021. "Draw Io." 2021. <https://www.diagrams.net/>.

DML. 2010. "Modular Requirements for Records Sytems." *DML Forum Foundation* 1.

Dorairaj, Siva, James Noble, and Petra Malik. 2012. "Knowledge Management in Distributed Agile Software Development." *Proceedings - 2012 Agile Conference, Agile 2012*, 64–73. <https://doi.org/10.1109/Agile.2012.17>.

Dybå, Tore, and Torgeir Dingsøyr. 2008. "Empirical Studies of Agile Software Development: A Systematic Review." *Information and Software Technology* 50 (9–10): 833–59. <https://doi.org/10.1016/j.infsof.2008.01.006>.

Feiz, Setareh, Zeinab Bte Khalifah, and Ali Ramezani Ghotbabadi. 2009. "Customer Relationship Management in Organizations." *International Journal on Information Sciences and Computing* 3 (1): 38–46. <https://doi.org/10.18000/ijisac.50039>.

Flora, Harleen, and Swati Chande. 2014. "A Systematic Study on Agile Software Development Methodologies and Practices." *International Journal of Computer Science and Information Technologies* 5 (3): 3626–37.

GitHub. 2021. "GitHub: Where the World Builds Software." 2021. <https://github.com/>.

GitLab. 2021. "Iterate Faster, Innovate Together | GitLab." 2021. <https://about.gitlab.com/>.

Grisham, Paul S, and Dewayne E. Perry. 2005. "Customer Relationships and Extreme Programming." *ACM SIGSOFT Software Engineering Notes* 30 (4): 1–6. <https://doi.org/10.1145/1082983.1083113>.

Hadar, Irit, Sofia Sherman, Ethan Hadar, and John J. Harrison. 2013. "Less Is More: Architecture Documentation for Agile Development." *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013 - Proceedings*, 121–24. <https://doi.org/10.1109/CHASE.2013.6614746>.

Hoda, Rashina, James Noble, and Stuart Marshall. 2012. "Documentation Strategies on Agile Software Development Projects." *International Journal of Agile and Extreme Software Development* 1 (1): 23. <https://doi.org/10.1504/ijaesd.2012.048308>.

IEEE, ISO/ IEC/. 2019. Systems and software engineering - Content of life-cycle information items (documentation), issued 2019.

Indumini, Udesika, and S. Vasanthapriyan. 2018. "Knowledge Management in Agile Software Development- A Literature Review." *2018 National Information Technology Conference, NITC 2018*, 1–7. <https://doi.org/10.1109/NITC.2018.8550066>.

IPQ. 2005. Informação e documentação Gestão de documentos de arquivo. Parte 1: Princípios directores, issued 2005.

IPQ. 2015. Gestão de documentos. Parte 1: Princípios e métodos, issued 2015.

Khalil, Carine, and Sabine Khalil. 2020. "Exploring Knowledge Management in Agile Software Development Organizations." *International Entrepreneurship and Management Journal* 16 (2): 555–69. <https://doi.org/10.1007/s11365-019-00582-9>.

KnowledgeBase Manager Pro. 2017. "Knowledge Management Software | Knowledge Base Software | KnowledgeBase Manager Pro." 2017. <http://www.web-site-scripts.com/knowledge-management/overview.html>.

Lesyuk, Andry. 2013. *Mastering Redmine*. Packt Publishing.

Loggem, Brigit Van. 2014. "Software Documentation: A Standard for the 21st Century." *ACM International Conference Proceeding Series*, 149–54. <https://doi.org/10.1145/2618168.2618192>.

- Maciel, Claudia P.C., Érica Ferreira de Souza, Ricardo de Almeida Falbo, Katia Romero Felizardo, and Nandamudi L. Vijaykumar. 2018. "Knowledge Management Diagnostics in Software Development Organizations: A Systematic Literature Review." *ACM International Conference Proceeding Series*, 141–50. <https://doi.org/10.1145/3275245.3275260>.
- Maric, Mirjana, Predrag Matkovic, Pere Tumbas, and Veselin Pavlicevic. 2016. "Documenting Agile Architecture: Practices and Recommendations." *Lecture Notes in Business Information Processing* 264 (September): 56–71. https://doi.org/10.1007/978-3-319-46642-2_4.
- Microsoft. 2021. "SharePoint." 2021. <https://www.microsoft.com/pt-pt/microsoft-365/sharepoint/collaboration>.
- Microsoft. 2021. "Visual Studio Code." 2021. <https://code.visualstudio.com/>.
- Microsoft. 2021. "Microsoft Project | Software de Gestão de Projetos." 2021. <https://www.microsoft.com/pt-pt/microsoft-365/project/project-management-software?market=pt>.
- Mishra, Alok, and Deepti Mishra. 2013. "Software Project Management Tools." *ACM SIGSOFT Software Engineering Notes* 38 (3): 1–4. <https://doi.org/10.1145/2464526.2464537>.
- Monday. 2021. "Monday.Com." 2021. <https://monday.com/lang/pt/>.
- Montoni, Mariano, Gleison Santos, Karina Villela, Rodrigo Miranda, Ana Regina Rocha, Guilherme H. Travassos, Sávio Figueiredo, and Sômulo Mafra. 2004. "Knowledge Management in an Enterprise-Oriented Software Development Environment." *Federal University of Rio de Janeiro*.
- Nonaka, Ikujiro. 1994. "A Dynamic Theory of Organizational Knowledge Creation." *Organization Science* 5 (1): 14–37. <https://doi.org/10.1287/orsc.5.1.14>.
- Pohl, Klaus. 2010. "Requirements Engineering by Klaus Pohl." Springer.
- Priya, M.A. Krishna, and Justus Selwyn. 2016. "An Exploratory Study on the Practices of Knowledge Management in Software Development Organizations." *International Journal of Business Intelligents* 5 (1): 1–7. <https://doi.org/10.20894/ijbi.105.005.001.001>.
- Razzak, Mohammad Abdur, and Rajib Ahmed. 2014. "Knowledge Sharing in Distributed Agile Projects: Techniques, Strategies and Challenges." *2014 Federated Conference on Computer*

Science and Information Systems, FedCSIS 2014 2: 1431–40.
<https://doi.org/10.15439/2014F280>.

Redmine. 2014. “Redmine.” 2014. <https://www.Redmine.org/>.

Runeson, Per, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *In Software Engineering in Software. Agile Processes in Software Engineering and Extreme Programming*.

Saigon Technology. 2021. “Agile Vs Waterfall in Software Development .” 2021.
<https://saigontechnology.com/blog/agile-vs-waterfall-in-software-development>.

