Instituto Politécnico de Coimbra
Instituto Superior de Contabilidade
e Administração de Coimbra

Miguel Alves Almeida

Produtividade no desenvolvimento de *software*: um estudo laboratorial

Coimbra, junho **de** 2021

Produtividade no desenvolvimento de software: um estudo laboratorial
Miguel Alves Almeida
ISCAC | 2021

Instituto Politécnico de Coimbra
Instituto Superior de Contabilidade
e Administração de Coimbra

Miguel Alves Almeida

Produtividade no desenvolvimento de *software*: um estudo laboratorial

Trabalho de projeto submetida(o) ao Instituto Superior de Contabilidade e Administração de Coimbra para cumprimento dos requisitos necessários à obtenção do grau de **Mestre em Sistemas de Informação de Gestão**, realizada(o) sob a orientação do Professor Doutor António Trigo e o Professor Doutor João Varajão

Coimbra, junho de 2021

TERMO DE RESPONSABILIDADE

Declaro ser o(a) autor(a) deste projeto, que constitui um trabalho original e inédito, que nunca foi submetido a outra Instituição de ensino superior para obtenção de um grau académico ou outra habilitação. Atesto ainda que todas as citações estão devidamente identificadas e que tenho consciência de que o plágio constitui uma grave falta de ética, que poderá resultar na anulação do presente projeto.

PENSAMENTO

"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time." - Thomas A. Edison

AGRADECIMENTOS

Sempre me disseram que sozinhos nunca seriamos nada. Sendo sincero, das primeiras vezes que ouvi isto não liguei muito, talvez por nessa altura não ter maturidade para tal. Hoje, o desenvolvimento deste projeto é um reflexo disso mesmo, e com isto deixo abaixo alguns agradecimentos.

Ao meu orientador, professor Doutor António Trigo, por ter sido para além de um professor um amigo, que sempre acreditou no meu potencial e nunca me deixou desistir, motivando-me em cada momento. Pelo seu rigor científico, persistência, interesse permanente, empenho que contribuiu para que este projeto fosse concluído.

Ao meu coorientador, professor Doutor João Eduardo Varajão, por ter estado sempre presente em todos os momentos de criação, revelando-se cientificamente exigente e expressando sempre os seus pensamentos de forma a acrescentar valor ao projeto.

Aos meus pais, irmã e namorada pelo amor. Amor esse que esteve sempre presente em todo o projeto e que me deu forças para superar todas as dificuldades que enfrentei. Um obrigado muito especial aos meus pais, por tudo o que me ofereceram e por sempre acreditarem que eu era capaz, mesmo quando tudo indicava o contrário.

Aos meus avós, que mesmo não sabendo muito bem o que é isto da "tese", perguntavam-me todos os fins-de-semana como estava a correr e diziam-me sempre que sabiam que ia correr bem.

Aos meus amigos, por estarem sempre presentes na minha vida, em especial ao Ricardo Quinteiro e André Oliveira, por demonstrarem mais uma vez que acreditam em mim e também por me terem ajudado na criação deste projeto.

RESUMO

As tecnologias de informação fazem parte do nosso quotidiano e a cada dia que passa dependemos mais delas para as nossas atividades. O desenvolvimento de software foi evoluído ao longo do tempo com o objetivo de atender a essas necessidades. Em paralelo, as empresas tecnológicas viram-se obrigadas a adotar métodos e tecnologias que as tornassem mais produtivas e ao mesmo tempo, competitivas no mercado.

Com as aplicações de *software* cada vez mais complexas, é importante para as empresas entender quais as tecnologias e/ou plataformas permitem uma maior produtividade, custos mais reduzidos, tempos de desenvolvimento mais curtos, e menos recursos.

Neste contexto, no presente trabalho é avaliada a produtividade no desenvolvimento de software envolvendo diferentes tipos de tecnologia de desenvolvimento de software (*source code*, *low-code*) existentes no mercado, através da realização de uma experiência laboratorial que envolveu o desenvolvimento de uma aplicação de *software* em ambiente controlado.

Os resultados deste trabalho permitem concluir que o desenvolvimento e manutenção de software com tecnologia *low-code* é cerca de três vezes mais rápido do que com tecnologia *source-code*, o que corrobora a pouca literatura existente na área. Estes resultados são importantes para as organizações que desenvolvem software e mostram que estas devem no futuro ter em conta as tecnologias *low-code*, como alternativa às soluções mais convencionais baseadas em tecnologias *source code*.

Como limitações do trabalho identifica-se o protocolo do trabalho, as tecnologias utilizadas e os profissionais envolvidos dado que o seu perfil poderá ter influenciado os resultados.

PALAVRAS-CHAVE: Desenvolvimento de *Software*, Produtividade, *Source code*, *Low-code.*

ABSTRACT

Information technology is an important part of our everyday life and, moving forward, our daily activities rely ever more on it. Software development has been evolving through time with the goal of satisfying those requirements. In parallel, technology companies have been forced to adopt methods and technologies that make them more productive, to be competitive in the market.

With the complexity and size of software growing, it is important for companies to understand which technologies and/or platforms enable better productivity levels, reduced costs, shorter development times and fewer resources.

In this context, the present work evaluates the productivity in software development involving different types of software development technology (source code, low-code) existing in the market, through the realization of a laboratory experiment that involved the development of a software application in a controlled environment.

The results of this research show that developing and maintaining software with low-code technologies is about three times faster than with source code technologies, corroborating the small amount of existing literature in this field. These results are important for software development organizations and show that in the future they should consider low-code technologies as an alternative to the more conventional technologies based on source code.

The limitations of the study include the protocol, the technologies used, and the professionals involved, given that their profile may have influenced the results.

KEYWORDS: Software Development, Productivity, Source code, Low-code

ÍNDICE GERAL

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

Lista de abreviaturas, acrónimos e siglas

No presente documento são utilizadas as seguintes abreviaturas, acrónimos e siglas:

API - *Application Programming Interface*

COCOMO - *Constructive Cost Model*

COCOMO II - *Constructive Cost Model II*

EF - *Environmental Factor*

FTP - *File Transfer Protocol*

IFPUG - *International Function Point Users Group*

PaaS - *Platform-as-a-service*

PF - *Productivity Factor*

SDLC - *Software Development Life Cycle*

SLOC - *Software Lines of Code*

SRS - *Software Requirements Specification*

TCF - *Technical Complexity Factor*

TCP - *Transmission Control Protocol*

TSI - Tecnologias e Sistemas de Informação

TI - Tecnologias de Informação

UAW - *Unadjusted Actor Weight*

UCP - *Use Case Points*

UML - *Unified Modeling Language*

UUCP - *Unadjusted Use Case Points*

UUCW - *Unadjusted Use Case Weight*

VAF - Valor do Fator de Ajuste

1 INTRODUÇÃO

Este primeiro capítulo apresenta uma visão global do trabalho desenvolvido. Começa-se por enquadrar e justificar o tema escolhido, de seguida são apresentadas as motivações para a realização do projeto e os objetivos propostos, apresentando-se, por fim, a estrutura do relatório.

1.1 Enquadramento

As Tecnologias e Sistemas de Informação (TSI) desempenham um papel fundamental nas organizações, sendo poucas aquelas que conduzem os seus negócios sem a utilização das TSI (Varajão et al., 2009a; Varajao et al., 2009b). Desde os primeiros *mainframes* nos anos 60, ao computador pessoal nos anos 80, à *Internet of Things* dos dias de hoje, a adoção das TSI tem verificado um grande crescimento.

Associada às TSI surgiu a indústria de desenvolvimento de software que, tal como as TSI, tem crescido a um ritmo elevado nos últimos anos. A título de exemplo deste crescimento, nos Estados Unidos, o número de profissionais associados ao desenvolvimento de sistemas computacionais cresceu, entre 1991 e 2021 (fevereiro), de 409 mil para 2200 mil profissionais.

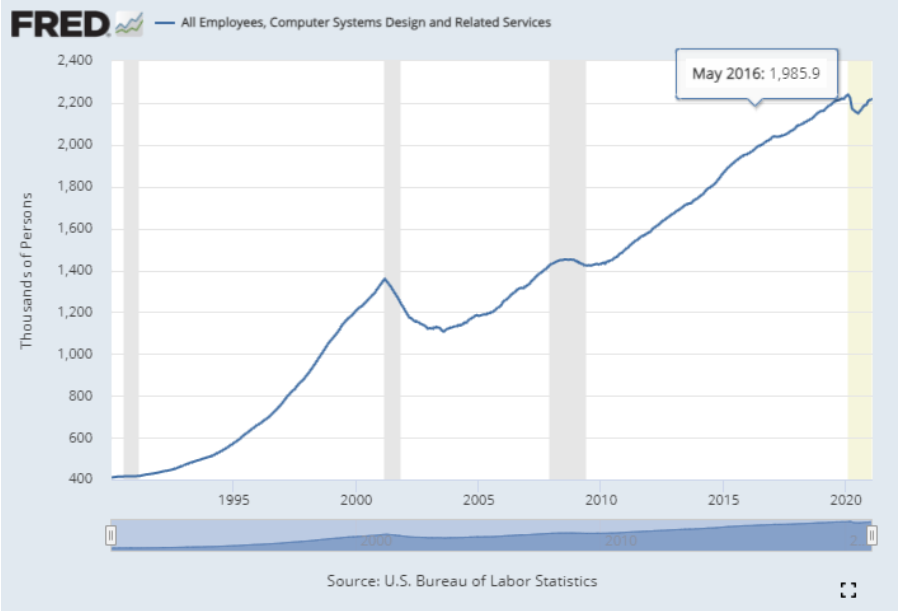


Figura 1.1. Número de profissionais associados ao desenvolvimento de sistemas computacionais entre 1991 e 2021

Fonte: FRED (2021)

O crescente volume e complexidade do *software* estimulam a criação de métodos e tecnologias de desenvolvimento mais expeditos, que permitam às empresas de desenvolvimento de software ser mais produtivas.

Tendo em conta esta realidade, várias plataformas *low-code* foram surgindo com o intuito de agilizar o desenvolvimento de *software*. Segundo a Gartner, o *low-code* será responsável por mais 65% da atividade de desenvolvimento de *software* até 2024 (Outsystems, 2021). Estas plataformas têm a particularidade de conseguirem oferecer uma resposta adaptada às exigências atuais de mercado, velocidade e agilidade nos negócios, sendo esse o principal motivo de interesse por parte das empresas (Outsystems, 2021). Não obstante popularidade crescente das plataformas *low-code*, são ainda escassos os estudos que comprovem a sua maior produtividade.

1.2 Motivações e objetivos

Embora as ferramentas de *low-code*, como a *OutSystems* (Outsystems, 2021), ganhem cada vez mais popularidade, no melhor do nosso conhecimento, não existem estudos científicos que comparem a produtividade do desenvolvimento de software com tecnologias *source code* e *low-code*.

Posto isto, estabelecem-se como finalidade deste trabalho realizar um estudo comparativo de produtividade das tecnologias de desenvolvimento de *software source code* e *low-code.*

Para investigar esta questão foi concebida uma experiência laboratorial com duas fases, uma referente ao desenvolvimento de software e outra à manutenção de software, em que dois programadores experientes nas tecnologias a testar, designadamente *Outsystems (Low-Code)* e *Django/Python (Source Code)*, implementaram uma aplicação seguind o mesmo protocolo.

A medição da produtividade foi feita tendo como referência a Análise de Pontos de Caso de Uso, que tem como objetivo estimar o tamanho e a complexidade de um determinado *software* tendo por base os seus requisitos funcionais apresentados sob a forma de casos de uso e de modo independente da tecnologia utilizada.

1.3 Estrutura do relatório

O presente relatório está organizado em cinco capítulos, que traduzem o curso dos trabalhos desenvolvidos para o cumprimento dos objetivos definidos.

O primeiro capítulo apresenta o enquadramento, os objetivos gerais e a estrutura do relatório.

No segundo capítulo, a revisão de literatura, apresenta os fundamentos teóricos relativos à produtividade no desenvolvimento de *software.* Na primeira secção, são apresentados conceitos associados ao desenvolvimento de *software* e algumas metodologias associadas. Na segunda secção, são abordadas diferentes tecnologias para o desenvolvimento de *software*. A terceira secção, foca a produtividade no desenvolvimento de *software,* abordando alguns fatores que podem ter influência na mesma. Por fim, na quarta secção, são apresentadas algumas abordagens utilizadas para estimar o esforço de desenvolvimento de *software*.

No terceiro capítulo, apresenta-se a metodologia que sustentou a realização deste projeto, descrevendo-se as experiências realizadas.

No quarto capítulo, são apresentados e discutidos os resultados, focando as aplicações desenvolvidas, em ambas as experiências, e a realização de uma análise comparativa do desenvolvimento das aplicações nos dois tipos de tecnologias selecionadas (*source code* e *low-code*), com o objetivo de avaliar a produtividade.

