Lisbon School of Economics & Management
Universidade de Lisboa

MESTRADO

GESTÃO DE SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

APLICAÇÃO DA METODOLOGIA SCRUM NA GESTÃO DE PROJETOS NUMA MULTINACIONAL DE CONSULTORIA TECNOLÓGICA

MARIA JOSÉ DE LEMOS ALMEIDA GOMES

OUTUBRO 2022

Lisbon School of Economics & Management
Universidade de Lisboa

MESTRADO EM
GESTÃO DE SISTEMAS DE INFORMAÇÃO

TRABALHO FINAL DE MESTRADO
RELATÓRIO DE ESTÁGIO

APLICAÇÃO DA METODOLOGIA SCRUM NA GESTÃO DE PROJETOS NUMA MULTINACIONAL DE CONSULTORIA TECNOLÓGICA

MARIA JOSÉ DE LEMOS ALMEIDA GOMES

ORIENTAÇÃO:

PROFESSOR DOUTOR RUI TRIGO PEREIRA (ISEG)

CRISTINA BESSA

OUTUBRO 2022

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que contribuíram para a concretização deste relatório de estágio.

Em primeiro lugar agradecer ao Professor Doutor Rui Trigo Pereira pela sua orientação, apoio e disponibilidade manifestada durante a elaboração deste relatório.

Agradecer também à minha orientadora Cristina Bessa e à minha colega Paula Martins, por todos os ensinamentos, paciência, apoio e disponibilidade ao longo de todo o período de estágio.

Aos meus pais, Alice e José e à minha avó Rosa, agradeço por sempre me apoiarem nas minhas decisões e por me darem a possibilidade de investir na minha formação.

Ao meu namorado Victor, por todo o carinho, motivação e companheirismo ao longo de todo este percurso.

Por último, mas não menos importante, aos meus amigos, em especial à Iara Chande e à Maria Beatriz Cruz, por todos os conselhos, motivação e palavras encorajadoras.

Obrigada!

RESUMO

Durante os últimos anos, as metodologias ágeis ganharam muita popularidade, estando cada vez mais a ser adotadas pelas empresas para o desenvolvimento de projetos de *software* devido à sua flexibilidade e eficácia. Com o recurso a estas metodologias, as empresas e as organizações conseguem rapidamente responder à procura das necessidades dos respetivos clientes e a oportunidades de mercado, podendo adquirir uma vantagem competitiva.

O presente relatório tem como objetivo apresentar as diferentes metodologias para a gestão e desenvolvimento de *software*, focando-se essencialmente na metodologia Scrum.

Neste relatório, é ainda realizada uma análise da utilização da metodologia Scrum no contexto de uma multinacional de consultoria tecnológica. Verificou-se que na organização em questão, foi feita uma adaptação do Scrum, nomeadamente na adoção do papel de Gestor de Projeto ágil, na alocação de um "*buffer*" em cada *sprint* para correção de *bugs*, na realização de reuniões diárias com o cliente e por último, nas tarefas atribuídas ao *Product Owner*.

As atividades de gestão de projeto desenvolvidas ao longo do desenvolvimento aplicacional proporcionaram a concretização de um projeto principal que consistiu no desenvolvimento de uma plataforma *web*, permitindo o uso de metodologia ágeis num contexto profissional.

PALAVRAS-CHAVE: Gestão de Projetos; Metodologias Ágeis; Scrum; Desenvolvimento de *Software*

ABSTRACT

During the last years, agile methodologies have gained a lot of popularity, being increasingly adopted by companies for the development of software projects due to their flexibility and effectiveness. With the use of these methodologies, companies and organizations can quickly respond to the demand of their customers' needs and market opportunities, being able to acquire a competitive advantage.

This report aims to present the different methodologies for software management and development, focusing essentially on the Scrum methodology.

This report also analyzes the use of the Scrum methodology in the context of a multinational technology consulting company. It was verified that in the organization in question, an adaptation of Scrum was made, namely in the adoption of the role of Agile Project Manager, in the allocation of a "buffer" in each sprint to fix bugs, in the daily meetings with the client and finally, in the tasks assigned to the Product Owner.

The project management activities carried out throughout the application development provided the materialization of a main project that consisted in the development of a web platform, allowing the use of agile methodologies in a professional context.

KEYWORDS: Project Management; Agile Methodologies; Scrum; Software Development

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS, ACRÓNIMOS E SIGLAS

CMMI	Capability Maturity Model Integration
DEV	Ambiente de Desenvolvimento
DSDM	Dynamic Software Method
FDD	Feature Driven Development
PROD	Ambiente de Produção
QA	Ambiente de Qualidade
TDD	Test Driven Development
XP	Extreme Programming

1. INTRODUÇÃO

O principal objetivo do presente relatório é apresentar e descrever as principais atividades desenvolvidas ao longo do estágio numa multinacional de consultoria tecnológica de referência, aplicando os conhecimentos adquiridos ao longo do mestrado em Gestão de Sistemas de Informação.

Neste sentido, são detalhadas um conjunto de tarefas relacionadas com a aplicação da metodologia Scrum na gestão de um projeto de desenvolvimento de uma plataforma *web*, mais concretamente as tarefas relacionadas com o levamento e gestão de requisitos e a realização de testes funcionais transversais.

Desta forma, o presente relatório encontra-se estruturado num conjunto de seis capítulos. O capítulo da revisão da literatura, contextualiza as principais áreas cobertas na realização do presente estágio. A seguir, no capítulo três, é apresentada a organização e o departamento onde foi desenvolvido o presente estágio, introduzindo o contexto organizacional. O quarto capítulo inicia com a apresentação dos principais objetivos que desencadearam o estágio e a experiência profissional. De seguida, é feita uma descrição mais detalhada do estágio, apresentado as principais metodologias utilizadas e a formação desenvolvida. Por fim, é realizada uma descrição pormenorizada das atividades desenvolvidas durante o estágio. No capítulo cinco é realiza uma análise crítica e comparativa entre as atividades desenvolvidas em contexto prático de estágio e a literatura. Por último, no capítulo seis é apresentada a conclusão do presente trabalho com a reflexão sobre a experiência de estágio, evolução pessoal, profissional e os desafios encontrados durante a realização do mesmo.

2. REVISÃO DA LITERATURA

No universo mais competitivo dos dias de hoje, as empresas e as organizações são confrontadas com a constante necessidade de se adaptarem a novos requisitos devido ao aparecimento de inovações, novas exigências e novas tecnologias. No desenvolvimento de *software* existe também uma constante mudança dos requisitos durante o ciclo de desenvolvimento, assim como uma necessidade de entregas rápidas (Moniruzzaman & Hossain, 2013). Para responder a estas alterações de uma forma eficaz, e para contribuir para uma maior eficiência nos projetos em curso, é necessário que as empresas e as organizações escolham o processo e a metodologia de gestão de projetos mais adequada para as características de cada projeto (Gaborov et al., 2021).

Os métodos mais tradicionais não se têm mostrado eficazes perante os desafios rápidos do setor tecnológico, principalmente no que diz respeito à capacidade de adaptação de novos requisitos funcionais e técnicos (Alsaqqa et al., 2020). Deste modo, surgem os designados métodos ágeis que se caracterizam essencialmente por seguir um processo iterativo de desenvolvimento e de entregas frequentes ao cliente. Desta forma, com esta abordagem ágil existe o acompanhamento dos diferentes desenvolvimentos por parte do cliente, o que possibilita a sua participação na avaliação e definição dos novos requisitos a adicionar (Kumar & Bhatia, 2012).

Nas metodologias ágeis, destaca-se entre outras o Scrum, uma abordagem mais focada na flexibilidade, adaptabilidade e produtividade direcionada essencialmente na gestão de projetos, que utiliza uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco (Schwaber & Sutherland, 2020). Entre as diferentes metodologias, o Scrum é, muitas vezes considerado o método ágil mais popular, com mais de 66% das empresas a adotarem e seguirem esta metodologia (Digital.ai, 2021).

2.1. Gestão de projetos

Um projeto pode ser caracterizado como um esforço temporário, único e progressivo para criar um produto, serviço ou resultado específico. (PMI, 2021) A *performance* e os resultados financeiros de uma empresa estão diretamente relacionada com a boa entrega e o sucesso dos

seus projetos, desta forma, dada a sua importância, é essencial que estes sejam bem geridos (Salameh, 2014).

De acordo com o *Project Management Institute*, uma das referências nesta área, a gestão de projetos corresponde, de uma forma genérica, à aplicação de conhecimentos, ferramentas, habilidades e técnicas às atividades de projeto para cumprir os requisitos e necessidades do projeto (PMI, 2021).

A realização de uma gestão de projetos eficaz, eficiente e adaptada às características do projeto pode trazer múltiplos benefícios, tanto para a empresa como para toda a equipa envolvida. Ao adotarem métodos e estratégias de gestão de projeto, as empresas conseguem reduzir riscos, custos e melhorar a taxa de comprimento e sucesso dos projetos (Salameh, 2014).

No setor da tecnologia, em particular, onde existe uma constante e rápida mudança de requisitos, a gestão de projetos apresenta um papel crucial para garantir o cumprimento, sucesso e a entrega do projeto (José et al., 2011).

Existem diferentes metodologias de gestão de projeto, que têm como objetivo auxiliar o controlo e acompanhamento do projeto ao longo da fase de desenvolvimento, que serão abordadas e detalhadas nos capítulos seguintes do presente trabalho.

2.2. Processo de desenvolvimento de software

Sommerville (2011) define o processo de desenvolvimento de *software* como sendo um conjunto de atividades relacionadas que auxiliam a produção de um *software*.

Apesar de existirem diferentes processos para o desenvolvimento de *software*, todos eles incluem quatro atividades comuns fundamentais para a engenharia de *software*:

1. **Especificação do *Software*:** Processo de levantamento e definição das funcionalidades e requisitos do produto, assim como de identificação das restrições no desenvolvimento do *software*.
2. **Desenho e implementação do *Software*:** Processo de desenvolvimento do *software* de acordo com os requisitos e funcionalidades definidas.
3. **Validação do *Software*:** Processo que consiste em testar o *software* para garantir que este se encontra em conformidade com os requisitos e atende às expectativas do cliente.

4. **Evolução do *software*:** Para que o *software* atenda as mudanças nas necessidades do cliente é necessário que este evolua as suas funcionalidades.

Os processos de desenvolvimento de *software* são complexos e a sua escolha pode variar consoante as características do projeto e da organização (Javanmard M & Alian M, 2015). É comum as organizações desenvolverem o seu próprio processo de desenvolvimento de *software* ou adaptarem algum processo existente à sua realidade (Soares M, 2005).

Entre os diferentes processos conhecidos, existem as metodologias tradicionais, que são orientadas para a documentação e apresentam um processo muito estruturado (Soares M, 2005), e as metodologias ágeis, que procuram desenvolver um *software* de uma forma mais flexível e com o mínimo de documentação (José et al., 2011).