Santos, Gleison, Karina Villela, Mariano Montoni, Ana Regina Rocha, Guilherme H. Travassos, Sávio Figueiredo, Sômulo Mafra, Adriano Albuquerque, Benito Diaz Paret, and Márcio Amaral. 2005. “Knowledge Management in a Software Development Environment to Support Software Processes Deployment.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3782 LNAI (Wm): 111–20. https://doi.org/10.1007/11590019_14.

Shen, Yuanyanhang, and Xiaodan Yu. 2019. “Project Management Tools as Boundary Objects in Agile Software Development.” *Association for Information Systems AIS Electronic Library*.

SmartBear Software. 2021. “API Documentation & Design Tools for Teams | Swagger.” 2021.
<https://swagger.io/>.

Sonatype. 2021. “Nexus Repository | Software Component Management.” 2021.
<https://www.sonatype.com/products/repository-pro>.

Souri, Alireza, Parvaneh Asghari, and Reza Rezaei. 2017. “Software as a Service Based CRM Providers in the Cloud Computing: Challenges and Technical Issues.” *Journal of Service Science Research* 9 (2): 219–37. <https://doi.org/10.1007/s12927-017-0011-5>.

Souri, Alireza, Amir Masoud Rahmani, Nima Jafari Navimipour, and Reza Rezaei. 2019. “Formal Modeling and Verification of a Service Composition Approach in the Social Customer Relationship Management System.” *Information Technology and People* 32 (6): 1591–1607.
<https://doi.org/10.1108/ITP-02-2018-0109>.

t2informatik. 2019. “Agile Documentation in Software Development.” 2019.
<https://t2informatik.de/en/blog/software-development/agile-documentation-in-software-development/>.

Tracy, Brian, Eat That Frog, and Time Power. 2014. *More Praise for Scrum: The Art of Doing Twice the Work in Half the Time*. 1º. Crown Business.

Trello. 2021. “Trello.” 2021. <https://trello.com/home>.

Vanhala, Erno, and Jussi Kasurinen. 2019. *The Role of the Customer in an Agile Project: A Multi-Case Study. Lecture Notes in Business Information Processing*. Vol. 370 LNBIP. Springer International Publishing. https://doi.org/10.1007/978-3-030-33742-1_17.

Weinreich, Rainer, and Iris Groher. 2016. “Software Architecture Knowledge Management Approaches and Their Support for Knowledge Management Activities: A Systematic Literature Review.” *Information and Software Technology* 80: 265–86. <https://doi.org/10.1016/j.infsof.2016.09.007>.

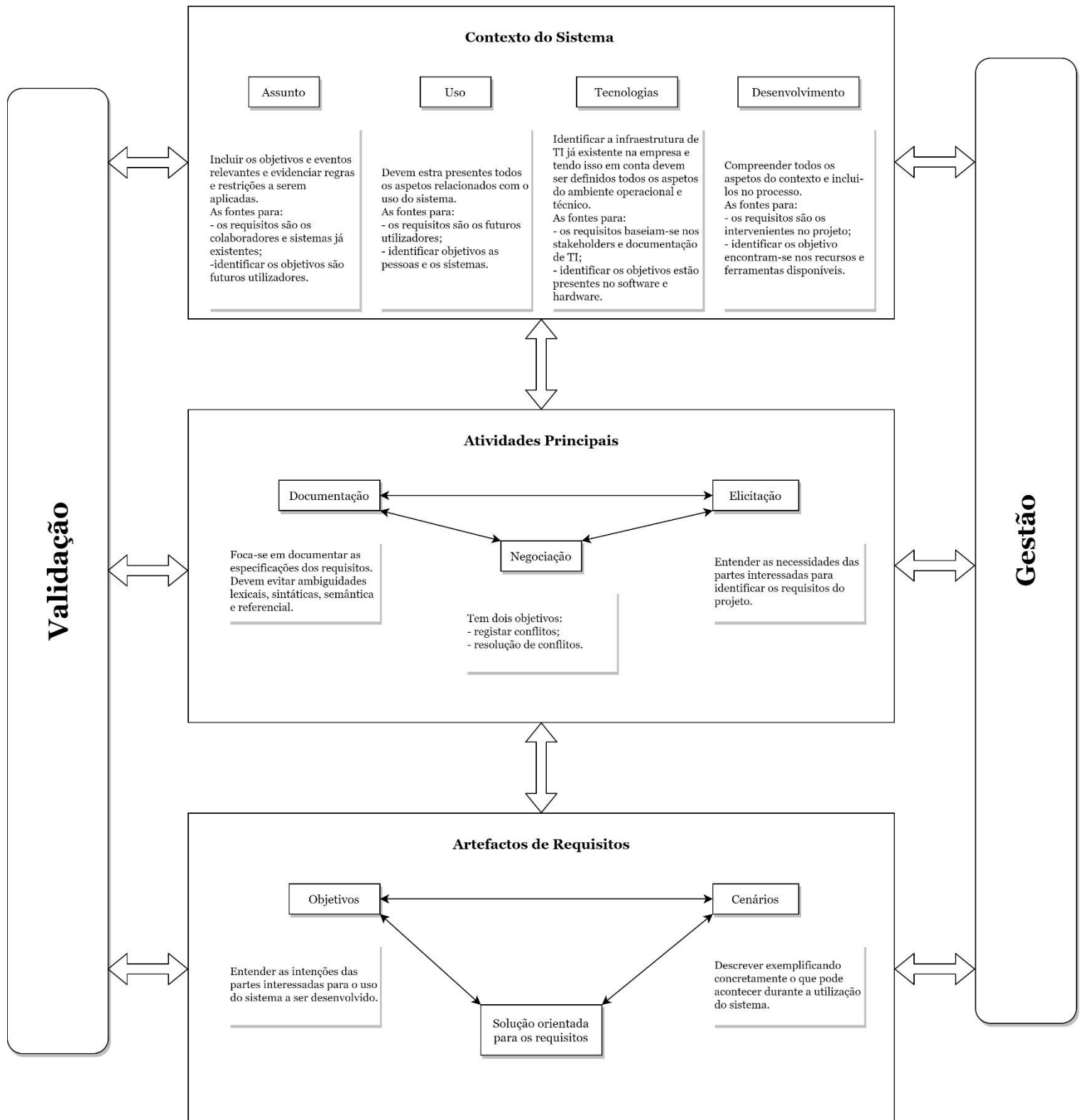
WHMCS. 2021. “WHMCS | Web Hosting Billing & Automation Platform.” 2021. <https://www.whmcs.com/>.

Wikipédia. 2021. “Wiki.” 2021. <https://pt.wikipedia.org/wiki/Wiki>.