Por último, o quinto capítulo, apresenta as considerações finais. É feita uma síntese do trabalho desenvolvido e são apresentadas limitações propostas de trabalho futuro.

2 REVISÃO DA LITERATURA

O *software* é omnipresente nas organizações dado que a sua evolução depende cada vez mais da existência de sistemas de informação em todos os níveis de gestão, sendo poucas as organizações que conduzem os seus negócios sem procurar explorar as vantagens da utilização de tecnologias da informação (Varajão et al., 2009). Muitas organizações procuram As soluções de *software* apresentam requisitos cada vez mais complexos, requerendo um maior cuidado no seu planeamento (Pressman, 2009).

2.1 Desenvolvimento de *software*

"*Software* consiste em: instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados (…)" (Pressman, 2009, p. 32). As aplicações de *software* são desenvolvidas para dar suporte a alguma necessidade, seja ela organizacional ou individual.

O processo de desenvolvimento é complexo e, por esse motivo, é importante que existam etapas bem definidas, bem como recursos humanos que sejam capazes de planear e implementar o produto desejado (Pimparkhede, 2018). Ao longo do tempo foram apresentados diferentes modelos para dar resposta às dificuldades existentes, com o objetivo de obter, uma maior eficiência no processo de desenvolvimento (Stoica et al., 2013).

2.1.1 *Software Development Life Cycle (SDLC)*

O ciclo de vida de desenvolvimento de *software*, *Software Development Life Cycle* (SDLC) em inglês, foi elaborado com o intuito de promover fases mais claras nos processos de desenvolvimento e manutenção de *software* (Pimparkhede, 2018), podendo ser dividido em seis etapas (Shylesh, 2017): Planeamento; Definição; Conceção; Implementação; Teste; e, Instalação e Manutenção.

O SDLC pode sofrer alguma alteração consoante a abordagem utilizada, mantendo sempre o mesmo objetivo, a entrega de valor para o cliente (Alshamrani & Bahattab, 2015).

2.1.2 Metodologias clássicas

As metodologias clássicas podem ser divididas em dois grupos alvo (Pressman, 2009): modelos sequenciais e modelos evolucionários. Os modelos sequenciais, como o próprio nome indica, têm uma única sequência de etapas, ou seja, não é possível passar à próxima etapa sem terminar a anterior. Um bom exemplo desta metodologia é o Modelo Cascata. Quanto aos modelos evolucionários, estes são caracterizados por serem iterativos e possibilitarem a entrega de várias versões, cada vez mais completas. Isto permite que o produto evolua ao longo do tempo. Uma metodologia que possui estas características é o Modelo Espiral.

2.1.2.1 *Modelo Cascata*

O Modelo Cascata segue uma abordagem sequencial. Segundo Sommerville (2011), este modelo pode ser dividido em cinco etapas distintas: análise e especificação de requisitos; *design*; implementação e testes unitários; integração e testes de sistema; e manutenção.

A análise e especificação de requisitos tem como objetivo obter detalhadamente a informação acerca do que é pretendido pelo cliente. Posteriormente, é realizado um estudo de viabilidade onde são analisados os aspetos técnicos e económicos que podem impactar o processo de desenvolvimento de *software*, tais como os recursos necessários, tarefas a realizar e estimativas de tempo. Tendo em conta os requisitos analisados, segue-se a etapa de *design*. Nesta etapa é projetada a arquitetura do projeto, isto é, os requisitos são traduzidos em representações gráficas. Segue-se a etapa mais longa, a de implementação. Já com o projeto bem definido, utilizando a linguagem previamente escolhida, o código-fonte é escrito. Ou seja, as representações gráficas são transformadas em código funcional. O projeto é desenvolvido em pequenas funcionalidades denominadas por unidades, que serão posteriormente testadas e integradas. Quando forem integradas todas as unidades, o código é entregue à equipa de testes. Nesta etapa, são efetuados testes ao sistema de forma a corrigir eventuais erros antes da colocação em produção. O cliente também está envolvido nesta fase, para verificar se todos os requisitos acordados foram implementados. A etapa de manutenção tem como objetivo dar suporte e manutenção ao *software* após a sua entrega (colocação em produção), procurando garantir que este funciona corretamente.

2.1.2.2 *Modelo Espiral*

O modelo espiral resultou da combinação entre modelos iterativos e modelos sequenciais, tendo como principal objetivo controlar riscos (Boehm, 2007). O modelo em espiral, ao contrário do modelo cascata, propõe o desenvolvimento do *software* em iterações, em que em cada iteração se desenvolve um conjunto de funcionalidades a que se dá o nome de protótipo de forma iterativa vai-se desenvolvendo o *software* até ao protótipo (versão) final do *software*.

Cada iteração que leva ao desenvolvimento de um protótipo do *software* subdivide-se em quatro etapas (Boehm, 1988):

1. Determinar de objetivos, alternativas e restrições;
2. Avaliar alternativas, identificar e resolver riscos;
3. Desenvolver e verificar próximo nível do produto;
4. Planear a próxima fase.

Este modelo é visto como uma espiral e não como um círculo pelo facto de em cada iteração em que se desenvolve um novo protótipo, se acrescentarem funcionalidades ao protótipo anterior, até se chegar ao protótipo (versão) final do *software*.

2.1.3 Metodologia ágeis

As metodologias ágeis tem tido uma grande adoção por parte das empresas de desenvolvimento de *software* (Laranjeira et al., 2019). Essa adoção deve-se ao facto desta metodologia ser iterativa, ou seja, os requisitos do projeto podem sofrer alterações ao longo do tempo, dependendo do *feedback* da última iteração, não estando assim dependentes dos requisitos inicialmente acordados (Leau et al., 2012). Dessa forma, existe a necessidade de colaboração próxima entre ambas as partes, cliente e equipa de desenvolvimento. Este tipo de desenvolvimento leva a que as equipas entreguem valor mais rapidamente e com maior qualidade (Alahyari et al., 2017), o que teve repercussão ao nível da entrega de *software*, com o surgimento de um novo conceito – *DevOps* – que propõe a existência de cadeias integradas de entrega de *software*, desde o desenvolvimento à operação do mesmo (Sousa et al., 2019).

2.1.3.1 *Scrum*

O *Scrum* é apresentado como uma *framework* simples, que ajuda as pessoas, equipas e organizações a gerar valor através de soluções adaptativas para problemas complexos (Schwaber & Sutherland, 2017).

Nesta *framework*, a equipa responsável pelo desenvolvimento do produto é constituída por um *Product Owner*, um *Scrum Master* e pela equipa de desenvolvimento, designada por *Scrum Team*. O *Product Owner* é responsável por clarificar o que se pretende implementar e de que forma se deve proceder. Tudo isso é refletido numa lista que apresenta as funcionalidades do produto, denominada por *Product Backlog*, organizada por prioridade. O *Scrum Master*, resumidamente, tem como objetivo ajudar todas as pessoas envolvidas a entender todo o processo *Scrum*. A equipa de desenvolvimento é responsável pelo desenvolvimento do produto (Schwaber & Sutherland, 2017).

O desenvolvimento do produto é dividido em *Sprints*. Um *Sprint* é um conjunto de funcionalidades que têm de ser realizadas num determinado período de tempo, tipicamente 2 a 4 semanas (Schwaber & Sutherland, 2017). Para cada *Sprint* é feito um planeamento onde são discutidas quais as funcionalidades a desenvolver, elaborando-se uma lista a que se dá o nome de *Product Backlog*. Quando o *Sprint* termina, uma parte do produto é entregue e é iniciado um novo *Sprint*. Ao longo do tempo, algumas funcionalidades podem sofrer alterações, sendo assim inseridas, novamente, no *Product Backlog* e introduzidas num próximo *Sprint*. Este processo é repetido até não existir mais nenhuma funcionalidade no *Product Backlog*, indicando assim que o produto está finalizado.

Todos os dias é realizada uma reunião breve onde cada elemento da equipa deve responder a três perguntas básicas (Schwaber & Sutherland, 2017):

- O que fiz ontem?
- O que vou fazer hoje?
- Existem impedimentos na realização de alguma tarefa?

Desta forma, todos os elementos conseguem ter uma visão clara da progressão do *Sprint* e quais os pontos que devem ter uma maior atenção.

2.1.3.2 *Kanban*

O *Kanban* é um método ágil e visual que permite gerir e conduzir o fluxo de trabalho (Ahmad et al., 2018). O quadro de *Kanban* é constituído por colunas e cartões. Cada coluna tem um estado, previamente definido, por exemplo *To Do/In Progress/Done*. Os cartões descrevem as tarefas e são inseridos numa coluna, mediante o estado das mesmas (*To Do/In Progress/Done*). Estes cartões podem ter cores diferentes, em função da criticidade das tarefas, tendo assim cada tarefa uma prioridade diferente.

2.1.4 Comparação entre metodologias clássicas e ágeis

As metodologias clássicas, da qual a metodologia em cascata é o melhor exemplo, apresentam apenas um ciclo sequencial de atividades bem compartimentadas (ver Figura 2.1), sendo a versão final do produto entregue ao cliente no fim do ciclo de desenvolvimento. Já nas metodologias ágeis é evidente a existência de várias iterações. Neste tipo de abordagem são entregues as várias funcionalidades do produto ao cliente, de forma incremental, até este estar finalizado.

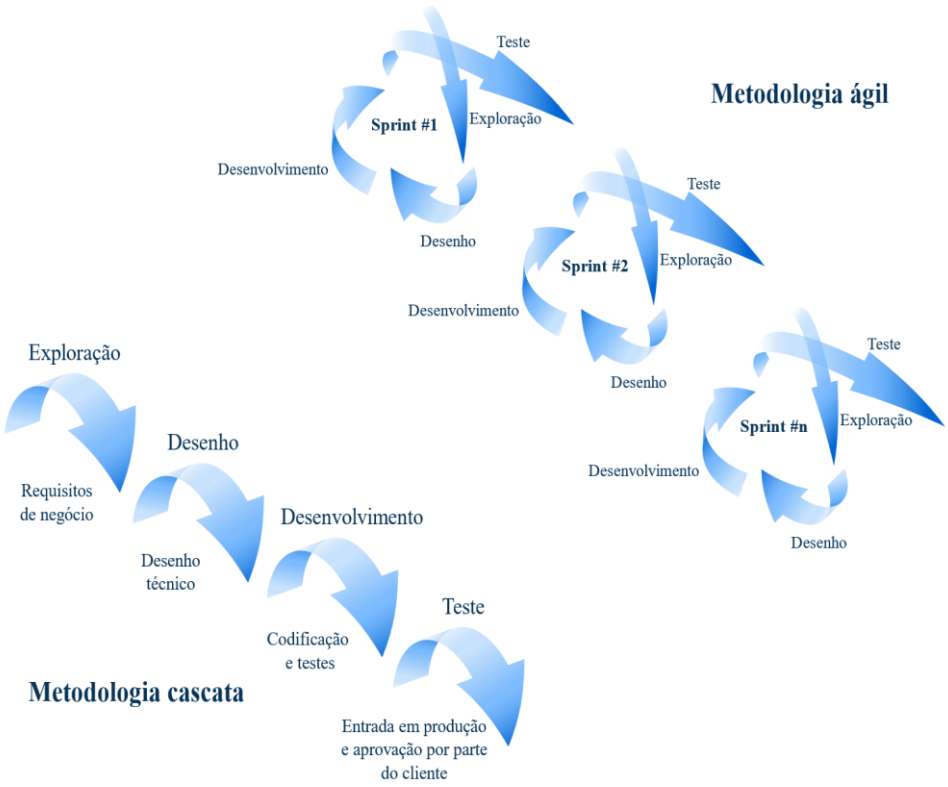


Figura 2.1. Metodologias clássicas vs. ágeis

Fonte: Sousa et. al (2019) e STH (2019)

Ambas as metodologias são utilizadas nos dias de hoje (Laranjeira et al., 2019), existindo, no entanto, diferenças entre elas, como se pode verificar na Tabela 2.1.

Tabela 2.1. Comparação entre as metodologias clássicas e ágeis

Características	**Metodologia clássicas**	**Metodologia ágeis**
Planeamento	Embrionário	Em cada iteração
Escala de Projeto	Pequena/Média escala	Grande escala
Teste	Posterior à fase de desenvolvimento	Em cada iteração
Mudança	Resistente	Flexíveis
Comunicação	Inibe	Estimula

Fonte: Leau et al. (2012)

As metodologias ágeis têm por base as metodologias clássicas, o que indica que existem bastantes semelhanças entre elas. As metodologias ágeis são mais flexíveis, em oposição às clássicas, que são mais resistentes à mudança. Isto acontece nas metodologias clássicas pelo facto do seu planeamento ser realizado no início do projeto e não ao longo do tempo, como é o caso das metodologias ágeis. Esse planeamento é feito em cada iteração, tendo em conta o *feedback* do cliente. Um ponto que também é bastante importante nas metodologias ágeis é a fase de testes. No caso das metodologias ágeis, os testes são realizados em cada iteração, facilitando assim a deteção precoce de erros e possíveis melhorias. Já nas metodologias clássicas a fase de testes é realizada quando o desenvolvimento termina, o que claramente dificulta a identificação de erros (Leau et al., 2012).