2.3. Metodologias de desenvolvimento de Software

2.3.1. Metodologias Tradicionais

Até ao início dos anos 2000, as metodologias tradicionais dominavam a indústria de desenvolvimento de *software* (Mirza & Datta, 2019). Também conhecidas como metodologias pesadas ou orientadas a documentação, estas são caracterizadas por envolverem um planeamento extensivo, fases de processo pré-definidas e documentação extensa. Uma das metodologias tradicionais mais utilizadas, até aos dias de hoje, é o modelo Cascata ou *Waterfall* (Alsaqqa et al., 2020).

Estas metodologias podem ser muito úteis quando se trata do desenvolvimento de um *software* bastante complexo, uma vez que ajudam a eliminar o desenvolvimento de *software* desorganizado e sem planeamento e a entregar um produto de alta qualidade de forma sistemática que atende todos os requisitos (Alsaqqa et al., 2020).

No entanto, para que os projetos tenham sucesso na implementação deste tipo de metodologias, é necessário que os requisitos sejam estáveis e previsíveis. Isto faz com que muitos projetos, que seguem os métodos tradicionais de desenvolvimento de *software*, enfrentem muitos problemas, na sua maioria relacionados com a manutenção e a alteração de requisitos a meio do processo. Algumas desses alterações podem resultar em mudanças drásticas no processo de desenvolvimento (Soares M, 2005; Javanmard M & Alian M, 2015).

Devido a estes problemas, os métodos tradicionais começaram a ser questionados pelas organizações, o que levou a uma mudança na gestão e desenvolvimento de processos de *software*, surgindo as metodologias ágeis (Alsaqqa et al., 2020; Soares, 2005).

2.3.2. Metodologias Ágeis

Os métodos de desenvolvimento de *software* ágil surgiram significativamente antes da década de 90, com o objetivo inicial de reduzir a documentação extensa característica das metodologias tradicionais, e de melhorar a capacidade de adotação de mudanças durante o processo de desenvolvimento, sem comprometer o processo ou criar um esforço adicional (Alsaqqa et al., 2020; Molina Ríos & Pedreira-Souto, 2019).

O termo metodologias ágeis veio a tornar-se popular apenas em 2001, quando um grupo de dezassete especialistas em processos de desenvolvimento de *software* se reuniu e formalizou o movimento *Agile*, resultando na publicação do *Agile Manifesto.* Neste documento encontram-se descritos um conjunto de valores e princípios fundamentais que orientam e caracterizam este tipo de metodologias (Javanmard M & Alian M, 2015).

Na Tabela I são apresentados os valores descritos por Beck et al., (2001) no *Agile Manifesto*:

Tabela I - Valores do *Agile Manifesto*

Mais valorizados		Menos valorizados
Indivíduos e interações	acima de	Processos e ferramentas
Software em funcionamento		Documentação extensiva
Colaboração com o cliente		Negociação de contratos
Resposta rápida à mudança		Seguir um plano

Fonte: Beck et al., (2001)

Segundo os autores do *Agile Manifesto*, os valores apresentados à direita na tabela não são rejeitados, apenas têm uma importância secundária quando comparados com os valores apresentados à esquerda na tabela.

No *Agile Manifesto*, além dos quatros valores apresentados, foram também estabelecidos doze princípios orientadores do desenvolvimento ágil. Esses princípios são os seguintes (Beck et al., 2001):

1. A prioridade do desenvolvimento ágil é satisfazer o cliente através de uma entrega rápida e contínua de *software* com valor;
2. Mesmo numa fase final do processo de desenvolvimento, devem ser aceites alterações de requisitos. Os processos ágeis utilizam a mudança para criar vantagem competitiva do cliente;
3. Realizar entregas frequentes de *software* em funcionamento. Essas entregas devem ser realizadas num período de semanas a alguns meses, com preferência a intervalos mais curtos;
4. Ao longo de todo o projeto, as pessoas responsáveis pelo negócio e os *developers* devem trabalhar em conjunto diariamente;
5. Construir projetos com indivíduos motivados, fornecendo-lhes o ambiente e o suporte de que precisam, confiando nas suas capacidades de realizar o trabalho;
6. A forma mais eficiente e eficaz de transmitir informações para e dentro de uma equipa de desenvolvimento é através de uma boa comunicação *face-to-face;*
7. A principal medida de progresso é um *software* em funcionamento;
8. Os processos ágeis promovem o desenvolvimento sustentável. S*ponsers*, *developers* e utilizadores devem estar aptos para manter um ritmo constante indefinidamente;
9. Atenção contínua ao bom *design* e à excelência técnica aumenta a agilidade;
10. A Simplicidade (arte de maximizar a quantidade de trabalho não realizado) é fundamental;
11. Os melhores requisitos, *designs* e arquiteturas surgem de equipas auto-organizadas; e
12. A equipa reflete, em intervalos regulares, sobre formas de se tornar mais eficaz. Após esta reflexão a equipa ajusta o seu comportamento de acordo com as ações definidas.

As metodologias ágeis de desenvolvimento de *software* são baseadas em processos de desenvolvimento iterativos e incrementais, onde os requisitos e as soluções são desenvolvidos por equipas multifuncionais e auto-organizadas (Moniruzzaman & Hossain, 2013).

Este modelo de desenvolvimento leve, foca-se essencialmente na satisfação do cliente, fornecendo entregas rápidas e contínuas de *software* em funcionamento (McCormick, 2012).

Segundo Alsaqqa et al. (2020, p. 250), o termo agilidade refere-se à *"capacidade de promover de forma adaptativa uma resposta rápida a qualquer mudança, seja no ambiente, nos requisitos do utilizador ou em quaisquer restrições de entrega.*"

Deste modo, um dos principais focos deste modelo de desenvolvimento de *software* é a capacidade de adaptação à mudança de requisitos ao longo do processo. O *software* é desenvolvido em ciclos curtos ou pequenas iterações, o que facilita a introdução de alterações nos requisitos do produto (McCormick, 2012).

2.4. Metodologias ágeis existentes

Ao longo do tempo, diferentes tipos de metodologias ágeis foram desenvolvidas. Apesar de estas metodologias apresentarem diferentes abordagens, todas elas seguem os valores e princípios descritos na *Agile Manifesto* (Stoica et al., 2013).

Algumas das metodologias ágeis de desenvolvimento de *software* mais utilizadas são: Scrum, *Extreme Programming*, *Dynamic Software Development Method*, *Feature Driven Development* e *Test Driven Development* (Alsaqqa et al., 2020; Javanmard M & Alian M, 2015).

Tabela II - Descrição das metodologias ágeis mais populares

Método Ágil	**Descrição**
Scrum	O Scrum é uma metodologia baseada num processo de desenvolvimento iterativo e incremental para gestão de projetos de desenvolvimento ágil. Concentra-se na forma como as equipas podem ser organizadas para produzir *software* em funcionamento num ambiente em constante mudança. Nesta abordagem, todo o ciclo de desenvolvimento é dividido num conjunto de iterações, onde cada iteração é chamada de *sprint* e tem uma duração média de 2 a 4 semanas. O processo de desenvolvimento inclui três fases: pré-planeamento, desenvolvimento e encerramento do projeto.

	A equipa Scrum é caracterizada por ser auto-organizada e por apresentar três papeis principais: *Product Owner*, *Scrum Master* e Equipa de Desenvolvimento.
Extreme Programming	O XP foca-se na melhoria da qualidade do *software* e da capacidade de resposta rápida à mudança de requisitos, sendo bastante utilizado para o desenvolvimento de *software* de alta qualidade. Esta metodologia baseia-se em cinco valores: comunicação, simplicidade, *feedback*, respeito e coragem. Apresenta como principais características interações curtas com *releases* e *feedback* rápido, forte interação com o cliente, comunicação e coordenação entre os *developers*, interação e testes contínuos, propriedade coletiva do código, documentação estritamente necessária e programação em pares (*pair programming)*.
Dynamic Software Development Method	O método dinâmico de desenvolvimento de *software* caracteriza-se por ser uma abordagem iterativa e incremental, onde a qualidade do *software* é um valor muito crítico. A ideia principal do DSDM é definir os recursos e o tempo do projeto e posteriormente ajustar a quantidade de funcionalidades que podem ser desenvolvidas com esses recursos e nesse período. O processo de desenvolvimento segundo o DSDM apresenta cinco fases: estudo de viabilidade, estudo de negócios, iteração do modelo funcional, iteração de *design* e construção e implementação. As duas primeiras fases são sequenciais e feitas apenas uma vez. As restantes fases são iterativas e incrementais.
Feature Driven Development	FDD é um processo iterativo e incremental de desenvolvimento de *software* ágil. Esta abordagem foca-se na gestão do desenvolvimento com base numa lista de funcionalidades do projeto. É considerado um método de desenvolvimento de *software* altamente adaptável, uma vez que aceita mudanças de requisitos na fase final do processo de desenvolvimento. O foco principal do FDD é entregar resultados de alta qualidade ao longo de todas as fases do processo de desenvolvimento. Esta abordagem apresenta cinco etapas sequenciais: desenvolvimento do modelo geral, elaboração da lista de funcionalidades, planeamento por

	funcionalidade, *design* por funcionalidade e desenvolvimento por funcionalidade.
Test Driven Development	O TDD baseia-se em ciclos iterativos de desenvolvimento curtos. Nesta abordagem de desenvolvimento de *software*, antes do desenvolvimento real do produto, existe um processo de desenvolvimento e execução de testes automatizados. As etapas características desta abordagem são: adicionar um teste, executar todos os testes e verificar se este novo teste falha, fazer pequenas alteações no código, efetuar mais testes e, por último realizar *code refactoring*.

Fonte: (Alsaqqa et al., 2020; Javanmard M & Alian M, 2015)

2.4.1. Scrum

O Scrum é uma *framework* ágil focada no processo de desenvolvimento de *software* que foi desenvolvida e implementada por Jeff Sutherland e Ken Schwaber. Esta metodologia apresenta como objetivo acelerar o processo de desenvolvimento, mantendo a qualidade do produto final (Sutherland, 2014).

Segundo Sutherland e Schwaber, o Scrum é uma *framework* baseada num processo iterativo e incremental, que procura otimizar a previsibilidade e controlar os riscos do processo de desenvolvimento. O seu foco principal centra-se em encontrar uma estratégia de trabalho para a equipa de forma que esta consiga desenvolver o produto de forma flexível e num ambiente em constante mudança (Javanmard M & Alian M, 2015).

Esta *framework* foi construída com base em três pilares fundamentais, que se encontram descritos na seguinte tabela:

Tabela III - Pilares fundamentais do Scrum

Pilares	Descrição
Transparência	Todos os aspetos relevantes ao processo de desenvolvimento devem ser conhecidos e estar transparentes para toda a equipa.

Inspeção	Todos os aspetos do processo de desenvolvimento devem ser revistos com frequência, com a finalidade de detetar qualquer problema ou inconformidade que possa pôr em causa o produto final.
Adaptação	Caso algum aspeto do processo de desenvolvimento se desvie do aceitável, os processos devem ser revistos e ajustados.