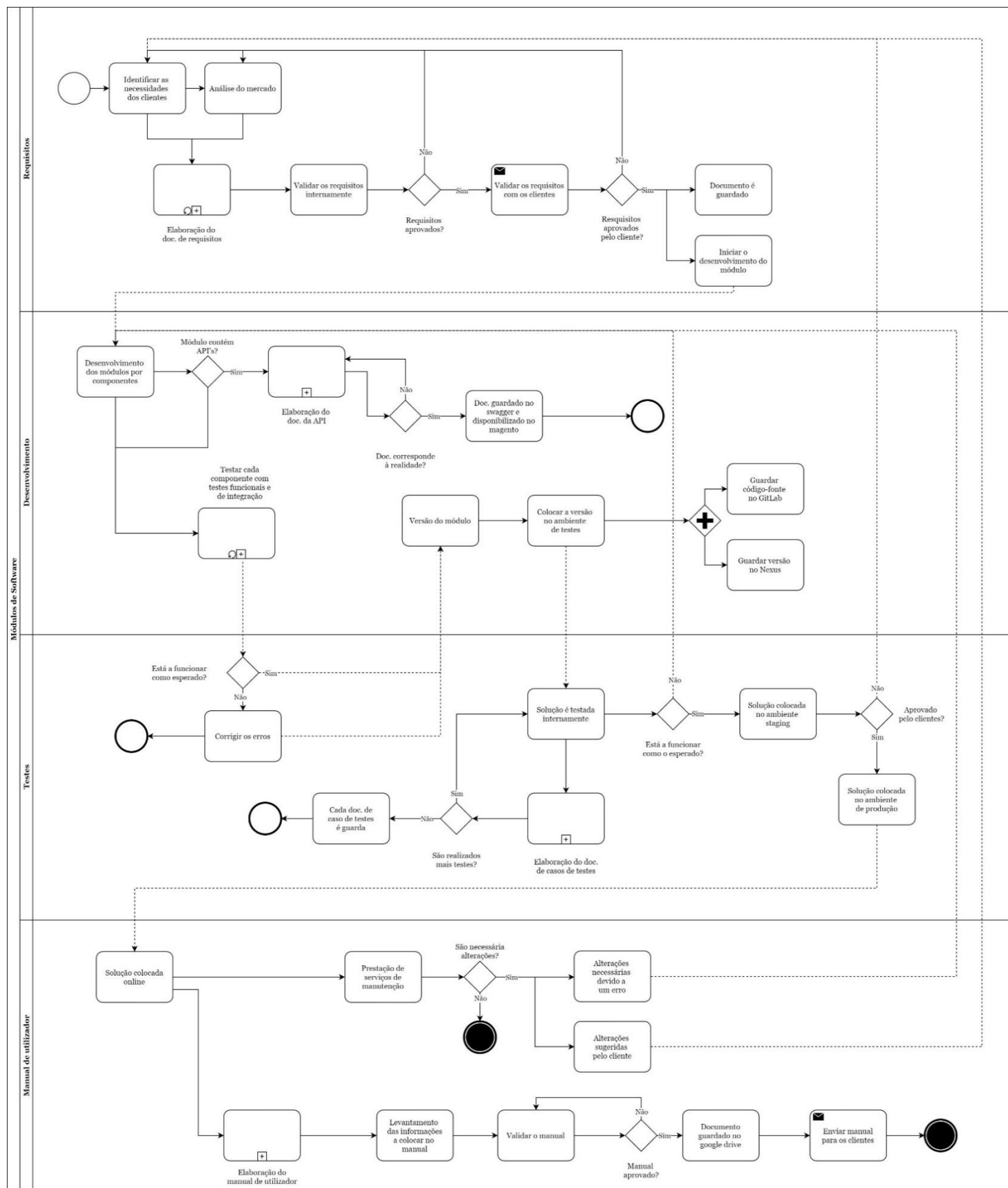
Yu, Xiaodan, and Stacie Petter. 2014. “Understanding Agile Software Development Practices Using Shared Mental Models Theory.” *Information and Software Technology* 56 (8): 911–21. <https://doi.org/10.1016/j.infsof.2014.02.010>.

Anexos

Anexo A: Abordagem metodológica



Anexo B: Processo de desenvolvimento dos módulos de software



Anexo C: Proposta de documentos a serem criados

Documento de Requisitos

Nome do Módulo

Introdução

Qual a necessidade de criação do módulo, a sua finalidade e que valor pode agregar.

Contexto

Descrever quais os objetivos do módulo, as suas características e as decisões tomadas sobre as funcionalidades a inserir.

Descrição do módulo

Visão geral do módulo, funcionalidades e restrições que podem levar ao não funcionamento das ações pensadas.

Identificação de requisitos

h3.Requisitos funcionais

Listar os requisitos funcionais.

- 1º Requisito;
- 2º Requisito;
- (...)

h3.Requisitos não funcionais

Listar os requisitos não funcionais.

- 1º Requisito;
- 2º Requisito;
- (...)

Critérios de aceitação

Critérios para se conseguir obter o módulo.

[Novo ficheiro](#)

Documento de Arquitetura

Documento de Arquitetura

Descrição da arquitetura

Descrever a solução que será proposta.

Funcionalidades

Detalhar todas as funcionalidades do módulo.

[Link para diagramas com as funcionalidades](#)

[Novo ficheiro](#)

Também disponível em: [PDF](#) | [HTML](#) | [TXT](#)

Documento de API

Documento API

Descrição

Descrição do API.

Operações

Detalhar especificamente cada operação incluída com a API.

Link para o documento API gerado através do swagger

[Novo ficheiro](#)

Também disponível em: [PDF](#) | [HTML](#) | [TXT](#)

Documento de Testes

Documentos de Testes

Plano de testes

O plano de testes apresenta num possível cenário é referido o que é suposto acontecer e quando, as ações necessárias e os resultados esperados.

Links para o plano de testes completo

Casos de teste

Descrever o teste feito, apresentar os resultados esperados e os obtidos.

- [Teste 1](#)
- [Teste 2](#)
- (...)

[Novo ficheiro](#)

Também disponível em: [PDF](#) | [HTML](#) | [TXT](#)

Caso de Teste

Teste 1

Descrição do teste

Pequena descrição do teste realizado.

Resultados esperados

Apresentar os resultados esperados com a realização do teste.

Resultados obtidos

Apresentar os resultados obtidos com a realização do teste.

Informação adicional

Informação adicional pretinente sobre o teste realizados e conclusões retiradas.

[Novo ficheiro](#)

Também disponível em: [PDF](#) | [HTML](#) | [TXT](#)

Manual de Utilizador

Nome do Módulo

Introdução

Descrição do módulo.

Características

Características base e inovadoras do módulo e os seus benefícios

Requisitos

Apresentar os requisitos necessários para adquirir o módulo.