2.2 Tecnologias de desenvolvimento de *software*

No que toca às tecnologias de desenvolvimento de *software*, as mesmas podem ser divididas em três categorias: *source code*, *low-code* e *no-code*.

2.2.1 *Source code*

O *source code* é o conjunto de instruções lógicas que um programador escreve quando está a desenvolver uma determinada aplicação. Após escritas, essas instruções são interpretadas ou compiladas para serem executadas diferentes linguagens de programação de alto nível como *Python, Java, C/C++, C##, etc.*, são bons exemplos de tecnologias utilizadas no desenvolvimento de aplicações com recurso a *source code.*

Python é uma linguagem de programação de alto nível, interativa e orientada a objetos utilizada para desenvolvimento de aplicações (Sanner, 1999). Esta linguagem foi

desenvolvida por Guido van Rossuem em 1990 (Sanner, 1999). A sua sintaxe é bastante simples, elegante e de rápido desenvolvimento o que leva a que atualmente seja uma das linguagens *source code*, mais utilizadas (Python.org, 2021).

Por isso, é a linguagem selecionada para a realização desta experiência no contexto deste trabalho.

2.2.2 *Low-code*

O *Low-code* é uma tecnologia de desenvolvimento de *software* que consiste em minimizar a quantidade de codificação manual recorrendo a ferramentas gráficas. Com isto, as funcionalidades são elaboradas de forma mais rápida e com menos esforço por parte do programador, acelerando assim a entrega do *software.* Segue uma arquitetura de quatro camadas: aplicacional; integração de serviço; integração de dados; desenvolvimento (Sahay et al., 2020).

Um exemplo deste tipo de abordagem é a plataforma *Outsystems*, considerada pela Gartner como uma das aplicações líderes de mercado no seu relatório intitulado *The 2020 Gartner Magic Quadrant for Enterprise Low-Code Application Platforms* (Vincent et al., 2020), que se apresenta na Figura 2.2. Por essa razão, é adotada neste trabalho para uma experiência.

Outsystems é uma plataforma *low-code* que permite o desenvolvimento rápido de aplicações *web* e *mobile*, seguindo uma metodologia *Platform-as-a-service* (PaaS) e *Rapid Application Development* (Outsystems, 2021).

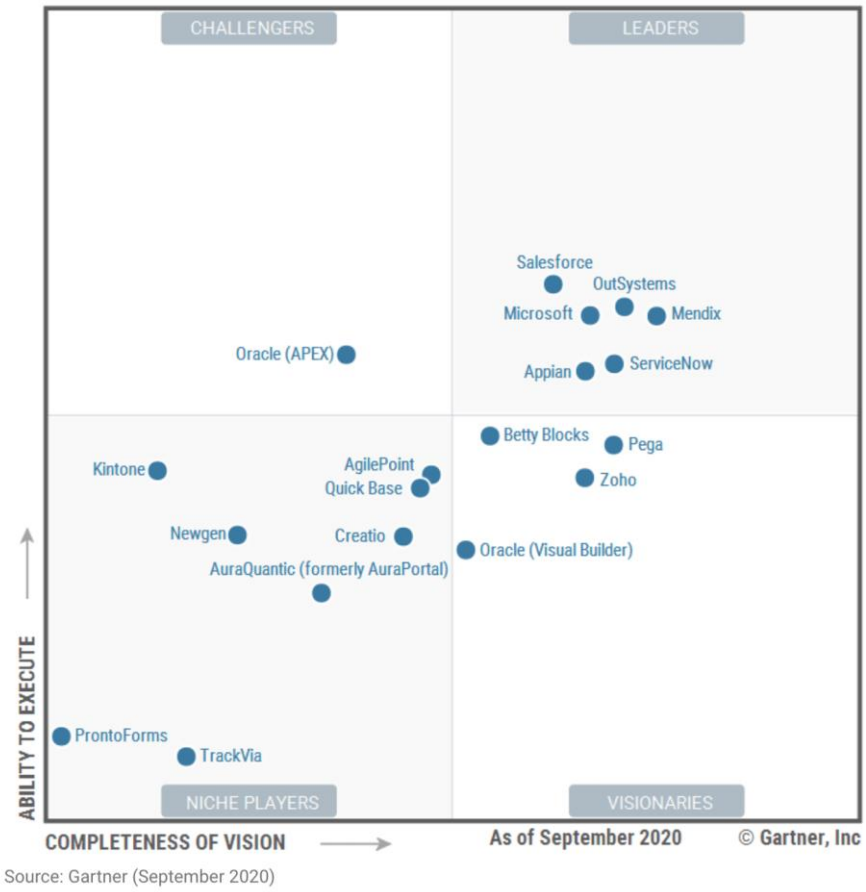


Figura 2.2. The 2020 Gartner Magic Quadrant for Enterprise Low-Code Application Platforms

Fonte: Vincent et al. (2020)

2.2.3 *No-code*

A abordagem *no-code* foi elaborada para pessoas que não têm qualquer formação ao nível de programação, não requerendo codificação manual. Todas as funcionalidades e *design* que o utilizador necessita já estão embutidas na plataforma. Existe, assim, alguma dificuldade em personalizar as aplicações, pois as interfaces e funcionalidades normalmente já estão pré-construídas. As aplicações *Bubble* e *Mendix* são exemplos de plataformas de desenvolvimento *no-code*.

Bubble é uma ferramenta de programação visual disponível na *cloud*. Nesta plataforma é possível a criação de aplicações *web* e *mobile* a partir do browser, ou seja, não existe qualquer aplicação que tenha que ser instalada previamente (Bubble, 2021).

Mendix é também uma plataforma *low-code / no code* para desenvolvimento *web* e *mobile* que não necessita de qualquer escrita de qualquer código (Sahay et al., 2020). Esta plataforma foi desenvolvida com o intuito de acelerar a entrega de aplicações em todo o seu ciclo de vida, desde a sua conceção até à implementação (Mendix, 2021).

2.2.4 Comparação entre *source code, low-code e no-code*

As tecnologias *low-code e no-code* apresentam bastantes semelhanças em comparação a tecnologia *source code*, como se pode verificar na Tabela 2.2.

Tabela 2.2. Comparação entre tecnologias de desenvolvimento source code, low-code e no-code

Características/Tecnologias	***Source code***	***Low-code***	***No-code***
Desenvolvimento	Difícil	Médio*	Fácil
Rapidez de entrega	Demorada	Rápida	Rápida
Conhecimento da linguagem	Elevado	Médio	Nenhum
Limitações/restrições ao desenvolvimento	Poucas	Algumas	Muitas
Custo	Baixo	Elevado	Elevado

*Requer algum conhecimento lógico.

As plataformas *low-code e no-code* fornecem um ambiente gráfico para a criação de aplicações que facilita o desenvolvimento aplicacional, ao contrário da tecnologia *source code* que requer codificação manual. Isto obriga a que quem queira desenvolver uma aplicação *source code* tenha de ter um conhecimento elevado da linguagem de programação que vai utilizar para que possa ser produtivo. O mesmo não acontece com as tecnologias *low-code e no-code*. Nestas tecnologias quase tudo é desenvolvido de forma gráfica, com pouca ou nenhuma programação, permitindo a pessoas sem formação em programação criar aplicações.

Estes fatores levam a que o desenvolvimento de aplicações com recurso às tecnologias *low-code*/*no-code* seja mais rápido, permitindo entregas mais rápidas e maior produtividade (Richardson & Rymer, 2016; Vincent et al., 2020). Um fator contra a utilização destas tecnologias é o custo, quando comparado com o *source code*, tipicamente mais elevado (Sahay et al., 2020).

As plataformas *low-code* têm mecanismos para criar código personalizado, sendo dessa forma possível implementar recursos que não estão disponíveis na plataforma.

2.3 Produtividade no desenvolvimento de s*oftware*

A competitividade entre organizações de desenvolvimento de *software* é bastante grande, como tal a produtividade tem de ser melhorada constantemente. Posto isto, é importante para as organizações adotar métricas de avaliação e definir boas práticas para aprimorar essa produtividade (Oliveira, 2017).

Segundo (Sadowski & Zimmermann, 2019), tendo em conta que não existe um consenso sobre a definição de produtividade, esta pode ser traduzida pela razão entre *output* e o *input* de um determinado projeto:

$$\text{Produtividade} = \frac{Output}{Input}$$

sendo que o *input* é o esforço dedicado ao projeto, o que é fácil de medir (ex.: tempo dedicado). Por outro lado, para o *output* o mesmo não se aplica, pois ainda não foi encontrada uma forma definitiva de o medir, pelo facto de se tratar da quantidade e qualidade do *software* (Sadowski & Zimmermann, 2019).

2.3.1 História da produtividade no desenvolvimento de *software*

A forma de medir a produtividade tem vindo a ser refinada ao longo do tempo e com ela vários métodos foram desenvolvidos para dar uma melhor resposta aos problemas existentes na área.

Wagner e Ruhe (Wagner & Ruhe, 2018) apresentam a história da produtividade no desenvolvimento de software, cujos principais marcos se resumem de seguida.

Entre 1970-1979 surgem as primeiras preocupações, com a medição da produtividade de desenvolvimento de software, tendo os autores *Waltson* e *Felix* sugerido a utilização da métrica de *Software Lines of Code (SLOC)*, (em português, número de linhas de código), como a métrica para estimar o esforço de desenvolvimento das aplicações, pois, segundo estes autores, a complexidade dos programas tem um grande peso na produtividade. No fim da década, em 1979, Albrecht propôs a análise por pontos de função para expressar que o tamanho de um sistema não deve ser medido por linhas de código, mas sim pela quantidade de funcionalidades existentes no mesmo (Sadowski & Zimmermann, 2019).

De seguida, entre 1980-1989 surgem vários trabalhos sobre a análise da produtividade, utilizando pro base o número de linhas de código e os pontos de função que levam à criação do modelo *COnstructive COst MOdel (COCOMO)*, de *Boehm*. Esta abordagem tem como objetivo estimar o esforço de desenvolvimento de um determinado *software* em função do seu tamanho (Singal et al., 2020). Nesta década surgiram também os primeiros trabalhos focados noutros fatores que não o da complexidade da aplicação a desenvolver, como, por exemplo, o ambiente de trabalho dos programadores.

Na década de 1990 surgem vários estudos que se focam na tendência de estudar os fatores sociais associados ao desenvolvimento de software e o seu impacto na produtividade, tais

como, a duração do projeto, o tamanho da equipa, rotatividade de membros de equipa, ambiguidade e conflitos de funções. Outros estudos, como o de *Chatzoglou* e *Macaulay*, identificaram que a experiência dos programadores, o conhecimento e a persistência por parte dos membros da equipa eram importantes, tendo também realçado que os recursos disponíveis, ferramentas, técnicas utilizadas e estilo de gestão são fatores importantes (Wagner & Ruhe, 2018). É nesta década que surge uma nova técnica de estimação do esforço do desenvolvimento de software, baseada nos casos de uso a desenvolver, designada de análise de pontos de caso de uso, proposta em 1993 por *Gustav Karner*, da *Rational Software Corporation* (Carroll, 2005).

Na década seguinte, em 2000, surge uma atualização do modelo COCOMO, denominada COCOMO II (Hjalmarsson et al., 2013). Nesta década surgem também mais trabalhos sobre produtividade em desenvolvimento de *software* focados em outros fatores que não o da complexidade da aplicação a desenvolver.

2.3.2 Fatores que afetam a produtividade do desenvolvimento de *software*

A alta produtividade de um determinado *software* não pode ser definida apenas por um fator, mais sim por um conjunto de fatores, que estão diretamente ligados ao seu desenvolvimento. Dessa forma, é importante que haja um equilíbrio desses fatores, tendo em conta as especificações do projeto (Lavazza et al., 2018).

De seguida são apresentados alguns grupos de fatores diretamente relacionados com a produtividade, fatores relacionados com as pessoas, o produto e a organização.

2.3.2.1 *Fatores relacionados com as pessoas*

Os fatores relacionados com as pessoas podem-se dividir em dois tipos: os individuais, ligados ao individuo, e os de equipa, que se refere a como o indivíduo se relaciona com os restantes indivíduos.

É legitimo afirmar que a experiência de um programador pode afetar diretamente a eficiência na realização de uma determinada tarefa. Segundo (Sadowski & Zimmermann, 2019), a facilidade de compreender o que determinado programa faz, tem um peso importante no tempo de finalização de determinada tarefa. As aptidões/competências de um indivíduo podem também afetar a produtividade do desenvolvimento de *software* pelo facto deste não ser capaz ou não ter o conhecimento, por exemplo, acerca de uma

determinada linguagem de programação. Outro fator que afeta a produtividade é a motivação, sendo importante que esta seja incentivada numa organização.