Fonte: Schwaber & Sutherland (2020)

Ciclo de desenvolvimento Scrum

O processo de desenvolvimento, segundo esta *framework*, divide-se em três grandes fases, que se encontram representadas na Figura 1: a fase de pré-planeamento, a fase de desenvolvimento e a fase de encerramento do projeto (Alsaqqa et al., 2020; Coram & Bohner, 2005) .

Na fase de planeamento, também conhecida por fase de *pré-sprint*, é feito o levantamento e definição dos requisitos do utilizador, não sendo esperado que todos os requisitos sejam reportados pelo cliente nesta fase do processo. Posteriormente, todos estes requisitos são escritos em formato de *user-stories*[1] e são organizados numa lista, designada por *Product Backlog*, de acordo a prioridade do cliente e o esforço de desenvolvimento. O *Product Backlog* geralmente apresenta uma atualização continua, uma vez que, durante o processo de desenvolvimento, podem ser adicionados novos requisitos, ou alteradas as prioridades dos mesmos. Nesta fase também é definida a equipa de projeto, assim como as ferramentas e recursos a serem utilizados (Alsaqqa et al., 2020).

A fase que se segue é a fase de desenvolvimento, onde o processo é dividido em várias iterações bem definidas, com uma duração média de 2 a 4 semanas, designadas por *Sprints* (Coram & Bohner, 2005).

No início de cada *Sprint*, faz-se a *Sprint Planning Meeting*, que consiste numa reunião de planeamento onde a equipa seleciona e estima quais os requisitos do *Product Backlog* vão ser

[1] Uma *user-story* é uma declaração curta e não técnica de um requisito de produto escrita do ponto de vista do utilizador. É geralmente escrita de acordo com a seguinte estrutura: "Como um <tipo de utilizador>, eu quero <realizar alguma tarefa> para que eu possa <atingir algum objetivo. >" (Agile Academy, n.d.).

implementados no *sprint* (Sharma et al., 2012). Esta lista de requisitos é denominada de *Sprint Backlog* (Abrahamsson et al., 2017).

Ao longo do *Sprint*, são realizadas reuniões diárias, as *Daily Scrum Meetings*, que tem como objetivo sincronizar as atividades entre os elementos da equipa, melhorar a comunicação e identificar possíveis impedimentos (Alsaqqa et al., 2020). Estas reuniões têm uma duração de 15 minutos e são geralmente realizadas sempre no mesmo horário e local (Andrei et al., 2019).

No final de cada *Sprint* é realizada uma reunião de revisão, a *Sprint Review*, onde é validado se todos os requisitos para o *sprint* foram implementados. Durante esta reunião também são apresentados ao cliente os desenvolvimentos entregues naquela iteração e discutidos quais os requisitos a implementar no próximo *sprint*. Por fim, é realizada a *Sprint Retrospective*, onde a equipa planeia e identifica as melhorias que devem ser implementadas nos próximos *sprints* de forma a aumentar a qualidade e a eficácia da equipa (Schwaber & Sutherland, 2020).

A fase de encerramento do projeto é quando existe um acordo entre a equipa e o cliente de que todos os requisitos foram concluídos. Nesta fase é entregue toda a documentação do produto, assim como o manual de utilizador (Alsaqqa et al., 2020).

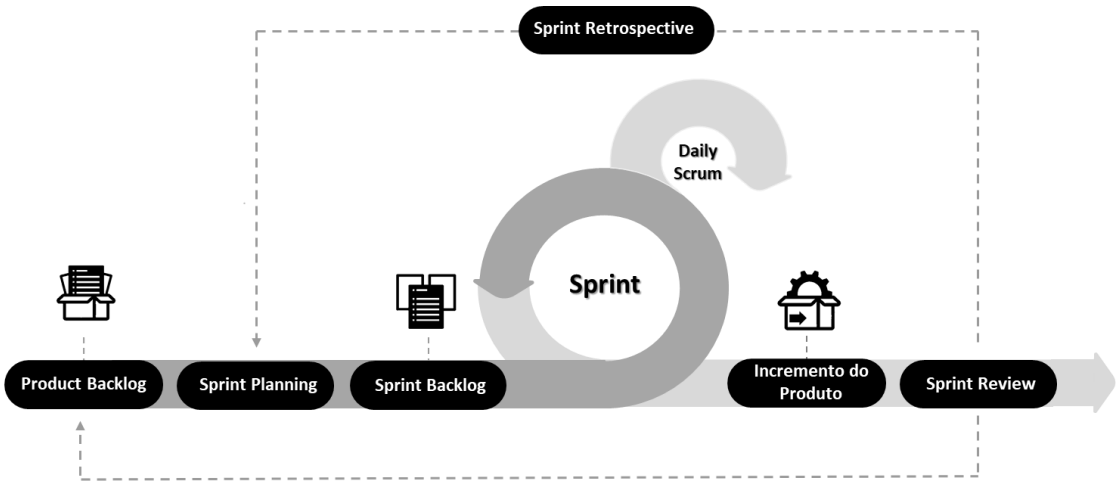


Figura 1 - Ciclo Scrum

Fonte: Scrum.org (adaptado)

Equipa Scrum

Para uma implementação de sucesso desta metodologia é crucial que a equipa tenha por base cinco valores fundamentais: Compromisso, Coragem, Foco, Abertura e Respeito (Andrei et al., 2019). A equipa de desenvolvimento deve comprometer-se a alcançar os seus objetivos, ter coragem para encarar e solucionar problemas complexos e focar-se nos objetivos da equipa e de cada *sprint*. Os elementos da equipa também devem estar sempre abertos a novos desafios e mudanças. Quando a equipa consegue incorporar todos estes valores, consegue cumprir os três pilares do Scrum (transparência, inspeção e adaptação), tornando-se mais confiante (Schwaber & Sutherland, 2020).

A Equipa Scrum apresenta três papeis principais que têm tarefas e responsabilidades distintas durante o processo: *Product Owner, Scrum Master* e a Equipa de Desenvolvimento.

O primeiro tem a visão do produto e é o representante do cliente. É responsável por criar e priorizar os requisitos de acordo com o valor que estes representam para o cliente. Além de ajustar as funcionalidades e as suas prioridades no *Product Backlog*, o *Product Owner* é responsável por definir as metas da equipa e por aceitar ou rejeitar as entregas de produto (Alsaqqa et al., 2020; Andrei et al., 2019).

O *Scrum Master* é a pessoa responsável por garantir que todas as práticas, valores e regras do Scrum estão a ser compridas pela equipa durante o projeto. É também ele o responsável por conduzir todas as reuniões Scrum e proteger o *sprint* de qualquer interferência externa. Outro papel do *Scrum Master* é o de garantir que a equipa está funcional e otimizada, eliminando qualquer impedimento que possa afetar o trabalho de desenvolvimento da equipa e garantir que existem todos os recursos necessário para a realização das atividades do *sprint* (Alsaqqa et al., 2020; Andrei et al., 2019) .

A Equipa de Desenvolvimento Scrum é caracterizada por ser uma equipa pequena, geralmente composta por 10 ou menos elementos auto-organizados, responsáveis por definirem e organizarem as tarefas do *sprint* (Schwaber & Sutherland, 2020). Esta equipa pode ser composta por *business analysts*, *system analysts*, *developers*, arquitetos de *software, UI/UX designers* e *testers*. Trata-se também de uma equipa multifuncional, onde cada elemento apresenta as *skills* necessárias para criar valor em cada *sprint* (Andrei et al., 2019; Schwaber & Sutherland, 2020).

3. APRESENTAÇÃO DA EMPRESA

A instituição de acolhimento caracteriza-se como uma empresa multinacional tecnológica de origem portuguesa. A empresa encontra-se sediada em Portugal, com escritórios em diferentes cidades: Lisboa, Porto, Coimbra, Bragança, Açores, Madrid, Barcelona, Valência, San Sebastián , Dénia e Huelva. Esta empresa conta com mais de 1100 colaboradores, tendo projetos em vários países como Espanha, França e os países do Benelux.

A organização tem presença em três grandes mercados, o mercado de Saúde, de Farmácia Comunitária e o da Consultoria Tecnológica. Neste sentido, é uma empresa de referência na área da Saúde e na Farmácia, tanto no mercado nacional, como igualmente no mercado espanhol.

Na unidade da Saúde foca-se essencialmente no desenvolvimento, implementação e suporte de um conjunto de soluções especializadas para clínicas, hospitais e farmácias hospitalares.

No mercado de Farmácia Comunitária fornece uma oferta completa de serviços que incluem todo o tipo de soluções para as atividades de uma farmácia, nomeadamente consultoria, conceção e projetos de espaços de lojas, automação, infraestruturas e consumíveis.

A unidade de consultoria tecnológica dedica-se à prestação de serviços de transformação digital, tanto em entidades públicas como privadas, em todos os setores de atividade.

Presta serviços de implementação e suporte a soluções e plataformas de infraestruturas de IT e consultoria tecnológica e aplicacional de IT em Microsoft, OutSystems e Java. Esta unidade oferece também serviços de implementação de tecnologias inovadoras em projetos de *Digital Process Automation*, *Cybersecurity*, *Advanced Analytics* e *Cloud Management*.

A unidade em questão obteve a acreditação "CMMI - versão 2.0, com extensão das práticas à metodologia Agile". A atribuição desta certificação indica que a empresa possuí um nível de maturidade onde os processos são bem definidos e compreendidos pelos colaboradores. Um dos critérios que ajudou a atribuição desta certificação foi a utilização de metodologias ágeis no desenvolvimento de *software*.

No que diz respeito à sua Missão, pretende liderar a transformação digital, com foco nas pessoas e no bem-estar da sociedade global. Tem como Visão transformar as organizações e

melhorar a vida das pessoas através da inovação tecnológica. Apresenta ainda como valores a superação, inovação, ética e sustentabilidade.

4. CONTEXTO DO ESTÁGIO

4.1. Objetivos do Estágio

Tal como referido no capítulo anterior, o estágio profissional decorreu numa empresa multinacional tecnológica, mais concretamente na unidade de consultoria tecnológica. O estágio ocorreu entre os meses de março e agosto de 2022, com uma duração aproximada de 6 meses.

Após o contacto com a empresa, foram definidas as tarefas de estágio a realizar e foi elaborado um protocolo de estágio entre a universidade, a estagiária e a instituição de acolhimento. Neste protocolo foram definidos os objetivos de estágio, assim como, as normas a cumprir pela instituição de acolhimento, a universidade e a estagiária.