Guias de funcionalidades

Listar todas as funcionalidades do módulo.

Configuração

Como configurar o módulo.

Instalação

Processo para instalação do módulo.

Possíveis problemas

Possíveis problemas que podem surgir e proposta de resolução.

Atualização

Explicar o processo de atualização do módulo.

Questões frequentes

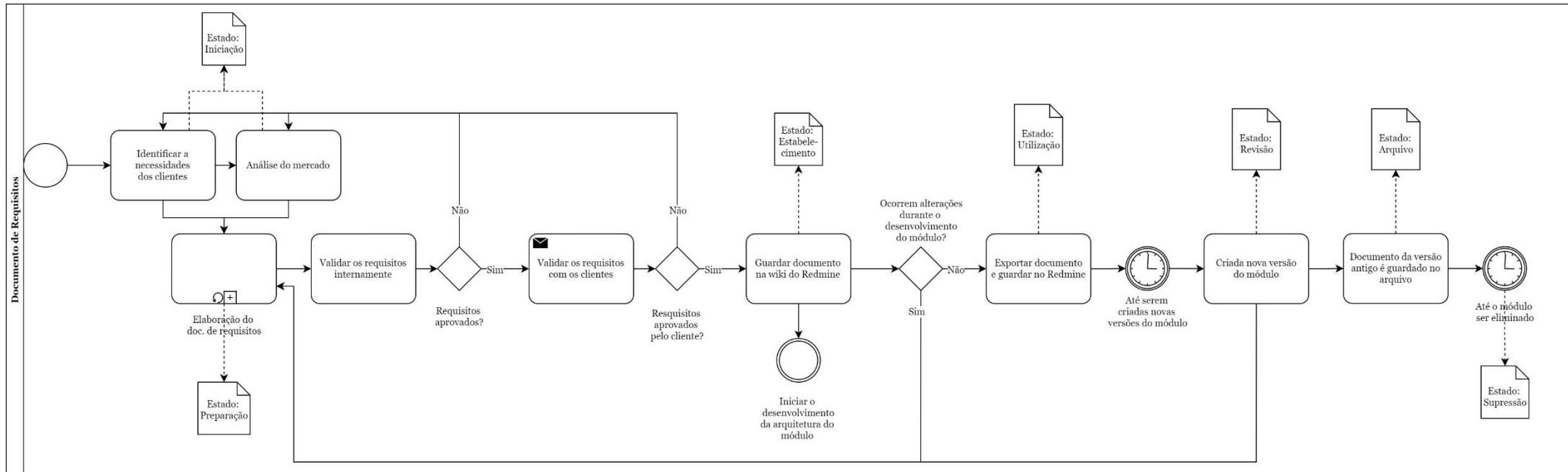
Apresentar questões que possam surgir.

Novo ficheiro

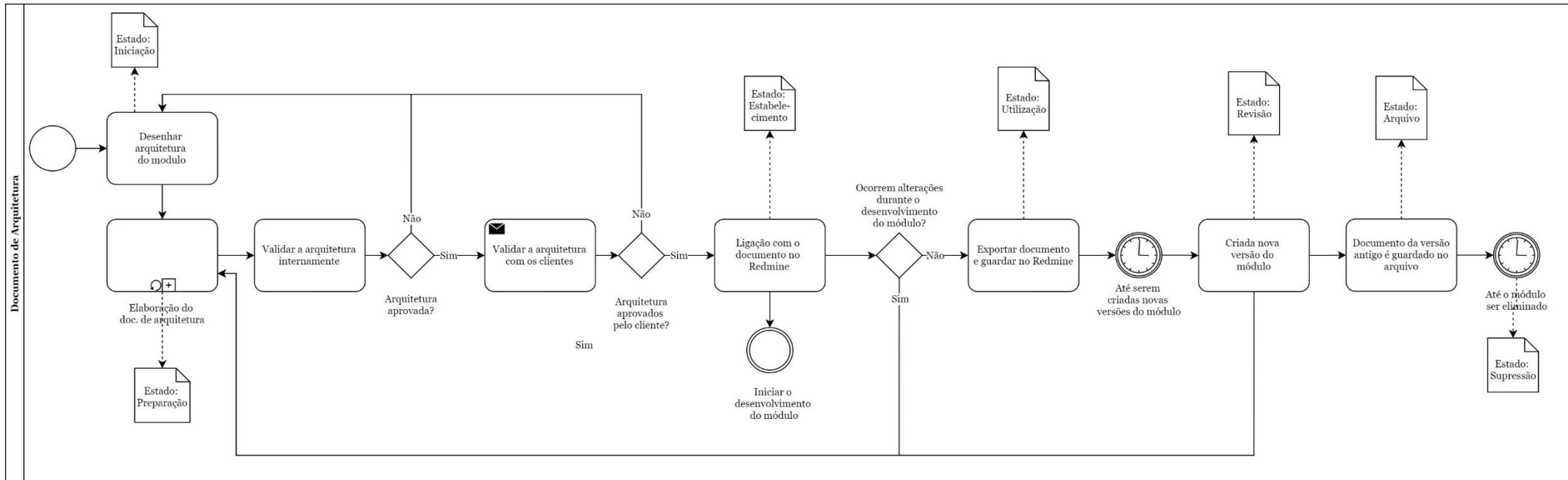
Também disponível em: [PDF](#) | [HTML](#) | [TXT](#)