A comunicação entre as pessoas da mesma equipa é bastante importante. Quanto maior for a equipa mais difícil será essa interação, pois vão existir mais "linhas" de comunicação entre eles (Oliveira, 2017). O tamanho da equipa difere consoante o projeto, podendo, por exemplo, existir projetos com equipas de cinco pessoas e projetos de cinquenta pessoas. A Figura 2.3 demonstra as possibilidades na comunicação entre dois programadores e quatro programadores, tendo em conta que todos comunicam entre si.

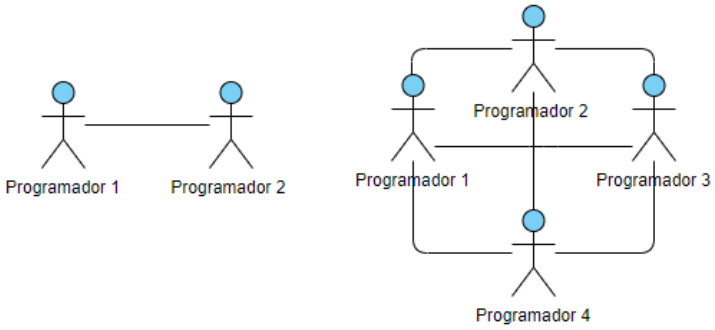


Figura 2.3. Comunicação entre elementos de uma equipa

Fonte: Oliveira (2017)

Como se pode observar na Figura 2.3 só existe uma "linha" de comunicação entre dois programadores, o que indica que a sua comunicação é simples e clara. No exemplo com quatro programadores, as "linhas" de comunicação aumentam, chegando a seis caminhos distintos, o que pode indicar que a sua comunicação não seja tão clara como no exemplo anterior. Este exemplo serve para ilustrar que, quanto maior for a equipa, neste caso de programadores, maior é o tempo gasto na passagem de informação, e por sua vez, maior será também a possível ocorrência de erros nessa comunicação. Uma boa prática para minimizar este tipo de problemas em grandes projetos é a criação de documentação, para que todas as pessoas tenham acesso à informação da mesma forma (Oliveira, 2017). A saída de membros da equipa, denominado por *Turnover,* é também um fator que pode afetar a produtividade. Equipas coesas e com pouca rotatividade normalmente apresentam uma maior produtividade (Canedo & Santos, 2019).

2.3.2.2 *Fatores relacionados com o produto*

As características definidas pelo cliente para um determinado *software* têm influência na produtividade. Existem bastantes fatores relacionados com o produto/*software*, como a

área de negócio na qual vai ser implementado (por exemplo a banca ou seguros). A complexidade da aplicação pode também afetar a produtividade, por exemplo, pelo facto de ser difícil de compreender e, dessa forma, ser mal projetada por parte do gestor de protejo (Oliveira, 2017).

A linguagem ou linguagens de programação utilizadas podem também trazer implicações no processo de desenvolvimento, devido a limitações existente nessa linguagem. Quanto maior for o nível de abstração da linguagem utilizada, melhor vai ser a produtividade (Canedo & Santos, 2019).

Existem correlações negativas relativamente à duração de projetos e produtividade, o que significa que quanto maior for a duração do projeto, menor será a produtividade. O mesmo acontece com o tamanho de *software*, que é uma das variáveis centrais no cálculo da produtividade e representa a quantidade de *software* gerado. O tamanho de *software* é diretamente proporcional à complexidade desse mesmo *software* e é por esse motivo que o tamanho tem uma relação negativa com a produtividade (Canedo & Santos, 2019).

2.3.3 Fatores relacionados com a organização

Quanto aos fatores do ponto de vista da organização, estes podem afetar direta ou indiretamente aspetos relacionados com os colaboradores, tendo em conta atitudes ou práticas implementadas (Wagner & Ruhe, 2018).

A organização é responsável pelo ambiente de trabalho, que influencia diretamente a produtividade dos seus colaboradores. Para que essa influência seja positiva, é importante ter um local de trabalho saudável e que vá de encontro às necessidades dos colaboradores. É importante, também, para os colaboradores existirem certificações em outras áreas, ou até mesmo na mesma área, para poderem evoluir o seu conhecimento e em paralelo trazer mais valor à sua organização. Outro aspeto que é importante frisar é a possibilidade de cada colaborador poder progredir na carreia, assim como ter outras responsabilidades, ou diferentes desafios, o que levará a um aumento considerável na sua produtividade. As recompensas e sistemas de mérito são também um ponto positivo na produtividade da organização (Canedo & Santos, 2019).

2.4 Estimação do esforço no desenvolvimento de *software*

A indústria de *software* está cada vez mais ligada à eficiência. Nesse contexto, a estimativa de esforço tem um papel fulcral no sucesso do desenvolvimento de *software*. De um modo geral, as técnicas de estimativa de esforço têm sido usadas para planear e analisar os requisitos de um determinado projeto, com o objetivo de este ser sustentável. Com o passar dos anos, muitos métodos foram apresentados com o objetivo de serem mais precisos e eficientes (Chowdary & Krishna, 2016).

2.4.1 Número de Linhas de Código

A primeira métrica utilizada para determinar o tamanho do *software* foi o número de linhas de código. Significando assim, que o tamanho do *software* desenvolvido seria igual ao total das suas linhas de código. Esta métrica é bastante discutível por vários motivos. Um desses motivos é o facto de existirem linhas que não deveriam ser contabilizadas da mesma forma, como as de comentário, declaração de variáveis e linhas em branco. Existem também problemas relacionados com as linguagens utilizadas no desenvolvimento do *software.* A título de exemplo, imagine-se o desenvolvimento de duas aplicações de *software* exatamente iguais, uma deles desenvolvido por uma equipa em *Assembly* e outra por outra equipa em *Python*. O tamanho do *software* medido em *Assembly* é, hipoteticamente, de dez mil linhas e em *Python* é, também, hipoteticamente, de três mil linhas. Ambas as equipas demoraram cinco meses a desenvolver a aplicação mas, pelo facto de a aplicação desenvolvida em *Assembly* ter mais linhas de código, a equipa que a desenvolveu é considerada como sendo mais produtiva, pois entregou mais linhas de código no mesmo tempo que a equipa que desenvolveu a mesma aplicação em *Python* (Pressman, 2009).

2.4.2 Análise de Pontos de Função

Na década de setenta, *Allan Albrecht* tinha como principal objetivo da sua pesquisa minimizar os problemas relacionados com a medição do tamanho de *software*, que na altura tinha como base a contagem das linhas de código (Lokan, 2005). Dessa forma, em 1979, propôs uma nova técnica denominada Análise de Pontos de Função, em inglês *Function Point Analysis*, que consiste na medição de *software* tendo em conta as funcionalidades solicitadas pelo utilizador. A sua unidade de medida são as funcionalidades. Dessa forma, esta medição é independente da tecnologia utilizada na

construção do *software*, podendo assim ser utilizada em diferentes tecnologias e linguagens de programação. O processo de medição tem como base os requisitos funcionais do utilizador. É importante frisar que esta análise não mede diretamente o esforço, a produtividade ou o custo, mas sim o tamanho funcional do *software* (Ferrucci et al., 2016). Contudo, o tamanho funcional, correlacionando com outras variáveis pode determinar a produtividade, esforço ou custo de um determinado *software*.

Esta abordagem sofreu algumas alterações no início da década de oitenta, pelo próprio *Allan Albrecht.* Algum tempo depois, mais precisamente em 1986, foi fundada a *International Function Point Users Group* (IFPUG) (em português Grupo Internacional de Utilizadores de Pontos de Função), uma organização sem fins lucrativos que é responsável pela gestão do Manual de Práticas de Contagem, onde está descrito todo o procedimento de medição de Pontos de Função (Ifpug, 2010).

O Manual de Práticas de Contagem (Ifpug, 2010) propõe o Método de Pontos de Função para a realização da análise de pontos de função com os seguintes passos:

1. Determinar o Tipo de Contagem, ou seja, determinar o que vai ser medido, existindo três tipos:
    a. Contagem de pontos de função do projeto de desenvolvimento;
    b. Contagem de pontos de função do projeto de melhoria;
    c. Contagem de pontos da função de aplicação.
2. Identificar o âmbito e fronteira da aplicação, onde se define o âmbito da aplicação, ou seja, o conjunto de requisitos funcionais do utilizador que vão ser incluídos na contagem e a fronteira, tendo em conta o *software* que vai ser analisado e os utilizadores do sistema ou outro sistema externo.
3. Determinar a contagem de pontos de função não ajustados (pontos de função brutos). Esta contagem divide-se em duas categorias: das funções tipo dados relacionadas com o manuseamento de dados da aplicação, internos ou externos, e com as funções tipo transação, que permitem a interação coma aplicação e que se dividem em três tipos: entradas externas, saídas externas e consultas externas.
4. Determinar o valor do fator de ajuste (VAF) definido pelo IPUG pela ponderação de 14 características gerais da aplicação, que na maior parte dos casos faz variar o valor anterior entre -10% e +10%, pelo que se tornou opcional a partir de 2002 (Vazquez et al., 2013). A técnica de análise por Pontos de Função considera que outros fatores afetam o tamanho funcional de uma aplicação.

2.4.3 Análise de Pontos de Caso de Uso

A linguagem *Unified Modeling Language* (UML), é utilizada para modelar e especificar o *software* a desenvolver (Arlow & Neustadt, 2002), através da utilização de modelos. Segundo (Rambe et al., 2020), "A UML é uma linguagem padrão amplamente usada para definir requisitos, analisar, desenhar e especificar a arquitetura de *software* (…)", utilizando diversos diagramas.

Tendo em conta a análise apresentada nesta secção, é importante abordar um diagrama específico da linguagem UML, o diagrama de casos de uso, que representa as funcionalidades disponibilizada por uma aplicação/sistema aos seus atores. A Figura 2.4 apresenta um hipotético diagrama de casos de uso, com os seguintes elementos gráficos: fronteira da aplicação/sistema a desenvolver, atores, casos de uso e relacionamentos.

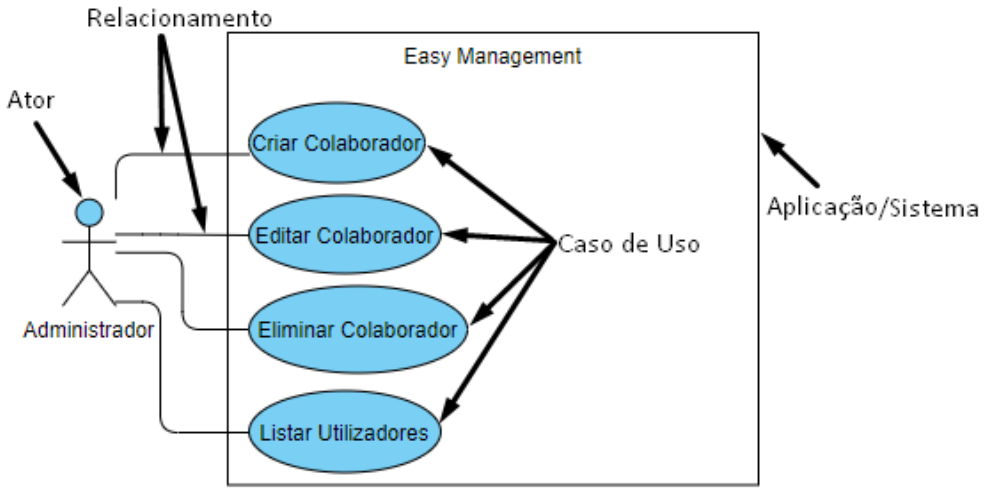


Figura 2.4. Exemplo de Diagrama de Caso de Uso

No exemplo da Figura 2.4 só existe um ator no sistema o "Administrador". O ator pode ser uma pessoa, o tempo, ou uma máquina ou uma aplicação, que por exemplo, que interage com o sistema para solicitar ou disponibilizar um tipo de serviço. Quanto aos casos de uso, estes representam as ações que se podem realizar no sistema, neste caso na aplicação "*Easy Management*", cuja fronteira se encontra representada por um retângulo (ver Figura 2.4). O nome do caso de uso deve ser inicializado sempre com um verbo, por exemplo "Criar", "Editar" ou "Eliminar", para reforçar a ideia de que é uma ação a ser realizada pelo ator (externo ao sistema). Para indicar a relação entre o ator e o caso de uso, é efetuada uma ligação entre ambos, do tipo "Associação", que indica que, neste caso, o "Administrador" pode "Criar Colaborador", "Editar Colaborador", "Eliminar Utilizador" e "Listar Utilizadores".

Em 1993, *Gustav Karner* propôs uma abordagem que tem como objetivo principal permitir estimar o tamanho e a complexidade de um determinado projeto a partir dos seus requisitos, descritos num diagrama de casos de uso, designada de análise de Pontos de Caso de Uso. Os pontos de caso de uso estão diretamente relacionados com a complexidade de natureza funcional, técnica e ambiental do projeto. Dessa forma, é necessário ter em conta alguns fatores como (Sousa et al., 2012):

1. Quantidade e complexidade dos atores presentes no sistema;
2. Quantidade e complexidade dos casos de uso presentes no sistema;
3. Alguns requisitos não funcionais que possam impactar o desenvolvimento do projeto;
4. O impacto que poderá ter o ambiente de desenvolvimento do projeto.