Os objetivos pré-definidos por parte da empresa para a estagiária foram os seguintes:

- Integração na equipa de trabalho e conhecimentos da atividade da empresa;
- Integração nas tarefas diárias, consolidação do funcionamento da empresa e dos processos de comunicação efetuados;
- Participação nas reuniões com o negócio para conhecimento do mesmo, e apoio na especificação de requisitos;
- Ser um participante ativo na especificação de requisitos funcionais e não funcionais;
- Apoiar no desenvolvimento e gestão da visão de projetos, do seu âmbito, dos planos para os mesmos e respetivos objetivos;
- Participar nas reuniões regulares de equipa;
- Acompanhar a equipa técnica no teste da aplicabilidade dos requisitos;
- Desenvolver as competências adquiridas no decorrer da formação académica e adquirir novos conhecimentos essenciais para o bom desempenho das tarefas designadas e futuro profissional.

4.2. Descrição das atividades realizadas

A fase inicial do estágio, consistiu num período de integração e familiarização da estagiária com a instituição de acolhimento, tendo sido apresentado a atividade da empresa, assim como as metodologias e ferramentas de trabalho utilizadas.

Durante a segunda semana, a estagiária participou, juntamente com outros estagiários de outras áreas, num *case study*. O objetivo desse *case study* era o de desenvolver uma plataforma de *e-commerce*, com o acompanhamento de diferentes colaboradores da empresa. Nesta atividade, as funções atribuídas à estagiária foram a elaboração de um documento com todos os requisitos para o site de *e-commerce*. Aqui a estagiária trabalhou em conjunto com os restantes elementos e no final da semana foi apresentado o trabalho desenvolvido aos colaboradores envolvidos.

Num momento seguinte, relativo à terceira e quarta semanas de estágio, a estagiária teve a oportunidade de ter várias sessões de formação teórica e práticas com o objetivo de obter as competências base necessárias para poder desempenhar funções de Analista Funcional, ou seja, para ser capaz de auxiliar no levantamento de requisitos, elaboração de *user-stories* e realização testes.

Nas restantes semanas de estágio, a estagiária acompanhou e auxiliou a orientadora em todas as suas tarefas em diferentes projetos, conseguindo adquirir competências através da aplicação prática. Desta forma, foi possível participar e contribuir em vários projetos, tendo o estágio se focado essencialmente num projeto principal, onde a estagiária esteve inserida do início ao fim. Uma vez que as tarefas realizadas nos diferentes projetos eram transversais e se focaram em atividades semelhantes, para efeitos deste relatório irão ser descritas as tarefas realizadas no projeto principal.

Tabela IV - Cronograma das atividades realizadas no estágio

Fonte: Autor

4.3. Apresentação do Projeto Principal

O projeto consistiu no desenvolvimento de uma plataforma que permite o cálculo da elegibilidade do cidadão à atribuição de um apoio social que garante acesso a serviços a um preço mais baixo.

A plataforma deveria permitir a receção, via formulário *web*, da informação do cidadão para o qual se pretendia calcular a elegibilidade, devia também guardar a informação do cidadão de forma segura em base de dados, assim como invocar, via iAP, os serviços da Segurança Social e Autoridade Tributária para obtenção de informação de elegibilidade.

Foi também desenvolvido um *backoffice* para a gestão de utilizadores e perfis, bem como para obtenção de informação estatística.

Nesse projeto foi adotada a metodologia de desenvolvimento ágil Scrum, adaptada à especificidade do projeto em causa. Este foi dividido em nove *sprints*, tendo cada *sprint* a duração de duas semanas.

A equipa era constituída por um Gestor de Projeto ágil, um *Product Owner*, um Arquiteto de *Software* e um Programador. A estagiária integrou a equipa com o objetivo de auxiliar a *Product Owner* nas suas tarefas e atividades.

4.3.1. Ferramentas utilizadas

Ao longo do estágio, foi necessário o uso de ferramentas de apoio à gestão e execução das diferentes tarefas realizadas pela estagiária no projeto em questão.

As ferramentas utilizadas foram as seguintes:

- **Confluence**

A plataforma Confluence foi utilizada para organizar e agregar toda a documentação do projeto, desde os requisitos do cliente, documentação enviada ao cliente para aprovação e anotações de reuniões.

- **JIRA**

Toda a gestão dos *sprints* foi realizada na plataforma JIRA. Nesta plataforma foi definido a duração do *sprint* (duas semanas), foram criados os épicos do projeto, escritas e geridas as *user-*

stories e criadas todas as tarefas a desenvolver durante o projeto. Através desta plataforma era possível atribuir as tarefas ao membro da equipa responsável, deste modo a equipa sabia quais as tarefas a cumprir durante o *sprint*. Aqui também foram criados e executados os casos de teste, assim como reportados todos os *bugs* encontrados durante os testes.

4.4. Descrição das atividades realizadas no projeto

Ao longo do projeto, a estagiária desempenhou um conjunto de tarefas que se repetiram ao longo dos nove *sprints* e que se agrupam em quatro grupos distintos: Levantamento de requisitos funcionais e não funcionais, Gestão do *Product Backlog*, Reuniões regulares de equipa e Realização de Testes.

À medida que a estagiária se integrou na equipa e adaptou às metodologias de trabalho foi-lhe concedida uma maior autonomia na realização das tarefas.

Na descrição das atividades realizadas serão ocultadas algumas informações por motivos de confidencialidade.

4.4.1. Levantamento de requisitos funcionais e não funcionais

Na instituição de acolhimento o início do processo de desenvolvimento de *software*, utilizando a metodologia Scrum, caracteriza-se pela realização de um *sprint* 0. Neste *sprint* 0 é realizado todo o planeamento inicial de forma a antecipar qualquer trabalho necessário para permitir que a equipa inicie o *sprint* 1 sem qualquer impedimento e de forma mais eficaz.

Nesta fase do processo a *Product Owner* trabalha em conjunto com o cliente para definir a visão do projeto, identificando as necessidades do cliente para o produto. É também nesta fase que existe a definição/*setup* da arquitetura e da infraestrutura, assim como se realiza o levantamento de todas as dependências técnicas.

Neste sentido, no projeto em questão, a estagiário acompanhou e auxiliou a *Product Owner* nas reuniões com o cliente para o levantamento dos requisitos funcionais (requisitos diretamente relacionados às funcionalidades da plataforma) e não funcionais (requisitos relacionados com o uso da plataforma em termos de desempenho, usabilidade, segurança, manutenção e tecnologias envolvidas). Durante estas reuniões iniciais, foram registados os épicos da plataforma definidos pelo cliente.

Uma vez realizado o levantamento e registo dos primeiros épicos, foi criado o *Product Backlog*, ou seja, foram listados todos os itens que foram identificados nas reuniões iniciais com o cliente. No entanto, os épicos representavam itens sem detalhe suficiente ou demasiado grandes para serem concluídos num único *sprint*. Deste modo, esses épicos foram depois fragmentados em *user-stories* menores de modo a poderem ser refinadas, priorizadas, estimadas e desenvolvidas num *sprint*. Cada *user-story* apresentava os critérios de aceitação associados, que são as regras definidas antes do desenvolvimento da funcionalidade de acordo com os requisitos estabelecidos pelo cliente e descrevem as condições que devem satisfazer a *user-story*.

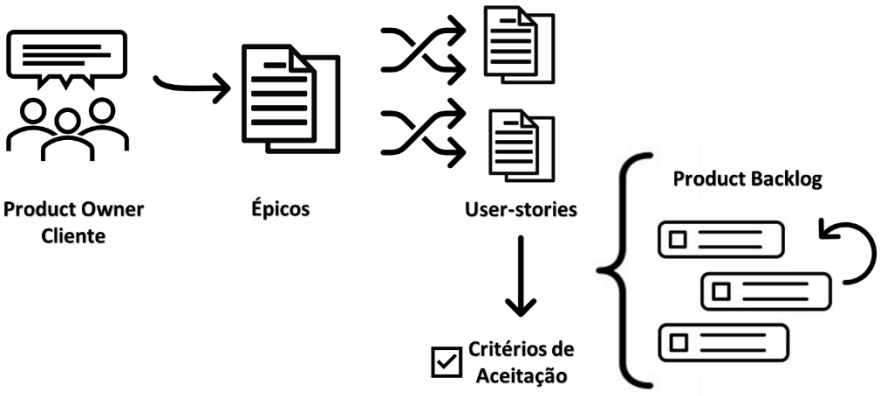


Figura 2 – Processo de levantamento de requisitos

Fonte: Autor

Para facilitar a organização e a rastreabilidade dos requisitos, cada *user-story*, para além de ter um código identificador único criado automaticamente pela plataforma, seguia ainda a seguinte estrutura: "#US001 - Como um <tipo de utilizador>, eu quero <realizar alguma tarefa> para que eu possa <atingir algum objetivo.>".

Após terem sido definidos todos os critérios de aceitação para uma determinada *user-story*, esta era enviada para o cliente para aprovação. Por vezes foi necessário realizar reuniões com o cliente para esclarecimento e refinamento das *user-stories*.

Após a aprovação do cliente eram realizadas sessões de *backlog refinement* com a equipa para apresentação e explicação das *user-stories* e para estas depois serem estimadas e priorizadas. Nestas reuniões iniciais foram definidas quais as *user-stories* que iriam ser desenvolvidas nos primeiros dois *sprints*.

As tarefas de criação de *user-stories* repetiam-se ao longo dos diferentes *sprints*, uma vez que iam existindo alterações aos requisitos e à prioridade dos mesmos.

***User-story* exemplo:**

Na criação das *user-stories* eram preenchidos os seguintes campos na ferramenta JIRA:

Tabela V - Atributos de uma *User-story*

Campo	Descrição
Issue Type	Permite selecionar o tipo de *issue* a ser criado, neste caso seria selecionado "*Story*".
Summary	Descrição curta da *user-story*. Seguia o padrão referido anteriormente.
Description	Corresponde a uma descrição mais detalhada. Neste campo eram detalhados todos os critérios de aceitação relacionadas com a *user-story* em questão.
Epic Link	Permite associar a *user-story* ao respetivo épico.
Priority	Permite a seleção do nível de prioridade. Era possível selecionar quatro níveis: *Lowest*, *Lower*, *Medium* e *High* .
Linked Issue	Permite associar a *user-story* a um outro *issue* previamente criado no JIRA.
Assignee	Elemento da equipa a quem a *user-story* está atribuída no momento.
Reporter	Campo preenchido automaticamente pela plataforma. Apresenta o elemento da equipa que criou a *user-story*.
Fix Version	Permite identificar qual a *release* onde os desenvolvimentos relacionados com a *user-story* iriam ser implementados.
Sprint	Campo preenchido automaticamente pela ferramenta quando a *user-story* era alocada a um *sprint*.
Attachments	Permite anexar documentos relacionados com a *user-story*.

Fonte: Autor

No anexo I, é apresentado um exemplo de uma *user-story* elaborada no projeto em questão. A *user-story* #US005 – Como utilizador pretendo criar e ocultar utilizadores, por forma a gerir os acessos ao BO, pretende descrever todos os critérios de aceitação relacionada com a criação de novos utilizadores no *backoffice* da plataforma *web*.

4.4.2. Gestão do Product Backlog

Uma vez que o projeto seguiu a metodologia Scrum, e tal como referido no tópico anterior, existiram alterações e acréscimos de novas funcionalidades e requisitos ao longo dos *sprints*. Desta forma, teve de ser feita uma constante gestão do *Product Backlog*.