Anexo D: Processos para o desenvolvimento de documentos

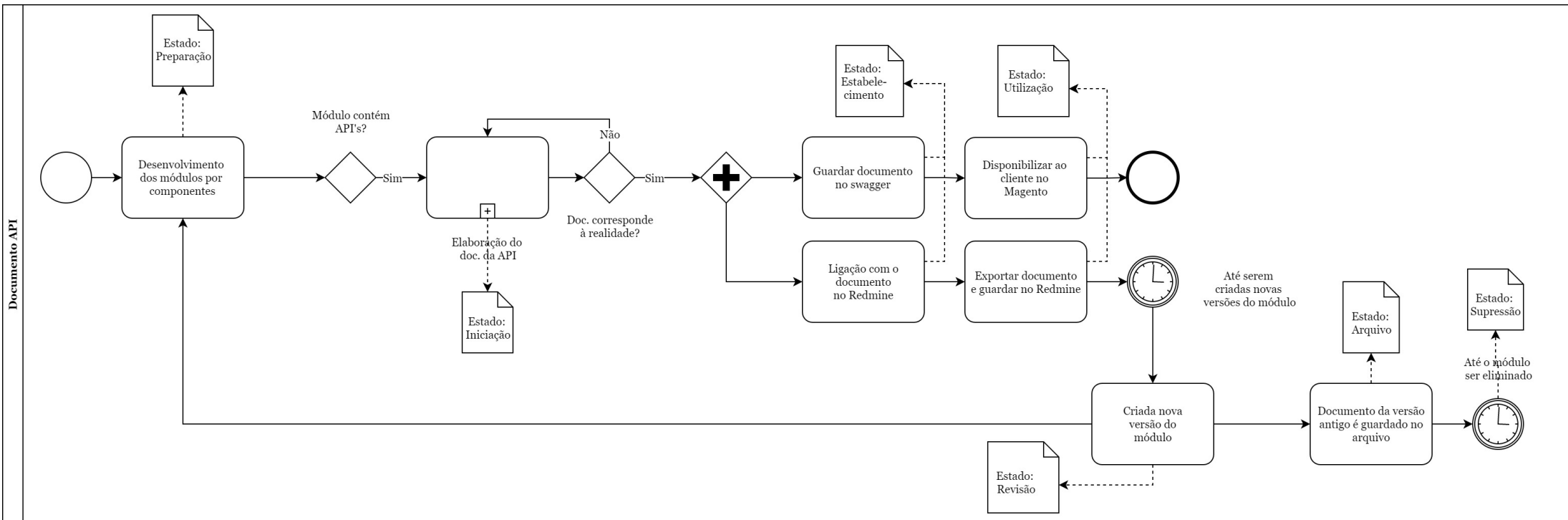
Documento de Requisitos



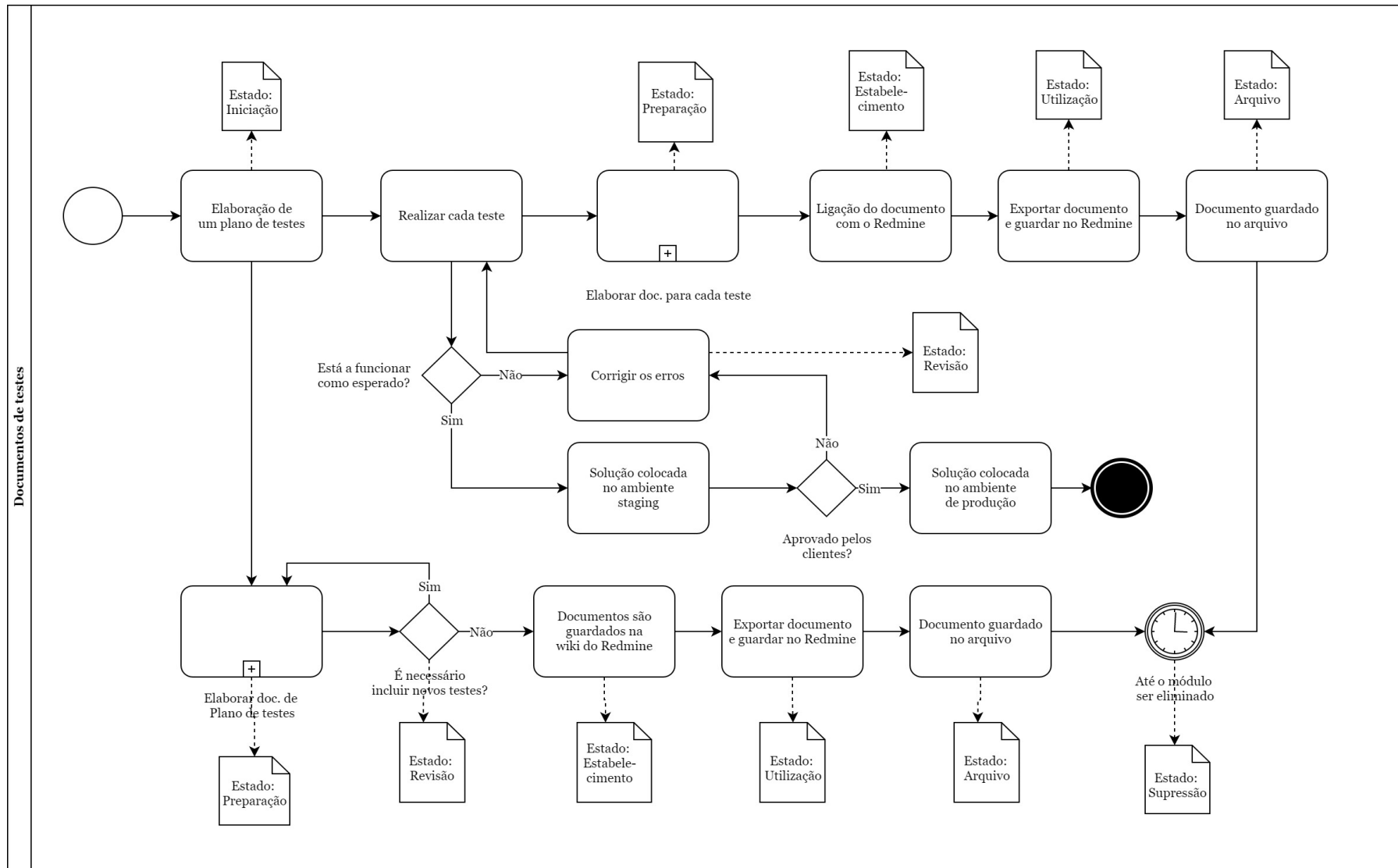
Documento de Arquitetura



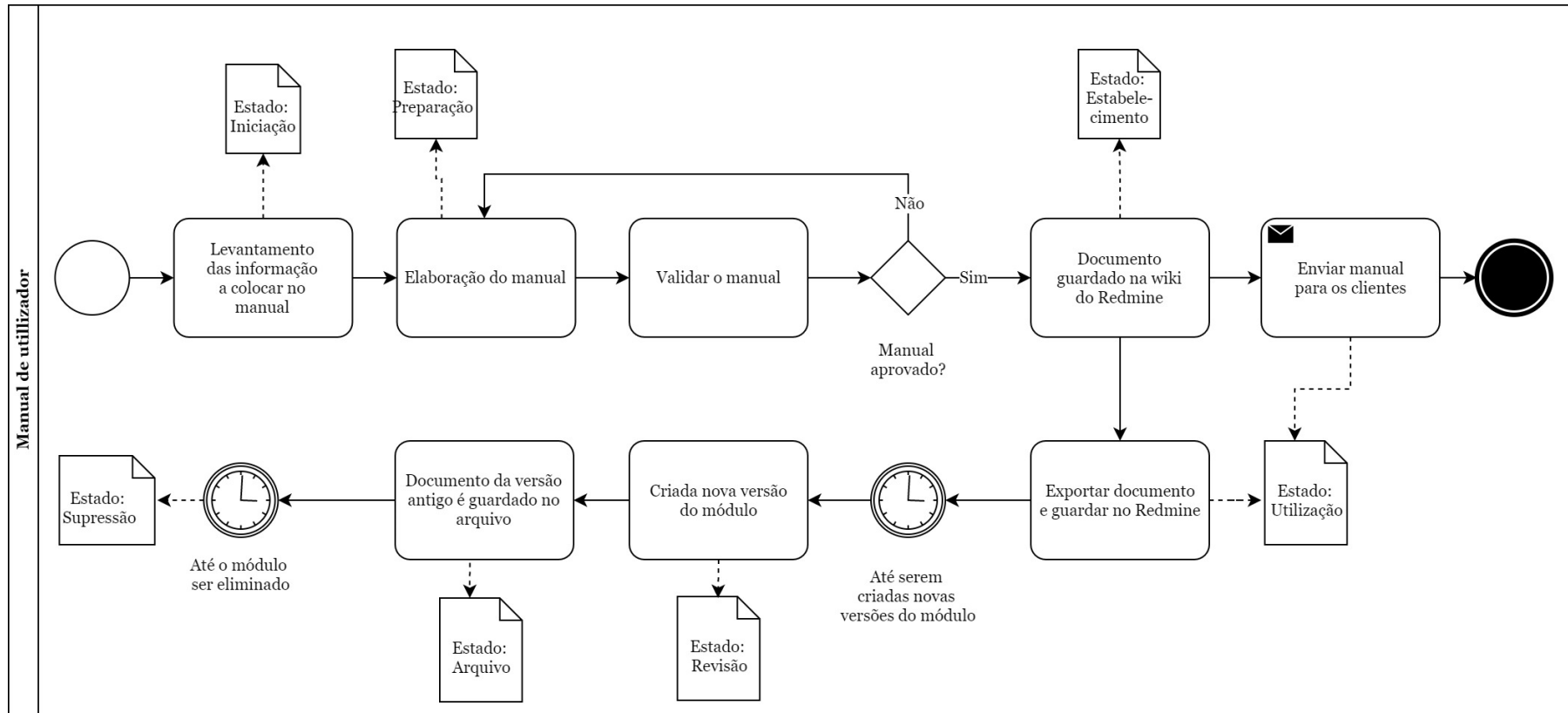
Documento de API's



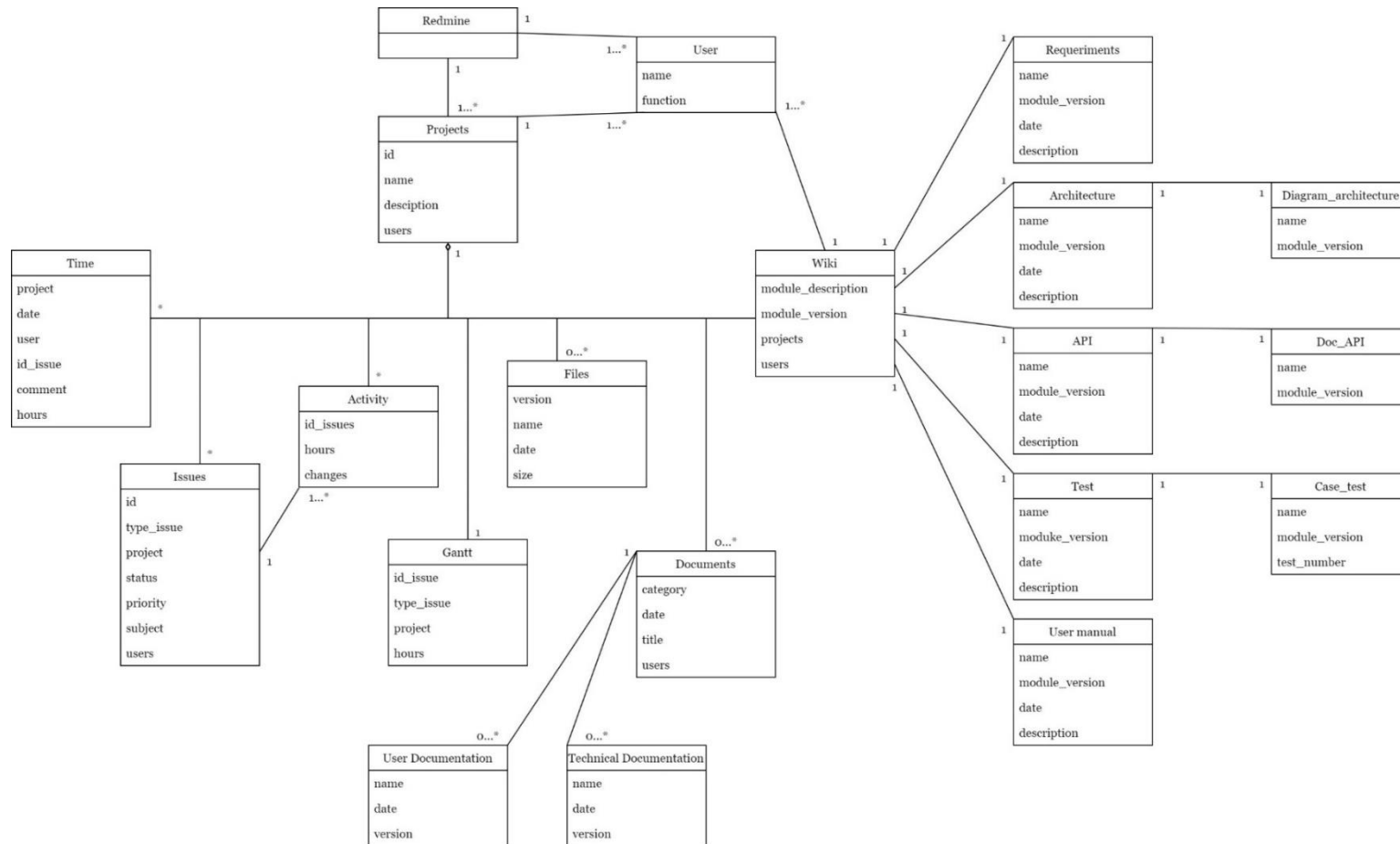
Documentos de Testes



Manual de Utilizador



Anexo E: Integração da solução no Redmine



Anexo F: Exemplo da integração no Redmine

Módulo “Wiki”

Nome do Módulo

Descrição do módulo

Versão

Link para o módulo

Documentação do módulo

Projetos Associado:

Exemplo

Novo ficheiro

✎ Editar

★ Observar

🔒 Bloquear

🔥 Renomear

🗑 Apagar

🕒 Histórico

✎

Inserir uma breve descrição do módulo.