O Método Pontos de Caso de Uso é composto por duas etapas (Nageswaran, 2001; Ochodek et al., 2010; Sousa et al., 2012):

1. Cálculo *Unadjusted Use Case Points* (UUCP) a partir das variáveis *Unadjusted Actor Weight* (UAW) e *Unadjusted Use Case Weight* (UUCW), relativas à complexidade dos atores e dos casos de uso respetivamente;
2. Ajuste do UUCP, tendo em conta um conjunto de fatores técnicos e ambientais representados pelas varáveis *Technical Complexity Factor (TCF) e Environmental Factor (EF)*. A combinação da variável UUCP com as variáveis TCF e EF resulta no número efetivo de *Use Case Points* (UCP) do projeto (A. Sousa et al., 2012). A Figura 2.5 resume estes passos:

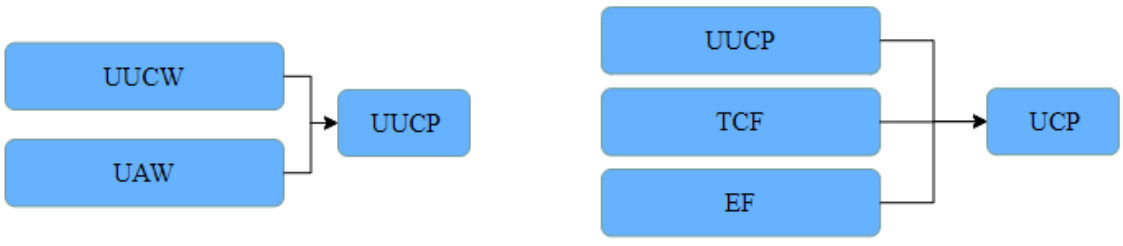


Figura 2.5. Cálculo do número efetivo de Use Case Points (UCP)

Para determinar o *UAW* todos os atores que interagem com o sistema têm de ser classificados consoante a sua complexidade. Nem todos os atores têm as mesmas características, podendo ser divididos em três categorias com pesos diferentes: simples (peso 1), se for um ator que interage com o sistema através de uma *Application Programming Interface* (API); mediano (peso 2), se interage com o sistema com o sistema através da linha de comandos ou através de algum tipo de protocolo, por exemplo *Transmission Control Protocol* (TCP) ou *File Transfer Protocol* (FTP); ou complexo (peso 3), se interage com o sistema através de uma interface gráfica.

O *UUCW* é determinado a partir de todos os casos de uso existentes no sistema, sendo estes também classificados consoante a sua complexidade, dividindo-se assim em: simples (peso 5), medianos (peso 10) ou complexos (peso 15). Quando um caso de uso é classificado como simples, isto indica que tem três ou menos passos e só interage com uma entidade da base de dados. Quanto à categoria mediano, significa que esse caso de uso tem quatro a sete passos e interage com pelo menos duas entidades da base de dados. Relativamente à categoria complexo, são todos aqueles casos de uso que tenham mais que sete passos e interajam com pelo menos duas três entidades da base de dados (Clemmons, 2006).

Com o UAW e o UUCW calculados, o valor do UUCP é determinado a partir da soma destas duas variáveis, sendo esta uma aproximação desajustada pelo facto de não ter em conta fatores técnicos nem ambienteis do projeto.

$$\mathrm{UUCP} = \mathrm{UAW} + \mathrm{UUCW}$$

Como referido anteriormente o UUCP deve ser ajustado tendo em conta o TCF e o TE. Para isso verifica-se a influência de cada um dos fatores apresentados na tabela da Figura 2.6., atribuindo a cada um dos fatores um peso entre 0 e 5 consoante a influência do mesmo no projeto. Depois somam-se todos os fatores, multiplicando os mesmos pelos pesos apresentados na tabela da Figura 2.6., obtendo dessa forma as variáveis *TFactor* e *Efactor* que serão utilizadas nas fórmulas abaixo para calcular o TCF e o TE.

$$\mathrm{TCF} = 0.6 + (0.01 \times TFactor)$$

$$\mathrm{EF} = 1.4 + (-0.03 \times EFactor\ )$$

Factor	Description	Weight
Technical complexity factors		
T1	Distributed system	2
T2	Performance	1
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Security features	1
T12	Access for third parties	1
T13	Special training required	1
Environmental factors		
F1	Familiarity with the standard process	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2
F7	Part-time workers	−1
F8	Difficult programming language	−1

Figura 2.6 .Peso dos fatores técnicos e de ambiente

Fonte: Ochodek et al. (2010)

Com as variáveis anteriormente definidas, nomeadamente UUCP, TCF e EF o valor efetivo do *Use Case Points (UCP)* é representado pela seguinte equação:

$$\mathrm{UCP} = \mathrm{UUCP} \times \mathrm{TCF} \times \mathrm{EF}$$

Finalmente, este valor é multiplicado pelo *Producitvity Factor* (PF), que representa o número de horas necessário para o desenvolvimento de cada UCP, com o valor de 20 horas para equipas que nunca tenham desenvolvido projetos de desenvolvimento de *software* (Clemmons, 2006; Ochodek et al., 2010). Este valor deve ser atualizado para os próximos projetos a desenvolver, com base nos projetos de desenvolvimento de software que foram desenvolvidos (Ochodek et al., 2010).

$$Esforço\ Total = UCP \times PF$$

2.4.4 Análise comparativa dos três métodos de estimação do esforço abordados

A métrica de linhas de código é utilizada para medir o tamanho de um determinado *software* a partir do seu número de linhas e com esse valor estimar a produtividade no desenvolvimento de *software*. Tendo em conta que existem, nos dias de hoje, tecnologias

que não contêm diretamente linhas de código, como é o caso das plataformas *low-code e no-code*, esta métrica torna-se inviável em vários contextos.

Em contrapartida, análises como pontos de função e pontos de caso de uso, permitem utilizar diferentes abordagens para estimar o esforço de desenvolvimento de software/produtividade. A análise de pontos de função tem como unidade de medida as funcionalidades solicitadas pelo utilizador, sendo assim independente da tecnologia. O mesmo acontece com a análise de pontos de função, em que a sua unidade de medida são os casos de uso.

3 METODOLOGIA

Neste capítulo apresenta-se a metodologia utilizada neste trabalho, a metodologia de experiência laboratorial. Após uma definição sucinta da mesma apresenta-se a descrição da experiência a realizar neste trabalho e os respetivos protocolos, cuja versão completa pode ser consultada nos apêndices 1 e 2.

3.1 Metodologia de Experiência Laboratorial

A investigação científica é um processo que tem como objetivo estudar de forma cuidadosa e critica um determinado objeto ou certos fenómenos específicos extraídos da realidade, através da aplicação de métodos científicos (Alvarenga, 2012). A abordagem metodológica que se irá utilizar neste trabalho de projeto será o da Experiência Laboratorial, que consiste na realização de experiências em ambiente controlado (Balijepally et al., 2009), em que os participantes têm que seguir um protocolo previamente definido, para que todos os fatores sejam rigorosamente cumpridos.

O método da experiência laboratorial envolve a manipulação de variáveis para estabelecer relações de causa e efeito. Uma experiência é uma investigação na qual uma hipótese é cientificamente testada, em que se avaliam como as variáveis independentes são manipuladas e as dependentes são medidas. As experiências devem ser objetivas, de modo que os pontos de vista do investigador não influenciem os resultados do estudo. Tal é positivo, pois torna os dados mais válidos, e menos tendenciosos (McLeod, 2012).

Uma experiência laboratorial é uma experiência realizada em condições altamente controladas, onde são possíveis medições precisas. O investigador decide onde a experiência terá lugar, a que horas, com que participantes, em que circunstâncias e utilizando um protocolo (McLeod, 2012).

O ambiente artificial e controlado da experiência laboratorial torna possível a medição exata dos tempos de execução das tarefas, algo que seria impossível de realizar noutro tipo de estudos como, por exemplo, experiências de campo, em que não é possível controlar estímulos externos que condicionem a realização das tarefas (Wenz, 2021).

3.2 Descrição da experiência laboratorial

Na experiência realizada, a variável em estudo foi a produtividade do desenvolvimento e manutenção de uma aplicação com recurso a diferentes tecnologias, medida com base no

tempo que esse desenvolvimento demorou, tendo por referência uma especificação em casos de uso.

A experiência foi efetuada mediante um plano de atividades previamente definido, onde são descritas todas as etapas a ter em consideração.

Na experiência realizada, o que se pretendeu estudar foi a produtividade no desenvolvimento de *software*, pelo que a questão de investigação subjacente é:

- Qual a diferença de produtividade das tecnologias de desenvolvimento de *software*, *source code* e *low-code*?

Para responder a esta questão de investigação realizou-se uma experiência laboratorial, dividida em duas fases (que também pode ser vista como duas experiências separadas), utilizando. As tecnologias de desenvolvimento de software, *Python (source code)* e *Outsystems (low-code)*:

1. Experiência de desenvolvimento de *software*, em que se propôs o desenvolvimento de uma aplicação com base num conjunto de casos de uso;
2. Experiência de manutenção de software, em que se propôs a alteração/desenvolvimento de uma aplicação com base em novos casos de uso.

Para a realização das experiências foram convidados dois programadores com vasta experiência no desenvolvimento de aplicações. No caso da experiência com *Outsystems*, o programador tem 2 anos em desenvolvimento com *Outsystems*. No caso da experiência *Django/Python* o programador tem 5 anos em desenvolvimento com *Django/Python*. Ambos são do género masculino.

De seguida apresentam-se os protocolos das experiências realizadas cuja versão completa pode ser consultada nos apêndices 1 e 2 do presente documento.

3.2.1 Fase I – Desenvolvimento de s*oftware*

A aplicação informática a desenvolver, denominada *Easy Management*, tem por objetivo suportar a gestão das atividades de um projeto, desde a sua criação até à finalização.

Esta aplicação inclui a gestão de Utilizadores, Projetos e Tarefas. A aplicação foi projetada para funcionar em *Web* e deverá ser compatível com diferentes *browsers*.

A aplicação contempla três perfis distintos de utilizadores, que têm acesso a diferentes funcionalidades na aplicação:

- O **Administrador** tem, somente, a possibilidade de gerir os utilizadores da plataforma;
- O **Gestor de Projetos**, pode criar Projetos e as respetivas Tarefas. Com este perfil, também é possível listar os (próprios) Projetos e Tarefas, sendo ainda possível editá-los e arquivá-los;
- No que respeita ao **Colaborador**, este tem a possibilidade de consultar e editar as suas Tarefas, as quais são previamente atribuídas pelo **Gestor de Projetos**. O **Colaborador** tem, também, a possibilidade de visualizar os Projetos que estão associados às suas Tarefas.

O sistema é composto por três módulos que agrupam as funcionalidades do sistema, encontrando-se identificados na Figura 3.1: Módulo Autenticação (P1); Módulo Gestão de Utilizadores (P2); Módulo Gestão de Projetos e Tarefas (P3).

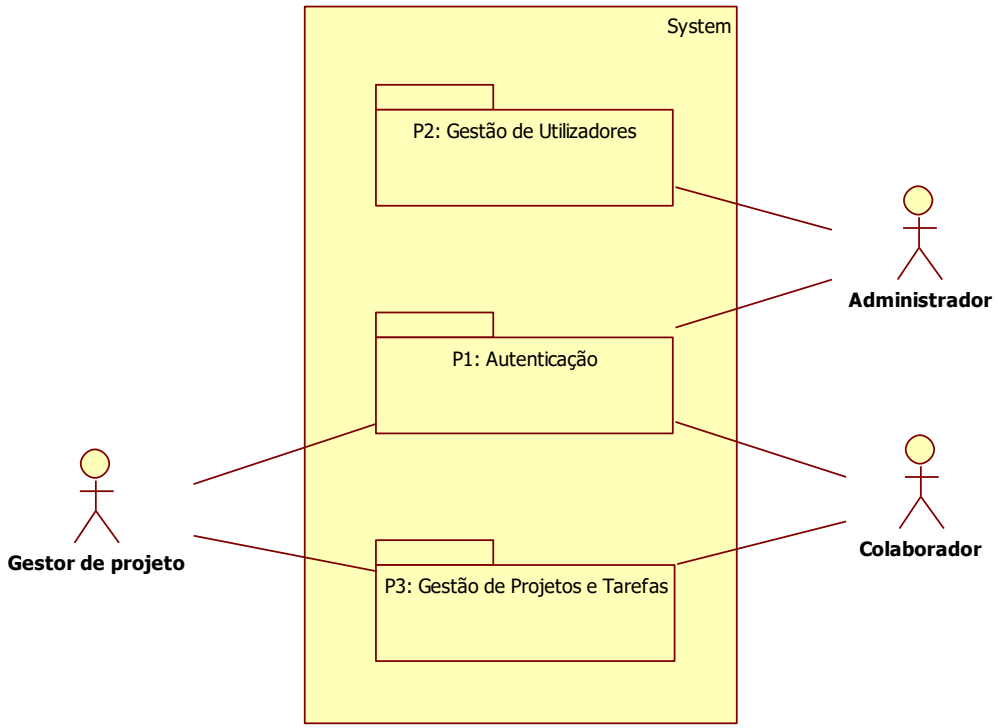


Figura 3.1. Módulos do Sistema

Relativamente ao Módulo de Autenticação, os seus casos de uso estão representados na Figura 3.2.