Com o *feedback* da reunião de revisão do *sprint* ou da reunião diária com o cliente , a *Product Owner* e a estagiária analisavam o *Product Backlog* para perceber se as alterações solicitadas poderiam ser feitas. Estas alterações podiam incluir novas *user-stories*, ajustar prioridades, integrar novos requisitos, gerir mudanças e mitigar riscos identificados. Antes de ser feita qualquer alteração ao *Product Backlog*, a *Product Owner* analisava se as alterações estavam em conformidade com o âmbito estabelecido para o projeto.

Eram realizadas sessões entre a estagiária e a *Product Owner*, onde era analisado o *Product Backlog* e definidas quais as alterações que tinham de ser feitas às *user-stories*. Depois dessa análise eram atribuídas as *user-stories* a serem alteradas ou criadas à estagiária. Após revistas e alteradas, as *user-stories* eram validades e aprovadas pela *Product Owner* para posteriormente serem enviada para aprovação do Cliente.

Após a aprovação do Cliente, eram novamente revista a priorização do *Product Backlog*, uma vez que tinham sido alteradas ou adicionadas novas funcionalidades a desenvolver.

Estas sessões eram realizadas próximas do final de cada iteração, garantindo assim, que o *Product Backlog* estava pronto para o próximo *sprint*, e que as tarefas estavam planeadas com antecedência.

4.4.3. Participação em reuniões regulares da equipa

Uma das características dos projetos que seguem a metodologia Scrum, é a existência de diferentes cerimónias ao longo dos *sprints*. Estas cerimónias incluem a reunião diária da equipa (*daily scrum meeting*), *sprint planning*, *sprint review* e *sprint retrospective*. Todas estas cerimónias se repetiam em todos os *sprints*.

A estagiária participou nas várias reuniões de equipa, contribuindo para o cumprimento dos objetivos definidos para as mesmas.

Reuniões diárias da equipa

No projeto em questão a equipa participava em reuniões diárias, as *daiy scrum meetings*. Estas reuniões tinham uma duração máxima de 15 minutos e apresentavam como objetivo disseminar o conhecimento sobre o que foi realizado, por cada membro da equipa, no dia anterior e identificar o trabalho a ser feito nas próximas 24 horas. Durante a reunião os membros da equipa respondiam às seguintes questões:

- O que eu fiz ontem?
- O que eu farei hoje?
- Existe algum obstáculo que impeça alcançar a meta do *sprint*?

Para além da *daily scrum meeting* com a equipa, ainda existia uma reunião diária da equipa com o cliente. Esta reunião tinha a mesma duração máxima de 15 minutos e tinha como propósito atualizar o cliente dos desenvolvimentos em cursos e também esclarecer qualquer dúvida ou dificuldade encontrada.

Sprint Planning

No início de cada *sprint* era realizada uma sessão de planeamento com toda a equipa, a *sprint planning*. Esta reunião tinha como objetivo definir o que iria ser entregue no *sprint* e como esse trabalho iria ser realizado.

Após a criação e priorização das *user-stories* pela estagiária e a *Product Owner*, estas eram planeadas e estimadas na reunião de planeamento do *sprint*.

Numa primeira sessão, a estagiária apresentava o *backlog* priorizado com as *user-stories* que deviam fazer parte do *sprint*. De seguida, a Equipa de Desenvolvimento, neste caso o Arquiteto de *Software* e o Programador, estimavam a complexidade necessária para desenvolver a funcionalidade descrita em cada uma das *user-stories* apresentadas e determinavam quais destas podiam ser executadas no *sprint*.

Para realizar esta avaliação, para cada *user-story*, cada membro da equipa de desenvolvimento, dizia quantas horas acreditava serem necessárias para desenvolver a funcionalidade em questão. Após acordado um valor, as horas definidas passavam a ser a estimativa para a *user-story*.

Para a equipa conseguir perceber quantas funcionalidades conseguia desenvolver no *sprint*, eram definidas o número de horas produtivas para esse *sprint*.

Para a contabilização dessas horas produtivas a equipa realizava os seguintes passos:

1. Analisava quantas horas teria cada *sprint* (dado que a equipa tinha um Arquiteto de *Software* e um Programador, seriam 16 horas por cada dia do *sprint*);
2. De seguida, subtraia a esse total de horas, as horas utilizadas para a realização das cerimónias de *sprint*;
3. Definia ainda um período para a correção de possíveis *bugs* que pudessem surgir no *sprint*, também designado de "*buffer*".

Após definidas as horas produtivas do *sprint*, e terminada a estimativa para cada *user-story*, a equipa conseguia selecionar as tarefas que podiam entrar no *sprint*.

Numa outra sessão, o Arquiteto de *Software* e o Programador, definiam como transformar as *user-stories* selecionadas para o *sprint* em tarefas. Na plataforma JIRA, dentro das *user-stories*, eram associadas as diferentes *sub-tasks* criadas pela equipa de desenvolvimento. Desta forma, quando todas as *sub-tasks* eram fechadas, significava que a funcionalidade estava totalmente desenvolvida e poderia ser testada.

Sprint Review

Ao final de cada *sprint*, era realizada uma Reunião de Revisão do *sprint*. O propósito da reunião não era o de obter a aprovação formal do cliente sobre o que tinha sido feito no *sprint*, nem era o de realizar uma sessão de testes de aceitação. O objetivo da reunião de revisão era demonstrar, em tempo real, o que tinha sido desenvolvido pela equipa às partes interessadas e recolher o *feedback* sobre o incremento das funcionalidades geradas no *sprint*, para fazer as adaptações necessárias, e assim, diminuir os riscos do projeto.

Apesar da reunião ser facilitada pelo Gestor de Projeto ágil, os restantes membros da equipa também participavam ativamente.

A reunião começava pela apresentação, por parte do Gestor de Projeto, da listagem das funcionalidades desenvolvidas no *sprint* em questão e pela apresentação do gráfico de *Burndown*. O gráfico de *Burndown* é um gráfico que monitoriza o progresso da equipa de

desenvolvimento, assim, como a sua velocidade. Através deste gráfico era possível perceber se a equipa tinha conseguido desenvolver todas as funcionalidades alocados para o *sprint.*

Uma vez realizada essa apresentação, a equipa procedia com uma demonstração das funcionalidades desenvolvidas no *sprint*. Os incrementos eram apresentados ao cliente para receber o seu *feedback* e perceber se o que tinha sido desenvolvido, até ao momento, correspondia ao que era pretendido. Nesta fase, por vezes, o cliente sugeria alterações que depois eram analisadas pela equipa.

Tendo em conta o *feedback* do cliente, a estagiária e a *Product Owner* atualizavam e priorizavam novamente o *Produc Backlog* para os próximos *sprints*.

Sprint Retrospective

A reunião de retrospetiva do *sprint*, geralmente realizada após a *sprint review* e antes da reunião de planeamento do próximo *sprint*, era uma sessão onde a equipa avalia o *sprint* que foi concluído.

O foco desta reunião não é o produto em si, mas sim o processo. Nesta reunião toda a equipa avaliava o que tinha corrido bem ao longo do *sprint* e o que deveria ser melhorado.

Na reunião de retrospetiva os membros da equipa respondiam às seguintes questões:

- O que correu bem e o que correu mal?
- O que devíamos começar a fazer?
- O que devíamos parar de fazer?
- O que devíamos continuar a fazer?

Deste modo, toda a equipa analisava e discutia os pontos levantados em cada uma das questões para verificar a necessidade de adaptações e, se necessário, elaborar um plano de ação para ser implementado nos próximos *sprints*.

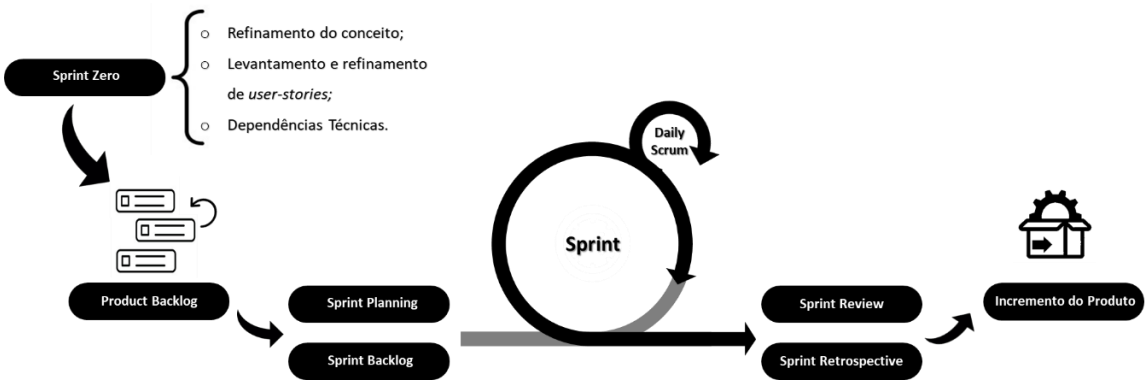


Figura 3 – Ciclo Scrum adaptado na instituição de acolhimento

Fonte: Autor

4.4.4. Realização de testes

Após o desenvolvimento de cada funcionalidade, é fundamental a realização de testes, de forma a garantir a conformidade da funcionalidade implementada com os requisitos do cliente. As atividades de testes eram desenvolvidas em todos os *sprints*.

No projeto em questão, era da responsabilidade da estagiária, a realização de testes funcionais. Os testes funcionais caracterizam-se por ser um tipo de testes que se baseiam nos requisitos funcionais. Com estes testes é possível verificar as funcionalidades da plataforma sem analisar a sua estrutura interna, uma vez que apenas é considerado o comportamento externo da plataforma. Estes testes eram realizados de forma manual, ou seja, para cada funcionalidade era feita uma simulação da utilização da plataforma como utilizador final, através do teste de vários cenários.

Uma vez que o projeto seguia uma metodologia ágil, os testes eram realizados ao longo do processo de desenvolvimento, ou seja, em cada iteração eram realizados testes a cada funcionalidade desenvolvida naquele *sprint*.

O primeiro passo na fase de testes da plataforma passava por realizar uma análise de todas as funcionalidades que iriam ser testadas no próximo *sprint* e perceber qual o esforço necessário para a realização desses tarefas.

De seguida, era criado um caso de teste, na plataforma JIRA, para cada *user-story* a ser testada. Após a criação do caso de teste, este era executado conforme as etapas planeadas e eram registados os resultados obtidos durante o processo.

No projeto em questão eram utilizados três ambientes distintos para as diferentes fases do ciclo de vida do projeto. Esses três ambientes eram: ambiente de desenvolvimento (DEV), ambiente de qualidade (QA) e ambiente de produção (PRD). Os testes funcionais eram normalmente realizados em ambiente de desenvolvimento.