Versão do módulo.

Documentos de Requisitos

Documentos de Arquitetura

Documento API

Documentos de Testes

Manual de Utilizador

Também disponível em: PDF | HTML | TXT

Wiki

Página inicial

Índice por título

Índice por data

Módulo “Documents”

Documentos

Novo documento

User documentation

Manual de Utilizador

18/06/2021 20:02

Technical documentation

Requisitos

18/06/2021 20:02

Arquitetura

18/06/2021 20:03

API

18/06/2021 20:03

Testes

18/06/2021 20:04

Ordenar por

Categoria

Data

Título

Autor

Módulo “Files”

Ficheiros						Novo ficheiro
#	Ficheiro	Data	Tamanho	D/L	MD5	
#2	Documentos_de_Testes.pdf	18/06/2021 20:05	25,1 KB	0	a68a2e238ca7816a59a8bce5700ccac6	🗑
	Documento_API.pdf	18/06/2021 20:05	25,1 KB	0	ba2e82095b310c787a3692c7b78d012e	🗑
	Documento_de_Arquitetura.pdf	18/06/2021 20:05	25,1 KB	0	13b18c3c653e9d8a1c91f4f187b82491	🗑
	Documento_de_Requisitos.pdf	18/06/2021 20:05	25,1 KB	0	c7c42931194b78be9d7b6e908faae937	🗑
	Manual_de_Utilizador.pdf	18/06/2021 20:05	25,1 KB	0	552521204e1cf4c7b2be31cdcf83fb67	🗑
#1	Documentos_de_Testes.pdf	18/06/2021 19:59	25,1 KB	0	a68a2e238ca7816a59a8bce5700ccac6	🗑
	Documento_API.pdf	18/06/2021 19:59	25,1 KB	0	ba2e82095b310c787a3692c7b78d012e	🗑
	Documento_de_Arquitetura.pdf	18/06/2021 19:59	25,1 KB	0	13b18c3c653e9d8a1c91f4f187b82491	🗑
	Documento_de_Requisitos.pdf	18/06/2021 19:58	25,1 KB	0	c7c42931194b78be9d7b6e908faae937	🗑
	Manual_de_Utilizador.pdf	18/06/2021 20:00	25,1 KB	2	552521204e1cf4c7b2be31cdcf83fb67	🗑

Anexo G: Exemplo de documentos

Página Inicial

Free Shipping Label ¶

 Editar  Observar  Bloquear  Renomear  Apagar  Histórico

Descrição do módulo

Este módulo permite definir regras gerais, sem ser necessário definir a mesma condição para vários produtos, um a um. Com este módulo é possível definir as regras no backoffice e a forma como elas se apresentam no frontend - no carrinho quando este responde às regras, e na página do produto, quando este responde às mesmas regras.

Versão

versão 3.1.4

[Link para o módulo](#)

Documentação do módulo

- [Documento de Requisitos](#)
- [Documento de Arquitetura](#)
- [Documento API](#)
- [Documentos de Testes](#)
- [Manual de Utilizador](#)

Projetos Associados:

[Free shipping label](#)

Documento de Requisitos

Free Shipping Label

Introdução

Este módulo apresenta no Shopping Cart e no MiniCart o valor em falta bem como o peso máximo para se obter o free shipping. Também é possível apresentar na página e no widget do produto se o produto em questão é ou não qualificado para o free shipping.

Contexto

Para um produto ser qualificado para o free shipping é necessário ter em consideração os seguintes aspetos:

- Como é definido o valor e o peso do free shipping:
 - Tanto o valor como o peso são definidos na configuração do módulo. Após a ativação do módulo adiciona-se o "minimum order amount" e o "maximum weight" sendo estes o valor e o peso máximo do free shipping respetivamente.
- Cálculo do valor em falta para o free shipping:
 - O valor restante para o free shipping é calculado através da subtração do valor definido para o free shipping pela soma dos preços dos produtos adicionados ao carrinho de compras.
- Peso máximo para o free shipping:
 - A soma dos pesos dos produtos ao carrinho adicionados não pode ser superior ao peso definido para o free shipping.

Descrição do módulo

A partir do momento que o preço do produto é igual ou superior ao valor definido para se obter o free shipping e o peso do produto for igual ou inferior ao peso definido nas configurações do módulo, o produto passa a ser qualificado.

Identificação de requisitos

Requisitos funcionais

- Ativar ou desativar o módulo através do back office;
- Escolher no back office apresentar ou não na página e no widget do produto se o produto em questão é qualificado para o free shipping;
- Apresentar no Shopping Cart e no MiniCart o valor e o peso necessário para se obter o free shipping;
- Escolher no back office o preço mínimo e o peso máximo para se obter o free shipping.

Requisitos não funcionais

- Funcionar na versão igual ou superior a 2.3.2 do Magento.

Critérios de aceitação

Para usufruir deste módulo, é necessário que tenha a sua loja Magento na versão 2.3.4 ou superior e o módulo instalado.

Documento de Arquitetura

Free Shipping Label

Descrição da arquitetura

Este módulo permite adicionar um novo método de envio, no qual os portes de envio são grátis, quando o(s) produto(s) do carrinho correspondem às regras definidas em backoffice.

Para sinalizar quando um produto se qualifica para o envio grátis, o módulo permite adicionar uma label na página de produto e no mini carrinho.

Funcionalidades

- Definir um valor a partir do qual é oferecido o envio.
- Permite adicionar esta informação em forma de barra, no carrinho e na página do produto.
- É possível definir um valor máximo de peso, em kg, para o(s) produto(s).
- Permite esconder, do cliente final, outros métodos de envio quando o carrinho cumpre todas as condições.

[Link para diagrama com as funcionalidades](#)

Documento de Testes

Free Shipping Label

Plano de testes

Foi realizado um teste num ambiente de testes no âmbito de um projeto específico com clientes.

É esperado que quando o cliente acede ao site apareça a barra a informar que os portes de envio são grátis quando o valor da compra é superior a X.

Quando este começa a sua compra é apresentada uma barra que indica o progresso entre o valor existente no carrinho e o valor mínimo para se obter portes grátis.

[Link para o plano de testes completo](#)

Casos de teste

Descrever o teste feito, apresentar resultados esperados e obtidos.

- [Teste 1](#)

Caso de teste

Teste 1

Descrição do teste ¶

No ambiente de testes são testadas todas as funcionalidades do módulo.

Resultados esperados

- Apareça uma barra informativa a dizer a partir de qual valor os portes são grátis.
- Ao longo da compra é esperado que a barra indique o progresso entre o valor existente no carrinho e o valor mínimo para se obter portes grátis.