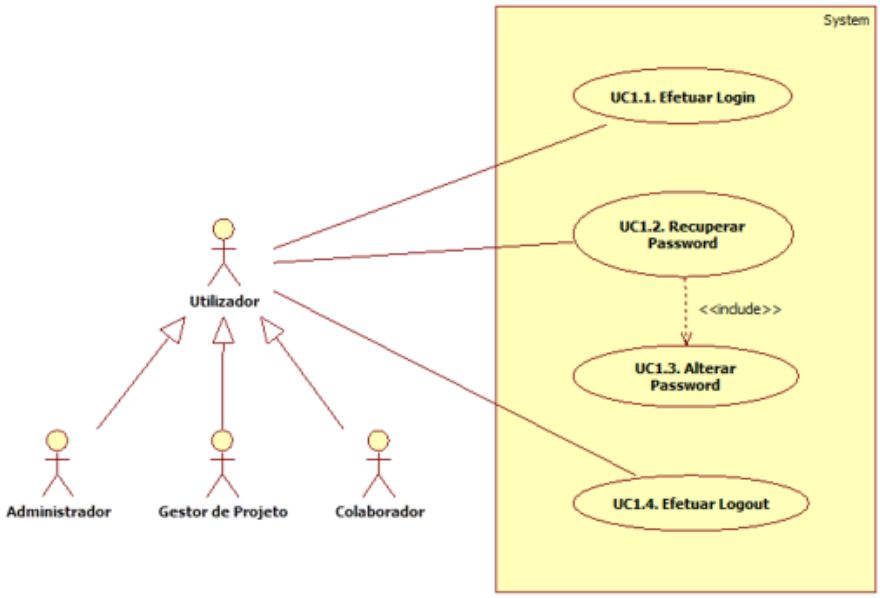


Figura 3.2. Diagrama de Casos de Uso (Autenticação)

Especificação de cada Caso de Uso:

- **UC1.1. Efetuar *Login*** - Este caso de uso permite que um utilizador se autentique na aplicação.
- **UC1.2. Recuperar *Password*** - Este caso de uso permite ao utilizador recuperar a sua *password* de acesso ao sistema.
- **UC1.3. Alterar *Password*** - Este caso de uso permite a um utilizador alterar a sua *password*.
- **UC1.4. Efetuar *Logout*** - Este caso de uso permite ao utilizador encerrar a sua sessão na aplicação.

Quanto ao Módulo de Gestão de Utilizadores, os seus casos de uso estão representados na Figura 3.3.

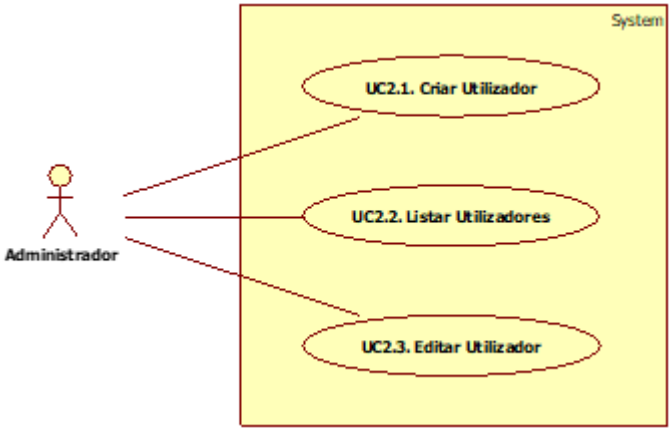


Figura 3.3. Diagrama de Casos de Uso (Gestão de Utilizadores)

Especificação de cada Caso de Uso:

- **UC2.1. Criar Utilizador -** Este caso de uso permite criar um utilizador.
- **UC2.2. Listar Utilizadores -** Este caso de uso permite a visualização de todos os utilizadores registados na aplicação
- **UC2.3. Editar Utilizador -** Este caso de uso permite editar os dados de um utilizador.

Por fim, quanto ao Módulo Gestão de Projetos e Tarefas, os seus casos de uso estão representados na Figura 3.4.

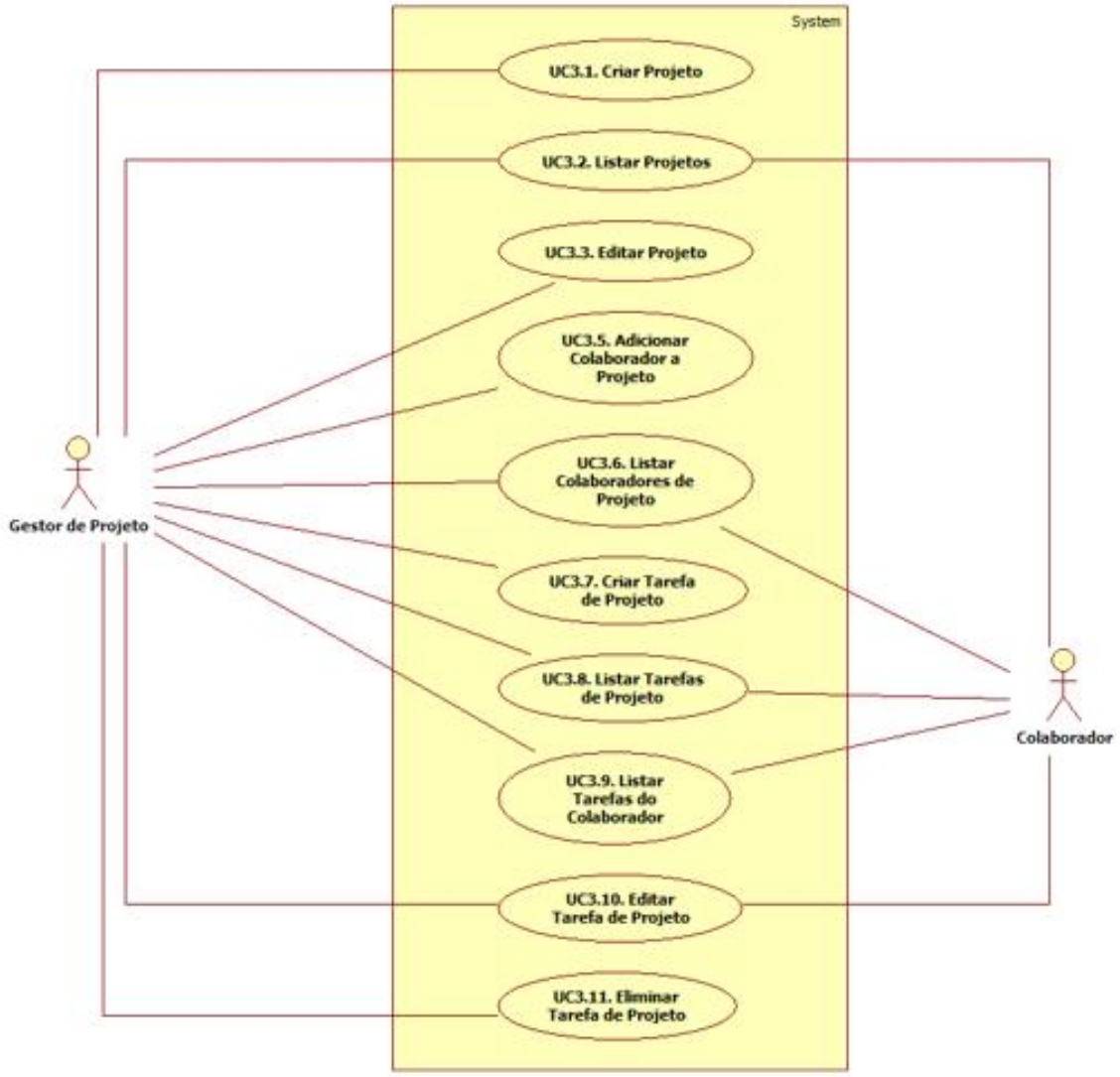


Figura 3.4. Diagrama de Casos de Uso (Gestão de Projetos e Tarefas)

Especificação de cada Caso de Uso:

- **UC3.1. Criar Projeto -** Este caso de uso permite criar um projeto.
- **UC3.2. Listar Projetos -** Este caso de uso permite listar e pesquisar os projetos existentes na aplicação associados ao utilizador.
- **UC3.3. Editar Projeto -** Este caso de uso permite alterar os dados de um projeto.
- **UC3.5. Adicionar Colaborador ao Projeto -** Este caso de uso só está disponível para o Gestor do Projeto e permite adicionar colaboradores ao projeto.
- **UC3.6. Listar Colaboradores do Projeto -** Este caso de uso só está disponível para o Gestor do Projeto e permite a visualização dos colaboradores associados ao projeto.
- **UC3.7. Criar Tarefa do Projeto -** Este caso de uso permite criar uma tarefa associada a um determinado projeto.
- **UC3.8. Listar Tarefas do Projeto -** Este caso de uso permite listar as tarefas existentes na aplicação associadas ao projeto.
- **UC3.9. Listar Tarefas do Colaborador -** Este caso de uso permite listar e filtrar as tarefas existentes na aplicação associadas ao utilizador em sessão.
- **UC3.10. Editar Tarefa do Projeto -** Este caso de uso permite editar uma tarefa associada a um determinado projeto.
- **UC3.11. Eliminar Tarefa do Projeto -** Este caso de uso permite eliminar uma tarefa.

No "Protocolo para Realização da Experiência Laboratorial - Fase I" presente no *Apêndice 1,* existe uma descrição mais detalhada de todos os casos de uso.

Para estimar o esforço, utilizou-se o método de Análise de Pontos de Caso de Uso, apresentado no capítulo anterior. Com isto, foi obtido um UUCP a partir da soma de duas variáveis, nomeadamente UAW e UUCW, calculadas previamente e indicadas na Figura 3.5.

Unadjusted Actor Weight (UAW)

Atores	Peso
Administrador	3
Gestor de Projetos	3
Colaborador	3
Total	**9**

Unadjusted Use Case Weight (UUCW)

Casos de Uso	Peso
UC1.1. Efetuar Login	5
UC1.2.Recuperar Password	5
UC1.3. Alterar Password	5
UC1.4. Efetuar Logout	5
UC2.1. Criar Utilizador	5
UC2.2. Listar Utilizadores	5
UC2.3. Editar Utilizador	5
UC3.1. Criar Projeto	5
UC3.2. Listar Projetos	5
UC3.3. Editar Projeto	5
UC3.5. Adicionar Colaborador ao Projeto	10
UC3.6. Listar Colaboradores do Projeto	10
UC3.7. Criar Tarefa do Projeto	15
UC3.8. Listar Tarefas do Projeto	10
UC3.9. Listar Tarefas do Colaborador	10
UC3.10. Editar Tarefa do Projeto	10
UC3.11. Eliminar Tarefa do Projeto	10
Total	**125**

Figura 3.5. Cálculo de UAW e UUCW

Dessa forma, o valor da variável UUCP é calculado da seguinte forma:

$$\text{UUCP} = \text{UAW} + \text{UUCW} = 125 + 9 = 134$$

Dado que se considerou o valor de 1 para EF para TCF o valor do UCP = UUCP, ou seja, 134.

Com base no protocolo, foi efetuada uma lista de tarefas para cada caso de uso, onde é indicado o caso de uso, o *mockup* e as tarefas a realizar. Essa lista pode ser observada no *Apêndice 3*.

3.2.2 Fase II – Manutenção de s*oftware*

A segunda fase do projeto destina-se a testar a produtividade das duas tecnologias na manutenção das aplicações existentes, seja para alterar ou criar novas funcionalidades. Assim, nesta fase, decidiu-se alterar uma funcionalidade já existente e adicionar uma nova funcionalidade como se descreve de seguida:

- O campo "Data de Fim" da tabela "Tarefa", anteriormente definido (Protocolo I), deve ser alterado para "Data de Fim Prevista" (*DeadLine*). Também deve ser adicionado a essa tabela um novo atributo designado por "Data de Fim Efetiva". Deve, ainda, existir a possibilidade de adicionar mais do que um colaborador a cada tarefa, podendo assim a tarefa ser realizada por vários colaboradores.
- Focando os detalhes do projeto, deve ser criado um novo separador, denominado "Indicadores", possibilitando ao gestor do projeto obter uma visão geral das tarefas. Cada colaborador deverá, também, ter disponível no menu principal um separador denominado "Desempenho", que permita obter uma visão geral das tarefas em que intervém.