Criação de casos de teste

Em cada caso de teste eram detalhados todos os passos necessários para testar a funcionalidade em questão. Os passos consistiam na descrição, de uma forma ordenada, das ações a serem realizadas durante a execução do caso de teste. Nos casos de teste eram também contempladas as pré-condições necessárias para a possível execução do caso de teste.

Para além da descrição das pré-condições e das ações a realizar, era também descrito, no caso de teste, o conjunto de resultados esperados.

Exemplo de Caso de Teste:

Para a criação dos casos de teste era utilizado o Xray Test, que é uma ferramenta de gestão de testes para o JIRA.

Na criação dos casos de teste eram preenchidos os seguintes campos na ferramenta:

Tabela VI - Atributos de um Caso de Teste

Campo	Descrição
Issue Type	Permite selecionar o tipo de *issue* a ser criado, neste caso seria selecionado "Xray Test".
Test Type	Identifica o tipo de teste a criar no JIRA. Deve corresponder a um dos tipos de teste associados ao projeto em que o caso de teste será criado. Era selecionado o valor "Manual".
Summary	Descrição curta do caso de teste.
Description	Corresponde a uma descrição mais detalhada do caso de teste.
Epic Link	Permite associar o caso de teste ao respetivo épico.

Priority	Permite a seleção do nível de prioridade. Era possível selecionar quatro níveis: *Lowest*, *Lower*, *Medium* e *High* .
Linked Issue	Permite associar o caso de teste a um outro *issue* previamente criado no JIRA. Era associado a *user-story* que estava a ser testada naquele caso de teste.
Assignee	Elemento da equipa a quem a *user-story* está atribuída no momento.
Reporter	Campo preenchido automaticamente pela plataforma. Apresenta o elemento da equipa que criou a *user-story*.
Attachments	Permite anexar documentos relacionados com a caso de teste.
Action, *Data* e *Expected Result*	Servem para descrever cada um dos passos do caso de teste.

Fonte: Autor

No anexo II, é apresentado o exemplo do caso de teste relacionado com a *user-story* referida anteriormente. O caso de teste #TC005 – Como utilizador pretendo criar e ocultar utilizadores, por forma a gerir os acessos ao BO, pretende descrever todos os passos a realizar na execução do caso de teste para testar a funcionalidade implementada.

Execução dos casos de teste

A próxima etapa no processo de testes envolvia a execução do caso de teste. Para a realização da execução eram introduzidos os dados definidos como *input* e eram seguidos todos os passos descritos no caso de teste, de forma a analisar o resultado gerado. Os resultados eram depois comparados com os resultados esperados do caso de teste. Dependendo do resultado gerado pela plataforma, podiam surgir defeitos ou *bugs* que deviam ser reportados e corrigidos.

Após a identificação do defeito, este era reportado na plataforma JIRA e associado ao passo do caso de teste onde tinha sido detetado, ficando o estado da execução do caso de teste como "*Failed*".

Assim que o *bug* estivesse corrigido era realizado de novo a execução do caso de teste, ou seja, eram realizados testes de confirmação. Se, ao realizar novamente a execução do caso de teste, este passasse com sucesso, o estado da execução transitava para "*Passed*".

Exemplo de execução de um caso de teste:

Na plataforma JIRA, para a realização da execução dos casos de teste, era criado um "*Test Execution*", onde eram adicionados todos os casos de teste que iriam ser testado naquele *sprint*.

Para a criação desse "*Test Execution*" eram preenchidos os seguintes campos:

Tabela VII - Atributos de uma Execução de Casos de Teste

Campo	Descrição
Issue Type	Permite selecionar o tipo de *issue* a ser criado, neste caso seria selecionado "*Test Execution".*
Summary	Descrição curta da execução. Geralmente era descrito qual o s*print* a que se referiam as execuções de caso de teste.
Epic Link	Permite associar a execução ao respetivo épico.
Linked Issue	Permite associar a execução a um outro *issue* previamente criado no JIRA.
Assignee	Elemento da equipa a quem a execução dos casos de teste está atribuída no momento.
Reporter	Campo preenchido automaticamente pela plataforma. Apresenta o elemento da equipa que criou a execução.
Sprint	Permite associar a execução ao respetivo *sprint*.
Attachments	Permite anexar documentos relacionados com as execuções.

Fonte: Autor

No anexo III é apresentado um exemplo da execução do caso de teste #TC005 – Como utilizador pretendo criar e ocultar utilizadores, por forma a gerir os acessos ao BO.

Reporte de erros

Caso fossem detetados defeitos ou *bugs* ao longo da execução de um caso de teste estes eram reportados à equipa de desenvolvimento através da criação de um *bug* na plataforma JIRA.

Exemplo de *bug*:

Na criação de bugs na ferramenta JIRA eram preenchidos os seguintes campos:

Tabela VIII - Atributos de um *Bug*

Campo	Descrição
Issue Type	Permite selecionar o tipo de *issue* a ser criado, neste caso seria selecionado "*Bug".*
Summary	Descrição curta do *bug*.
Description	Corresponde a uma descrição mais detalhada do problema. Deve apresentar os passos para reproduzir o problema, o que foi experienciado e o resultado esperado.
Priority	Permite a seleção do nível de prioridade. Era possível selecionar quatro níveis: *Lowest*, *Lower*, *Medium* e *High* .
Linked Issue	Permite associar o caso de teste a um outro *issue* previamente criado no JIRA.
Assignee	Elemento da equipa a quem o *bug* está atribuído no momento.
Reporter	Campo preenchido automaticamente pela plataforma. Apresenta o elemento da equipa que criou o *bug*.
Attachments	Permite anexar documentos relacionados com o problema. Podem ser relatórios ou capturas de ecrã.
Enviroment	Corresponde ao ambiente onde foi detetado o *bug* (ambiente de desenvolvimento, ambiente de qualidade ou ambiente de produção)

Fonte: Autor

No anexo IV é apresentado um exemplo de *bug* criado no projeto em questão. O *bug* foi detetado na execução do caso de teste #TC005 – Como utilizador pretendo criar e ocultar utilizadores, por forma a gerir os acessos ao BO.

Testes de aceitação

Em cada iteração, após as funcionalidades terem sido testadas com sucesso em ambiente de desenvolvimento, era feita uma passagem dos desenvolvimentos para o ambiente de qualidade, de forma a possibilitar o cliente a realizar os testes de aceitação. Estes testes serviam para o cliente validar se a plataforma se comportava de acordo com os requisitos definidos e verificar se este estava satisfeito relativamente ao produto desenvolvido.

A passagem para o ambiente de qualidade era realizada a seguir à *sprint review*, onde tinha sido feita uma demonstração e apresentação das funcionalidades desenvolvidas no *sprint.*

Caso o cliente, ao realizar os testes, detetasse algum defeito ou *bug*, este era reportado na plataforma JIRA para poder ser corrigido. Após a correção do *bug*, este era novamente testado em ambiente de desenvolvimento. Assim que o *bug* estivesse corrigido era feita novamente uma passagem a qualidade para o cliente voltar a validar. Este processo repetia-se até o cliente aceitar e aprovar os desenvolvimentos concluídos no *sprint*. Após a aprovação do cliente aos desenvolvimentos, era agendada com o cliente uma data para a passagem para produção.

5. ANÁLISE CRÍTICA E COMPARATIVA COM A LITERATURA

A finalidade desta análise crítica é estabelecer uma comparação entre a revisão de literatura e a aplicação da metodologia Scrum durante a realização das atividades de estágio.

As empresas estão, cada vez mais, a adotar metodologias ágeis para o desenvolvimento e gestão dos seus projetos, de forma a conseguirem satisfazer as necessidades dos seus clientes através de entregas rápidas e contínuas de produto e também para melhorarem a sua resposta à mudança de requisitos ao longo do processo de desenvolvimento (McCormick, 2012). Dentro dos métodos ágeis existentes, o Scrum é a metodologia mais adotada pelas empresas (Digital.ai, 2021).

No projeto principal de estágio, a metodologia implementada foi o Scrum. Apesar de terem sido seguidas muitas das diretrizes do Scrum, nomeadamente todo o processo de registo e organização dos requisitos, a realização de ciclos de desenvolvimento iterativos e curtos e a realização das diferentes cerimónias ao longo dos *sprints*, existiu a necessidade de adaptar esta metodologia à realidade da empresa e do projeto.

Segundo os conceitos do Scrum descritos na revisão da literatura, esta metodologia apenas apresenta três papeis principais: o *Scrum Master*, o *Product Owner* e a Equipa de Desenvolvimento (Alsaqqa et al., 2020; Andrei et al., 2019). Esta metodologia não inclui a existência de um Gestor de Projeto ágil. Na instituição de acolhimento, a função de Gestor de Projeto ágil tem como responsabilidades a gestão do portfólio de projetos, o controlo do orçamento e a coordenação do *Product Owner*. Desta forma, a equipa não é totalmente auto-organizada, uma vez que o orçamento é controlado por este Gestor.

Outra adaptação feita ao Scrum, está relacionada com o papel da *Product Owner*. O papel do *Product Owner* é o de ter a visão do negócio e ser o representante do cliente, priorizar e gerir o *Product Backlog*, assim como definir as metas da equipa e aceitar ou rejeitar as entregas de produto (Alsaqqa et al., 2020; Andrei et al., 2019). No projeto principal, não existiu nenhum elemento da equipa de desenvolvimento alocado especificamente para a realização das tarefas de testes, desta forma o papel de *tester* foi assumido pela estagiária e pela *Product Owner*. É bastante comum, na instituição de acolhimento, este tipo de atividades ser da responsabilidade da equipa funcional, pelo que muitos dos colaboradores que desempenham as funções de

Analista Funcional e de *Product Owner* apresentam certificações e conhecimento para a realização de testes.

A alocação de um período específico do *sprint* para a correção de *bugs* , também referido como *buffer*, é algo que também não está descrito nos conceitos do Scrum. No entanto, esta prática é bastante útil para a prevenção da acumulação de problemas no *Product Backlog*.

Por fim, a realização de uma reunião diária adicional com o cliente também é um ponto de diferença praticado no projeto. Dentro das cerimónias praticadas no Scrum apenas é contemplado a realização de *Daily Scrum Meetings* com a equipa (Alsaqqa et al., 2020).

6. CONCLUSÕES

Numa primeira instância, é importante salientar a concretização de todos os objetivos propostos inicialmente para o estágio e que foram descritos ao longo do presente relatório. A forte componente experimental do estágio, a autonomia dada para a realização de todas as tarefas e o acompanhamento por parte da orientadora potenciaram o desenvolvimento e aquisição de competências técnicas e profissionais.

Foi também possível, ao longo do estágio, aplicar os conhecimentos adquiridos no mestrado em Gestão de Sistemas de Informação. As aprendizagens adquiridas ao longo do percurso académico foram fundamentais, sendo mesmo uma mais-valia ao facilitarem a concretização de todas as atividades realizadas no estágio.

Outro ponto importante, foi igualmente a participação e a colaboração em diferentes atividades e projetos, o que proporcionou a possibilidade de contacto com diferentes ferramentas e técnicas, assim como o confronto com problemas reais de trabalho e de gestão de projeto.