Resultados obtidos

- Na página inicial aparece a barra informativa.
- Ao longo da compra a barra indica o progresso entre o valor existente no carrinho e o valor mínimo para se obter portes grátis.

Informação adicional

O módulo está a funcionar corretamente.

Manual de Utilizador

Free Shipping Label

Introdução

O nosso módulo "Skrey Free Shipping (Label)" permite adicionar um novo método de envio, no qual os portes de envio são grátis, quando o produto (ou os produtos) do carrinho correspondem às regras definidas em backoffice.

Para sinalizar quando um produto se qualifica para o envio grátis, o módulo permite adicionar uma label na página de produto e no mini carrinho.

Este módulo permite definir regras gerais, sem ser necessário definir a mesma condição para vários produtos, um a um. Com este módulo é possível definir as regras no backoffice e a forma como elas se apresentam no frontend - no carrinho quando este responde às regras, e na página do produto, quando este responde às mesmas regras.

Características

O módulo "Skrey Free Shipping (Label)" permite definir regras, no backoffice do Magento, de forma a definir um método de envio com o envio gratuito quando algum produto ou algum carrinho cumpre as condições definidas.

Este módulo apresenta no Shopping Cart e no MiniCart o valor em falta bem como o peso máximo para se obter o free shipping. Também é possível apresentar na página e no widget do produto se o produto em questão é ou não qualificado para o free shipping.

Esta informação pode ser encontrada de várias formas, no frontend: pode vir discriminada na página de produto, quando este cumpre os requisitos todos e se qualifica para o envio gratuito. Pode vir no carrinho de compras, e, neste caso, quando um produto não cumpre todos os requisitos, a label que informa do free shipping informa, também, de quanto falta, em valor monetário, para que se qualifique para este método de envio.

As condições que se podem definir neste módulo são: o valor mínimo de um produto/carrinho para que seja elegível para o envio grátis, e o peso máximo que um produto pode ter para se qualificar para este método de envio.

Outra das funcionalidades deste módulo é que este oferece a possibilidade de ocultar alguns métodos de envio que costumam ser visíveis para o consumidor final, quando o carrinho cumpre todas as condições para usufruir do método de envio grátis.

Requisitos

Para usufruir deste módulo, é necessário que tenha a sua loja Magento na versão 2.3.4 ou superior e o módulo instalado.

O módulo "Skrey Free Shipping (Label)" é compatível com as versões de Magento 2.3.2 e superiores. Este módulo encontra-se testado nas versões 2.3.2, 2.3.4 e 2.3.5.

Guia de funcionalidades

O módulo "Skrey Free Shipping (Label)" permite definir um valor a partir do qual é oferecido o envio grátis do(s) produto(s) do carrinho, através das regras estipuladas em backoffice, que são aplicadas na frente de loja.

Juntamente com esta funcionalidade, o módulo permite adicionar esta informação, em forma de label, no carrinho e na página do produto, quando este respeita as regras e é elegível para o envio grátis.

lém do valor monetário que é possível definir, com o "Skrey Free Shipping (Label)" é possível estipular, também, um valor máximo de peso, em Kg, para que o produto, ou o total de produtos do carrinho, sejam elegíveis para o free shipping.

Ou seja, com este módulo, o utilizador pode definir, em backoffice, as seguintes regras: o valor mínimo que um carrinho deve ter e o peso máximo que um produto deve ter para que se qualifique para o método de envio com portes grátis.

Este módulo também permite esconder, do cliente final, métodos de envio quando o carrinho cumpre todas as condições já definidas e se qualifica para o free shipping.

Configuração

No backoffice da loja em Magento, quando se acede ao módulo, podemos encontrar o seguinte:

No local onde diz "Minimum order price", o utilizador pode preencher, então, o valor mínimo necessário para um produto/carrinho ser elegível para este método de envio.

Imediatamente abaixo, onde se lê "Maximum order weight", o utilizador pode definir o peso máximo que um produto pode ter para que seja elegível para o envio grátis.

Neste seguimento vertical, o utilizador pode escolher mostrar esta informação na página do produto, ou não.

Por último temos a última funcionalidade do módulo, que pode ser editada de acordo com a preferência do utilizador. No último ponto, "Hide these shipping methods when cart is eligible for free shipping", o utilizador pode definir que outros métodos de envio poderão ficar ocultos para o cliente, quando o carrinho deste se qualifica para o envio gratuito.

Após a configuração das condições em backoffice, a página de produto deverá apresentar a informação sobre a elegibilidade de cada produto para o envio grátis.

Desta forma, o consumidor final tem sempre feedback sobre a condição do seu carrinho, e se este cumpre as condições para um envio grátis. Como tal, procedendo para o checkout, o módulo "Skrey Free Shipping" aparecerá como opção de escolha nos métodos de envio.

The screenshot displays a checkout interface with three main sections:

- MÉTODOS DE ENVIO:** A list of shipping methods. The "SKREY SHIPPING METHOD" is highlighted with a red border and shows a value of 0.00 €.
- LEVANTAMENTO EM LOJA:** A list of in-store pickup locations, each with a value of 0.00 €:
 - LOJA PORTO
 - LOJA BENFICA
 - LOJA LERIA
 - LOJA PARQUE DAS NAÇÕES
- DETALHES DO PEDIDO:** A summary of the order items and their total price.

Introduzir código de desconto		APLICAR
DETALHES DO PEDIDO		
Portátil MSI 15.6" Prestige 15 A11SC3-200PT	Portátil MSI 15.6" Prestige 15 A11SC3-200PT	
0x1	0x1	
		1.779,00 €

Instalação

Procesos para instalação do módulo.

Possíveis problemas

A label para o free shipping poderá não aparecer, nesse caso deverá verificar se o módulo se encontra devidamente instalado.

Atualização

A atualização do módulo é feita através do mesmo processo utilizado na instalação do "Skrey Free Shipping (Label)". Dependendo da forma pela qual se procedeu à instalação do módulo, a atualização segue o mesmo caminho, apenas com a versão mais recente do mesmo

Como tal, para atualizar o módulo, basta consultar aqui o manual de instalação do mesmo e proceder aos passos indicados.

Se for necessário alguma modificação na base de dados, por exemplo, o próprio módulo faz esse trabalho pelo utilizador.

Questões frequentes

- Como posso verificar que foi instalado com sucesso?

Para tal, deve verificar se o módulo está nas configurações do backoffice, seguindo o seguinte caminho: Stores > Configurations > Sales > Shipping Methods > Skrey Free Shipping.

- O módulo não está a funcionar, o que posso fazer?

Em primeiro lugar, verifique se procedeu corretamente à instalação do mesmo. Pode consultar aqui o nosso manual de instalação do módulo "Skrey Free Shipping (Label)".