Nesta nova versão não há alteração no Módulo Autenticação nem no Módulo Gestão de Utilizadores. O Módulo de Gestão de Projetos e Tarefas é o único que vai sofrer alteração nesta nova versão.

A Figura 3.6 apresenta os casos de uso refentes ao módulo Gestão de Projetos e Tarefas.

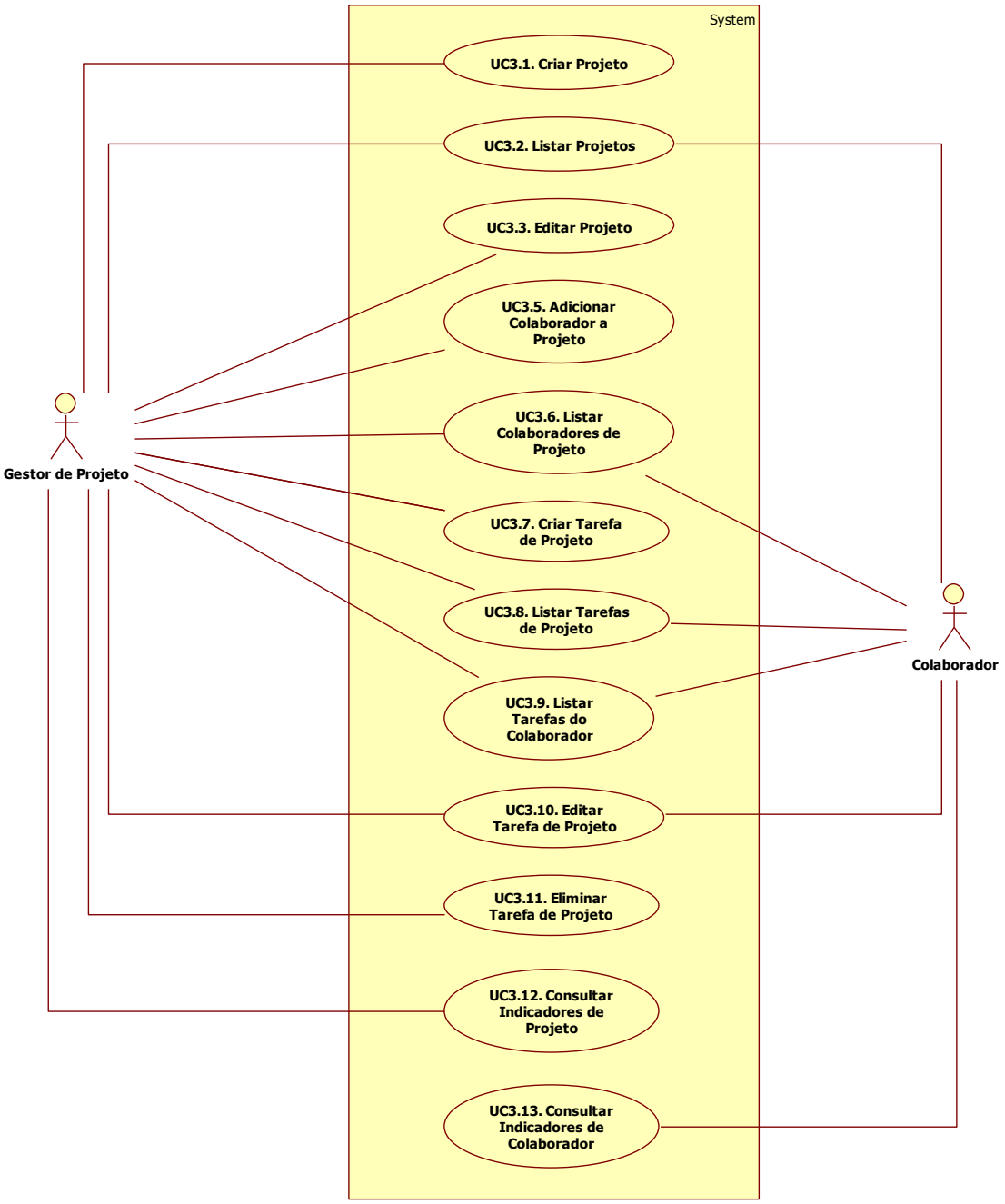


Figura 3.6. Diagrama de Casos de Uso do módulo de Gestão de Projetos e Tarefas da Fase II

Especificação dos novos Caso de Uso:

- **UC3.12 - Consultar Indicadores do Projeto** - Este caso de uso permite consultar os indicadores referentes ao projeto.
- **UC3.13 - Consultar Indicadores do Colaborador** - Este caso de uso permite consultar os indicadores referentes ao colaborador.

No "Protocolo para Realização da Experiência Laboratorial - Fase II" presente no *Apêndice 2*, existe uma análise mais detalhada em relação aos dois casos de uso identificados acima, como também das alterações efetuadas em casos de uso já existentes.

Relativamente à estimativa do esforço, foi utilizado o mesmo método que na primeira fase de desenvolvimento, obtendo assim o seguinte valor:

$$\mathrm{UUCP} = \mathrm{UAW} + \mathrm{UUCW} = 40 + 9 = 49$$

Com base neste protocolo foi elaborada a lista de tarefas para cada caso de uso (*Apêndice 4*), onde é indicado o caso de uso, o *mockup* e as tarefas a realizar.

4 RESULTADOS

Neste capítulo apresentam-se as aplicações desenvolvidas nos dois tipos de tecnologia, *source code* e *low-code*, de acordo com o protocolado no capítulo anterior, e é feita uma análise aos tempos obtidos em ambos os desenvolvimentos de forma a responder à questão de investigação. Por fim, comparam-se os resultados obtidos com os reportados na literatura.

4.1 Apresentação das soluções desenvolvidas

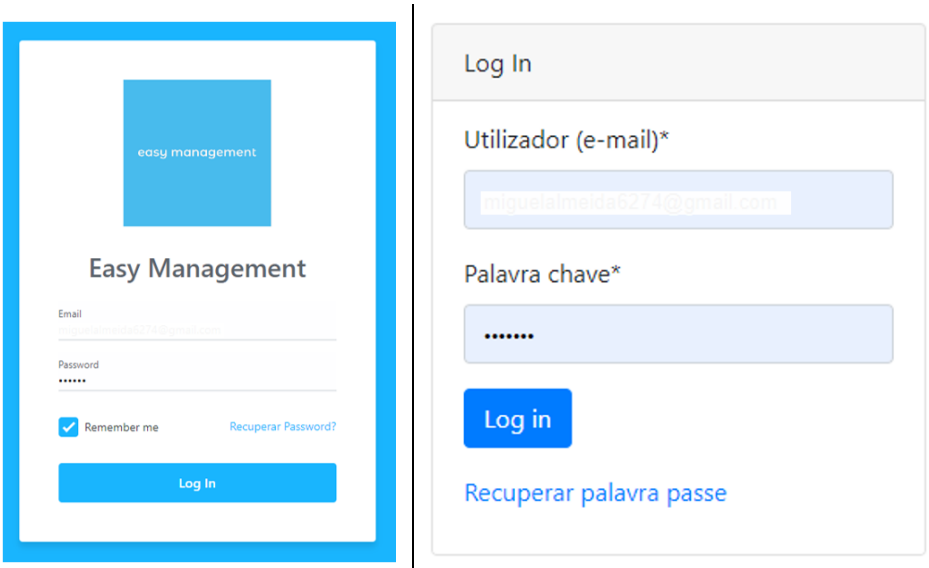
4.1.1 Fase I – Desenvolvimento de s*oftware*

A primeira fase vai de encontro ao que foi definido no "Protocolo para Realização da Experiência Laboratorial" presente no Apêndice 1, em que estão definidas, com detalhe, as funcionalidades a desenvolver. Nas próximas secções são apresentados os resultados da implementação em *Outsystems* e *Django/Python*.

4.1.1.1 *Módulo Autenticação*

1) UC1.1. Efetuar Login

Neste formulário, o utilizador deverá inserir as suas credenciais de acesso.

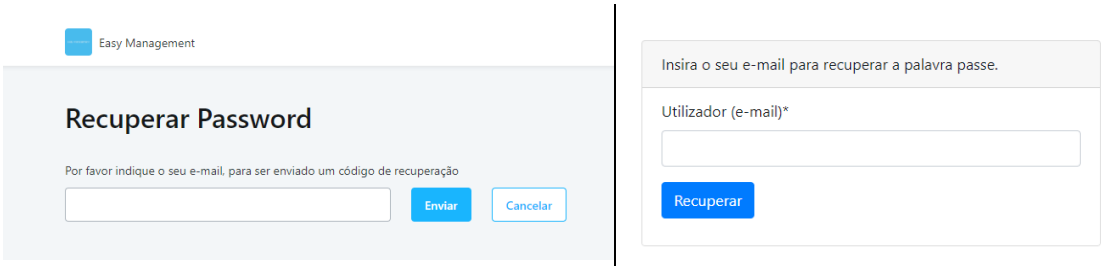


a) Outsystems *b) Django/Python*

Figura 4.1. UC1.1. Efetuar Login (Outsystems vs. Django/Python)

2) UC1.2. Recuperar Password

Caso o utilizador se esqueça das credenciais, terá a possibilidade de recuperar a sua *password*. Para isso, terá de clicar no link "Recuperar Password" que se encontra na mesma página do *Login* (ver Figura 4.1) e, em seguida, irá ser encaminhado para o formulário apresentado na Figura 4.2. Após inserir o seu endereço de e-mail e submeter o formulário, irá receber um e-mail com instruções e um link para efetuar a recuperação da *password*.

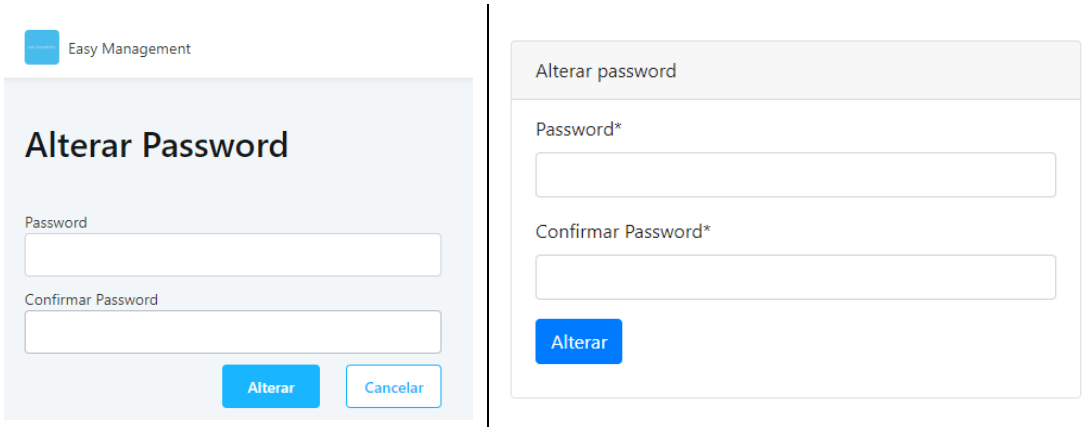


a) Outsystems *b) Django/Python*

Figura 4.2. UC1.2. Recuperar Password (Outsystems vs. Django/Python)

3) UC1.3. Alterar *Password*

Caso o utilizador pretenda alterar a sua *password*, terá de selecionar a opção "Recuperar *Password*" (ver UC1.2. Recuperar *Password*.), sendo na sequência enviada uma mensagem por *e-mail* com um *link* para o formulário apresentado na Figura 4.3.



a) Outsystems *b) Django/Python*

Figura 4.3. UC1.3. Alterar Password (Outsystems vs. Django/Python)

4) UC1.4. Efetuar *Logout*

Para que seja possível ao utilizador terminar uma sessão ativa na aplicação, terá disponível um ícone referente ao *Logout* (que está localizado no canto superior direito da barra de tarefas). Este ícone estará presente em todas as páginas, com a exceção das páginas de *Login*, Recuperar *Password* e Alterar *Password*.