A participação e a colaboração durante o estágio contribuíram, de certo modo, para o sucesso e bom desempenho das diferentes atividades atribuídas. Assim, no decorrer do estágio foram sendo atribuídas atividades com cada vez mais responsabilidade e autonomia na realização das diferentes tarefas. De referir também que todos os desenvolvimentos propostos no projeto principal descrito no relatório foram cumpridos, assim como todos os prazos, encontrando-se a plataforma *web* desenvolvida já totalmente em produção.

Durante o período de estágio, foram também identificados alguns desafios por parte da estagiária, tanto a nível laboral como na redação do relatório.

Relativamente à experiência de estágio, um dos principais desafios foi o facto de este ter ocorrido maioritariamente em formato remoto, tendo dificultado a integração da estagiária na cultura da empresa, com a orientadora e com os restantes membros de equipa. Contudo, o facto de terem sido realizadas várias sessões de acompanhamento com a orientadora no decorrer do estágio ajudou a ultrapassar essa falta de contacto presencial. Outro aspeto que contribuiu para ultrapassar esta limitação, foi a utilização da metodologia Scrum no projeto, que deu a possibilidade de conhecer de perto os diferentes papéis que cada membro da equipa

desempenhava, contribuindo para uma melhor comunicação entre os vários elementos técnicos da equipa.

Outra limitação a salientar está relacionada com uma atividade especifica do estágio, a realização de testes. A elaboração e execução dos casos de teste eram realizadas de forma manual na plataforma JIRA, o que exigia um grande esforço por parte da *Product Owner* e da estagiária. A criação destes casos de teste poderia ter sido feita de uma forma mais eficaz e automatizada, recorrendo à opção de importação de "Casos de Teste", a partir de documentos do tipo CSV, utilizando a funcionalidade "*Test Case Importer*" do XRay.

No que concerne à redação do relatório de estágio, uma limitação final identificada foi o facto da estagiária não poder partilhar algumas informações relativas à empresa e aos projetos em que esteve inserida, por se tratar de informações de cariz confidencial, o que dificultou a descrição pormenorizada das atividades desenvolvidas durante o estágio.

Contudo, a estagiária encarou todas os desafios e limitações apresentados como uma oportunidade de aprendizagem para aplicar no seu futuro profissional.

Em síntese, pode afirmar-se que a experiência de estágio correspondeu às expectativas da estagiária, uma vez que conseguiu potenciar as suas competências e desenvolver conhecimentos em novas tecnologias e metodologias, assim como entrar em contacto com o mercado de trabalho. No final do estágio, e como consequência, foi apresentada à estagiária uma proposta para continuar o percurso profissional na instituição de acolhimento.

BIBLIOGRAFIA

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile Software Development Methods: Review and Analysis. *VTT Publications*, *478*, 3–107. https://doi.org/10.48550/arxiv.1709.08439

Agile Academy. (n.d.). *Definition of User-story*. Retrieved July 16, 2022, from https://www.agile-academy.com/en/agile-dictionary/user-story/

Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies (IJIM)*, *14*(11), 246–270. https://doi.org/10.3991/ijim.v14i11.13269

Andrei, B.-A., Casu-Pop, A.-C., Gheorghe, S.-C., & Boiangiu, C.-A. (2019). A STUDY ON USING WATERFALL AND AGILE METHODS IN SOFTWARE PROJECT MANAGEMENT. *JOURNAL OF INFORMATION SYSTEMS & OPERATIONS MANAGEMENT*.

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern. Jon, Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Coram, M., & Bohner, S. (2005). The impact of agile methods on software project management. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 363–370. https://doi.org/10.1109/ECBS.2005.68

Digital.ai. (2021). *15th State of Agile Report*.

Gaborov, M., Karuović, D., Kavalić, M., Radosav, D., Milosavljev, D., Stanisaljev, S., & Bushati, J. (2021). Comparative analysis of agile and traditional methodologies in IT project management. *Journal of Applied Technical and Educational Sciences*, *11*(4), 1–24. https://doi.org/10.24368/jates.v11i4.279

Javanmard M, & Alian M. (2015). Comparison between Agile and Traditional software development methodologies. *Journal*, *36*(3), 1386–1394.

José, A., Andrade, F., de Oliveira, J. C., Alberto, P., Barbosa, M., Raquel, F., & Silveira, V. (2011). *Gestão de Projeto com Scrum: Um Estudo de Caso*.

Kumar, G., & Bhatia, P. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE), 2*, 2249–6343.

McCormick, M. (2012). Waterfall vs. Agile Methodology. *MPCS, N/A, 3*.

Mirza, M., & Datta, S. (2019). Strengths and Weakness of Traditional and Agile Processes - A Systematic Review. *Journal of Software*, *14*, 209–219. https://doi.org/10.17706/jsw.14.5.209-219

Molina Ríos, J., & Pedreira-Souto, N. (2019). Approach of Agile Methodologies in the Development of Web-Based Software. *Information*, *10*(10). https://doi.org/10.3390/info10100314

Moniruzzaman, A. B. M., & Hossain, S. (2013). *Comparative Study on Agile software development methodologies*.

PMI. (2021). *The standard for project management and a guide to the project management body of knowledge (PMBOK guide).* Project Management Institute, Inc.

Salameh, H. (2014). What, When, Why, and How? A Comparison between Agile Project Management and Traditional Project Management Methods. *International Journal of Management Reviews, Vol.2,*.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.

Soares, M. (2005). *Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software*.

Sommerville, I. (2011). *Software Engineering* (9th edition). Pearson.

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*, 64–76. https://doi.org/10.12948/issn14531305/17.4.2013.06

Sutherland, J. (2014). *Scrum - A arte de fazer o dobro de trabalho na metade do tempo*. Texto Editores Ltda. www.leya.com.br

ANEXOS

I. Exemplo de User-story

[ANCTSI-57] #US005 – Como utilizador [redacted] pretendo criar e ocultar utilizadores, por forma a gerir os acessos ao BO	
Status:	DEPLOYED TO PROD
Project:	[redacted]
Components:	Nome
Affects versions:	None
Fix versions:	v1.6.0.0

Type:	Story	**Priority:**	Low
Reporter:	[redacted]	**Assignee:**	Maria Almeida Gomes
Resolution:	Done	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	0h	**Remaining Estimate:**	0h
Σ Time Spent:	34h	**Time Spent:**	Not Specified
Σ Original Estimate:	24h	**Original estimate:**	24h

Attachments:	image-20220218-153554.png			
Issue links:	**Gantt End to End**			
	has to be finished together with	~~ANCTSI-62~~	#US009 - Como utilizador do BO preten...	DEPLOYED TO PROD
	Relates			
	relates to	~~ANCTSI-288~~	#TC005 - Como [redacted] pretendo criar e...	READY FOR PROD
	relates to	~~ANCTSI-127~~	#US024 - Como [redacted] pretendo efetuar...	DEPLOYED TO PROD
	relates to	~~ANCTSI-312~~	Quando o utilizador está inativo a me...	DEPLOYED TO PROD

	relates to	~~ANCTSI-314~~	Ao consultar os acessos de um utiliza...	DEPLOYED TO PROD
	relates to	~~ANCTSI-315~~	O sistema permite criar múltiplos uti...	DEPLOYED TO PROD
	relates to	~~ANCTSI-332~~	Quando é criado um novo utilizador o ...	DEPLOYED TO PROD
	relates to	~~ANCTSI-387~~	#CR005 - Alterar a opção do menu late...	DEPLOYED TO PROD

Sub-tasks:

Key	Summary	Type	Status	Assignee
ANCTSI-296	Análise à user-story	Sub-task	READY FOR PROD	[redacted]
ANCTSI-297	FE - Criação do ecrã de pesquisa de u...	Sub-task	Ready for Dev	[redacted]
ANCTSI-298	BE - Criação dos métodos de pesquisa ...	Sub-task	Ready for Dev	[redacted]
ANCTSI-299	FE - Criação da página de detalhe/cri...	Sub-task	Ready for Dev	[redacted]
ANCTSI-300	BE - Criação do método de criação de ...	Sub-task	Ready for Dev	[redacted]
ANCTSI-301	BE - Criação do método de edição de u...	Sub-task	Ready for Dev	[redacted]
ANCTSI-302	FE - Propagação das novas permissões ...	Sub-task	DEPLOYED TO PROD	[redacted]
ANCTSI-303	BE - Criação das tabelas para as nova...	Sub-task	Ready for Dev	[redacted]
ANCTSI-304	BE - Inclusão das permissões na geraç...	Sub-task	DEPLOYED TO PROD	[redacted]

Epic Link:	BO
Sprint:	IT070101 Sprint 7, IT070101 Sprint 8

Description

Critérios de Aceitação

#AC005.1 – No BackOffice, ou seja, ao efetuar login na plataforma com um utilizador [redacted] com permissões de consulta ou edição de Utilizadores deverá existir no menu principal uma opção designada "Utilizadores" para visualizar e gerir Utilizadores [redacted].

#AC005.2 – Ao clicar na opção de "Utilizadores", o utilizador deverá aceder a um ecrã para efetuar uma pesquisa por e-mail dos utilizadores com acesso ao BO e por estado. No topo do ecrã, no canto superior direito, alinhado com o título "Utilizadores" deverá existir o botão "Adicionar". Por baixo da pesquisa deverá mostrar por defeito a lista de todos utilizadores adicionados no BO, ordenada por ordem alfabética e-mail com as colunas email e estado (ativo ou inativo) e ainda um botão "Acessos" por linhas.

#AC005.3 – Deverá existir um botão de pesquisa, que quando carregado, deverá mostrar uma lista de utilizadores cujo e-mail contenha os caracteres inseridos no campo de pesquisa. A lista deverá estar ordenada por ordem alfabética de nome.

#AC005.4 – A lista de utilizadores deverá conter paginação para percorrer a lista, idêntica à paginação da lista de pedidos.

#AC005.5 – Ao pressionar em "Acessos" deverá aparecer um título no topo do ecrã "Acessos de [e-mail do utilizador]".

#AC005.6 – Deverá existir um botão de voltar que reencaminha o utilizador para o ecrã da Pesquisa de Utilizadores com os filtros e número de página guardados.

#AC005.7 – O ecrã deverá conter uma matriz de acessos onde cada linha deverá corresponder a uma opção de menu (pedidos de elegibilidade, utilizadores, prestadoras (ainda não existe), configurações (ainda não existe), etc) e as colunas deverão ser: consultar e editar. Deverá ainda existir um campo "estado" que indica se o utilizador está ativo para se poder autenticar ou inativo. No final deverá existir um botão de "editar".

#AC005.8 - Ao editar, o utilizador poderá acionar as permissões que pretender para determinado utilizador, sendo que, caso o utilizador selecione uma permissão de edição, a opção de consulta deverá ser acionada automaticamente. Não deverá existir edição sem consulta. Poderá ainda alterar o estado do utilizador, ativo ou inativo (dropdown list).

#AC005.9 – No final do ecrã de Acessos de Utilizador, deverá existir um botão de gravar, de forma a que o utilizador submeta as suas alterações.

#AC005.10 Um utilizador com permissões apenas de consulta não deverá visualizar nenhum botão de ação, como por exemplo "editar", "adicionar" ou "criar", nem deverá visualizar o card "elegibilidade de estudantes" para validação de pedidos com estado "aguarda [redacted]".

#AC005.11 - Ao pressionar em "Adicionar", deverá surgir um ecrã como título "Adicionar Utilizador", semelhante ao ecrã de edição. Deverá existir os campos para preenchimento:

“E-mail” com placeholder “Insira o e-mail”, e campo estado em forma de dropdownlist. o de seguida a tabela de acessos para preenchimento.

1 Caracterização da História / Interfaces gráficos

1.1 Ecrã de Utilizadores

1.1.1 Elementos do Ecrã de pesquisa:

1.1.1.1. Filtros de Pesquisa

Campo	Tipo de Informação	Descrição
E-mail	Texto Alfanumérico	Campo de pesquisa por *Email* do utilizador (1)
Estado	*dropwdownlist*	Campo de pesquisa (1) (2)

(1) Não é *case sensitive*

(2) *dropwdownlist* por ordem alfabética

1.1.1.2 Tabela de Resultados

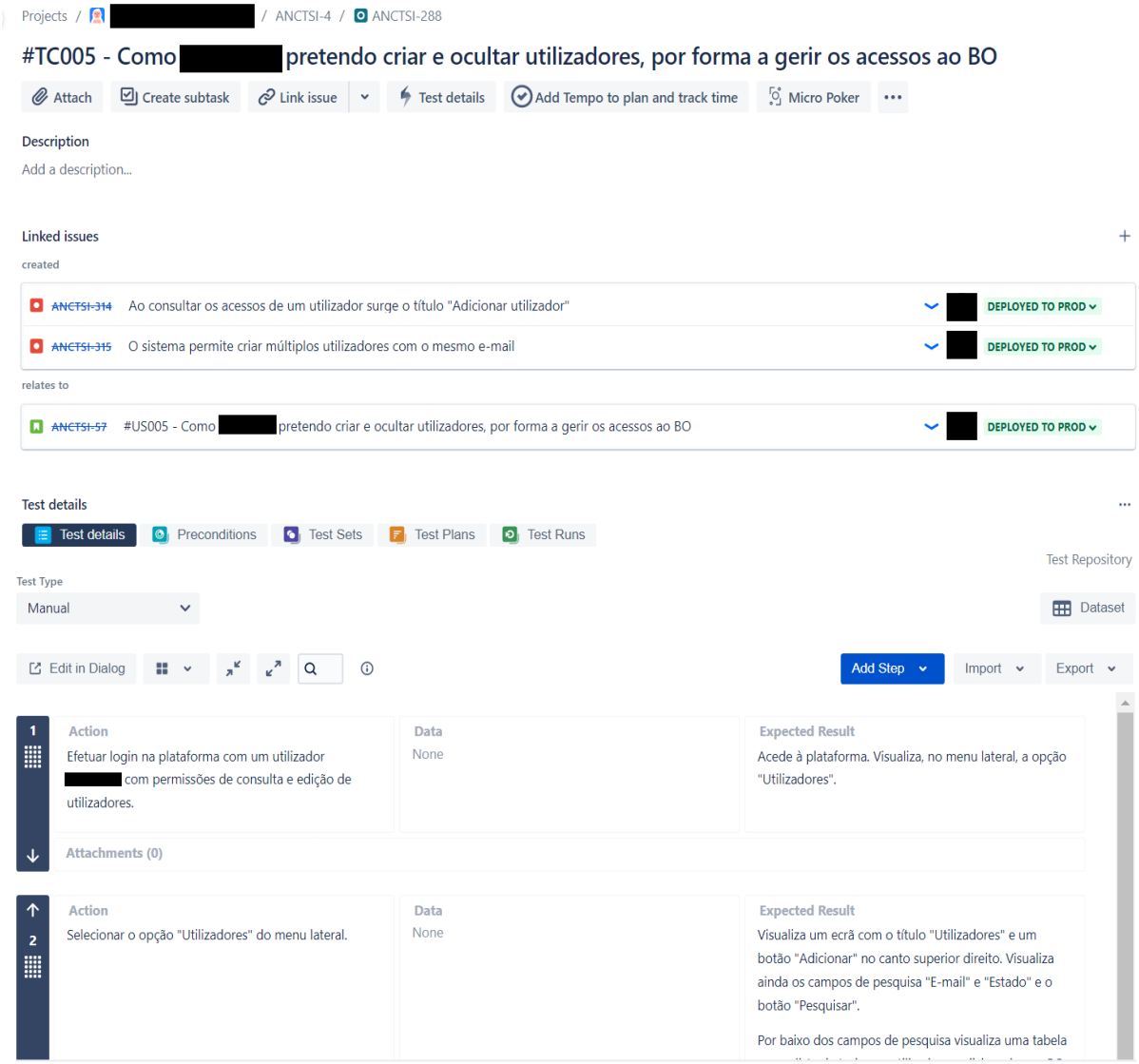
Coluna	Tipo de Informação	Descrição
E-mail	Texto Alfanumérico	*E-mail* do utilizador
Estado	Texto Alfanumérico	Indica se o utilizador tem acesso ao BO *(ativo ou inativo)*
Acessos *(botão)*	Botão	Botão que serve para abrir a página de Detalhe do Utilizador

1.1.2 Elementos do Ecrã de edição e adição

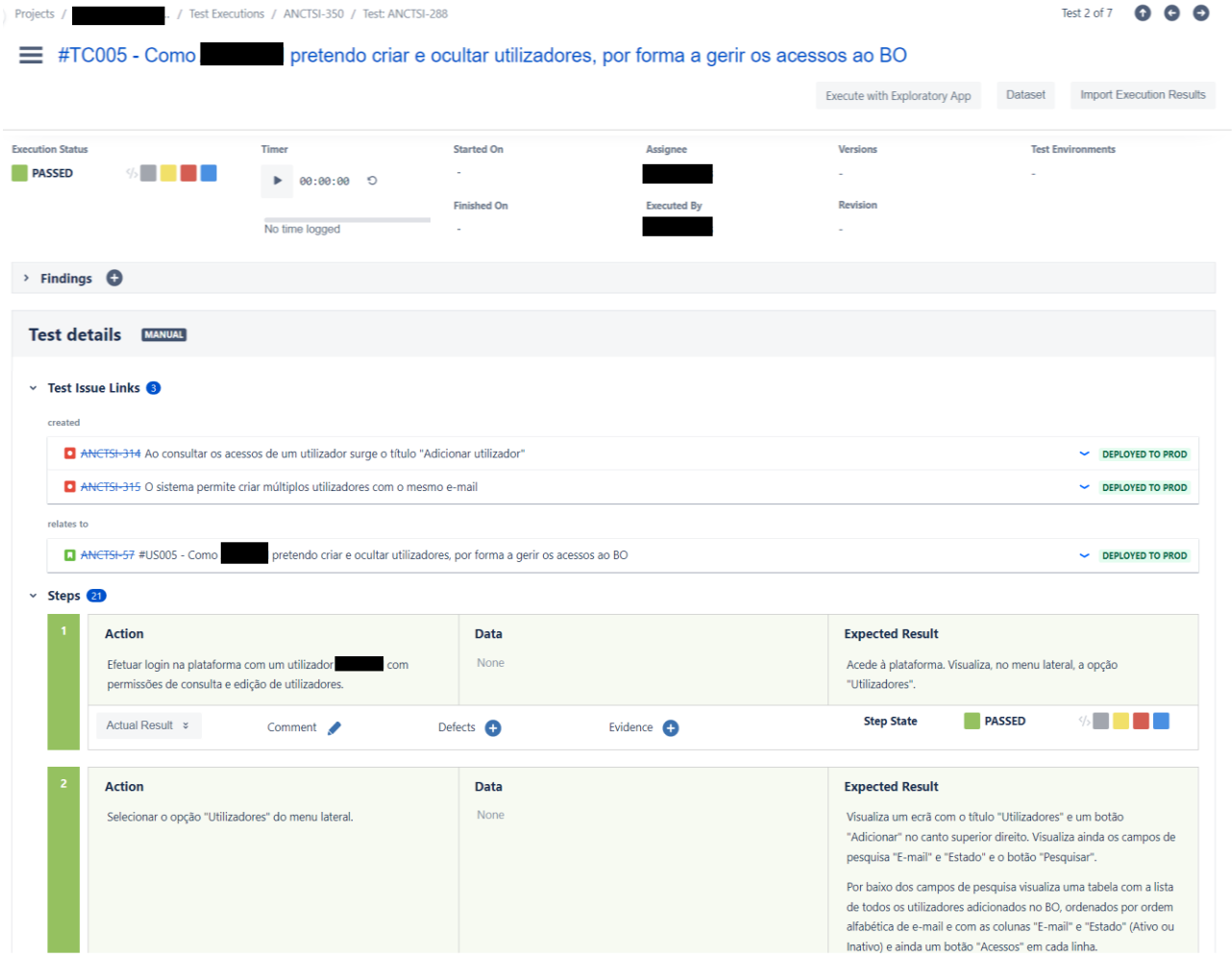
Campo	Tipo de Informação	Obrigatório	Descrição
E-mail	Texto Alfanumérico	S	*E-mail* do utilizador
Estado	Dropdownlist	S	Indica se o utilizador tem acesso ao BO *(ativo ou inativo)*
Matriz de acessos	botões	S (pelo menos uma permissão)	Botões que servem para acionar as permissões de acesso do utilizador (deverão ser semelhantes ao anexo, com botões parecidos ao pop-up de pesquisa de moradas do formulário de pedido)

Botão	Descrição
Botão "Pesquisar"	Botão que permite pesquisar os utilizadores registados no BO
Botão "Acessos"	Botão que permite aceder às permissões de acesso do utilizador.
Botão "Voltar"	Permite regressar ao ecrã da Pesquisa de Utilizadores
Botão "Gravar"	Permite gravar as alterações efetuadas no Detalhe do Utilizador
Botão "Adicionar"	Permite adicionar um utilizador à listagem

II. Exemplo de Caso de Teste



III. Exemplo de Execução de Caso de Teste



IV. Exemplo de Bug

Projects / [redacted] / ANCTSI-315

O sistema permite criar múltiplos utilizadores com o mesmo e-mail

Attach | Create subtask | Link issue | Add Tempo to plan and track time | Micro Poker | SmartDraw Connector | ...

Description

Com e-mails iguais como é feita a correlação com o utilizador real?
ver anexo.

Caso o e-mail já exista na BD (seja de operador, seja de administrador), ao pressionar em gravar no ecrã de adicionar, deverá surgir a mensagem de alerta "O e-mail já existe" e não deverá deixar avançar.

Environment

DEV

Attachments (2)