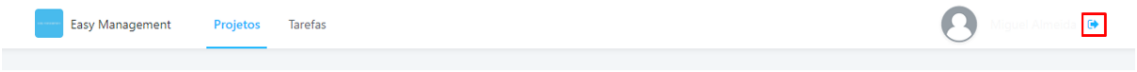


Figura 4.4. UC1.4. Efetuar Logout (Outsystems)

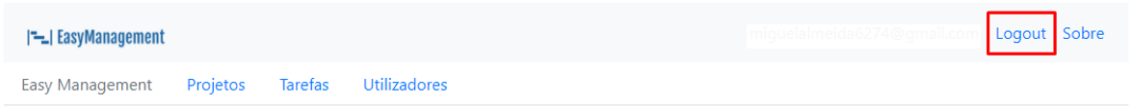
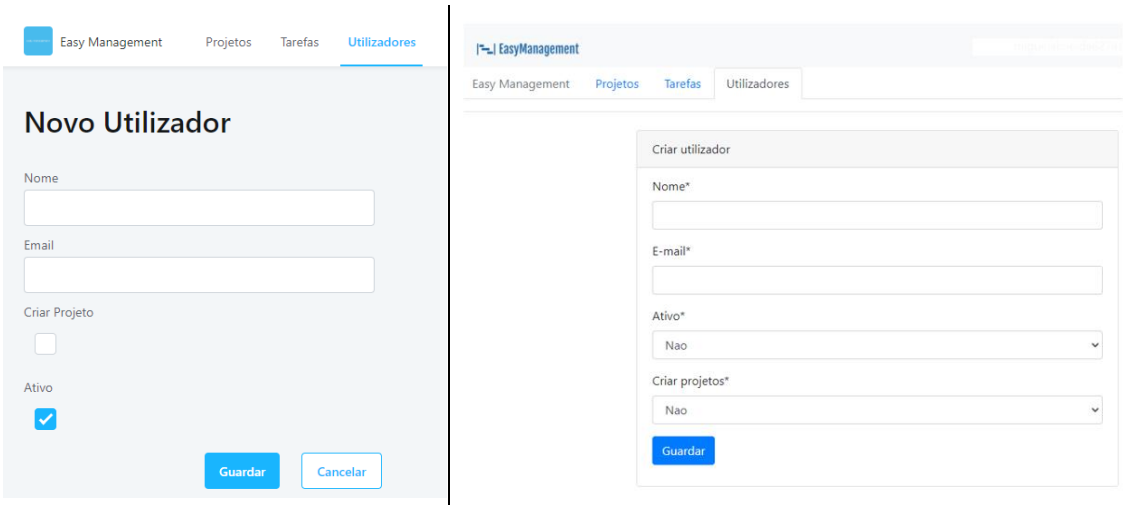


Figura 4.5. UC1.4. Efetuar Logout (Django/Python)

4.1.1.2 *Módulo Gestão de Utilizadores*

1) UC2.1. Criar Utilizador

Para criar um novo utilizador, o administrador terá primeiro de clicar no separador "Utilizadores", sendo então direcionado para a listagem de Utilizadores. De seguida, terá de clicar no *link* "Criar Utilizador". Posteriormente, será direcionado para o formulário apresentado na Figura 4.6. Neste formulário, o administrador insere o nome e email do novo utilizador. Pode, ainda, indicar se o mesmo tem permissões para criar novos projetos (*checkbox* Criar Projetos) e se está ativo para aceder à aplicação (*checkbox* Ativo).



a) Outsystems *b) Django/Python*

Figura 4.6. UC2.1. Criar Utilizador (Outsystems vs. Django/Python)

2) UC2.2. Listar Utilizadores

Para aceder à listagem de utilizadores, o administrador terá de clicar no separador "Utilizadores" presente na barra de tarefas. De seguida, será direcionado para a página que apresenta a lista dos utilizadores registados na aplicação, que se apresenta na Figura 4.7 e Figura 4.8.

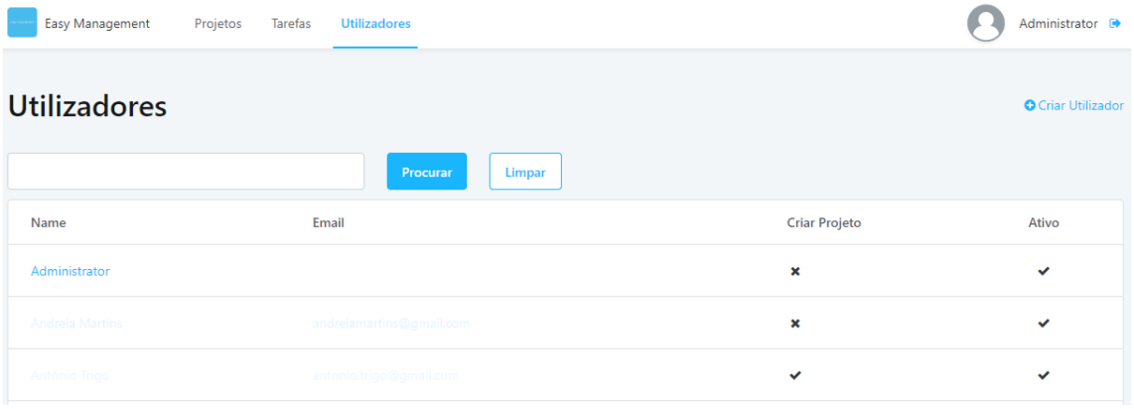


Figura 4.7. UC2.2. Listar Utilizadores (Outsystems)

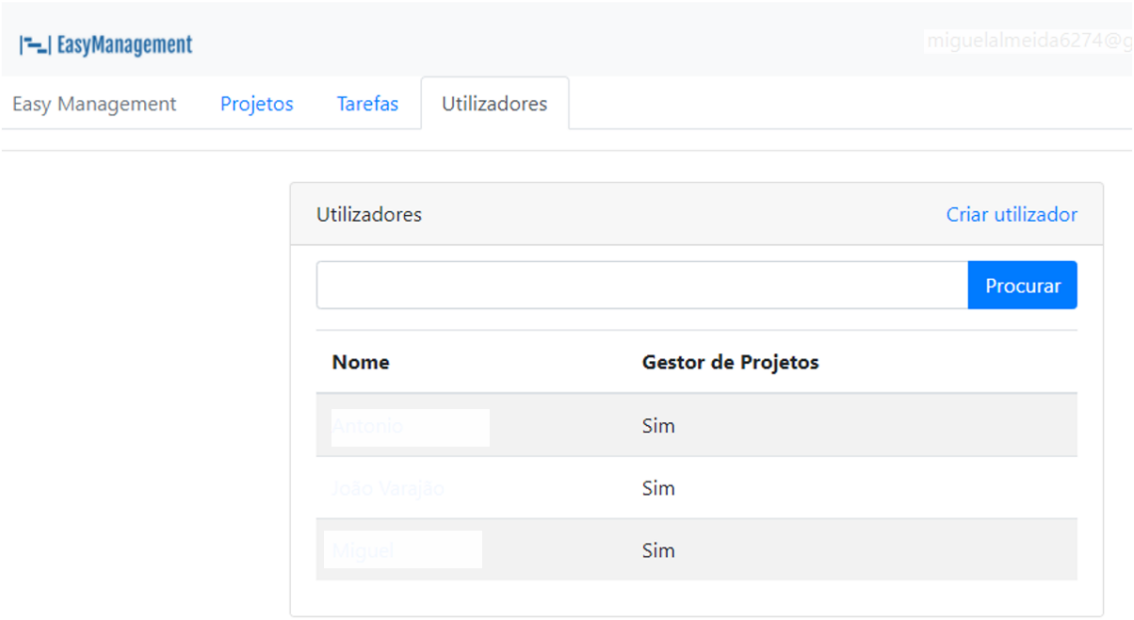


Figura 4.8. UC2.2. Listar Utilizadores (Django/Python)

3) UC2.3. Editar Utilizador

Para editar os dados de um utilizador já existente, o administrador terá primeiro de clicar no separador "Utilizadores", sendo então direcionado para a listagem de Utilizadores. De seguida, terá de clicar no nome do utilizador que pretende editar. Posteriormente, será direcionado para o formulário apresentado na Figura 4.9 e Figura 4.10.

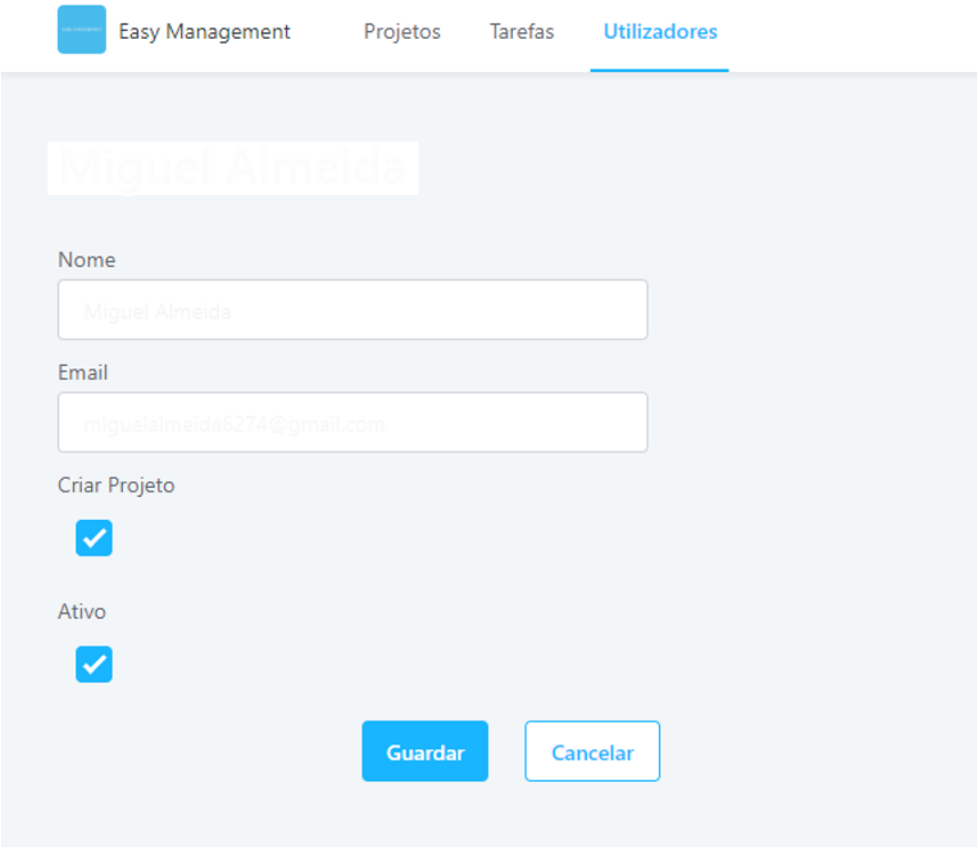


Figura 4.9. UC2.3. Editar Utilizador (Outsystems)

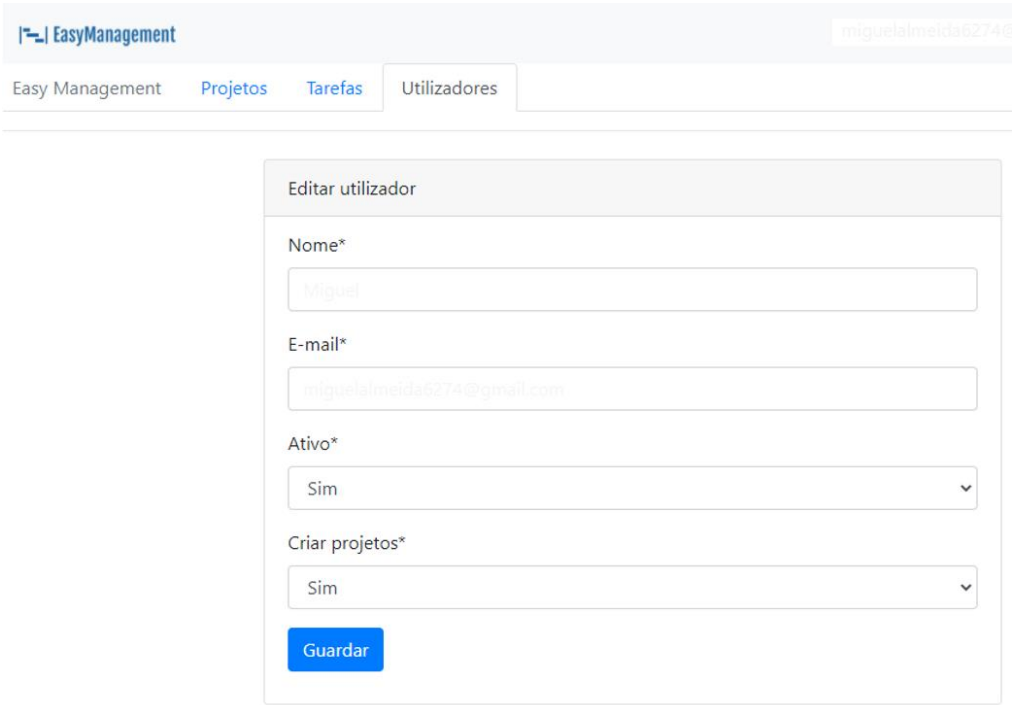


Figura 4.10. UC2.3. Editar Utilizador (Django/Python)

Módulo Gestão de Projetos e Tarefas

1) UC3.1. Criar Projeto

Para criar um novo projeto, cujo formulário se encontra na Figura 4.11 e Figura 4.12, o utilizador deverá clicar no link "Criar projeto" disponibilizado na página "Listar Projetos" (ver Figura 4.13 e Figura 4.14). Para criar um projeto, o utilizador necessita de inserir o código (identificador) e o nome do projeto, uma descrição, as datas de início e término do projeto, o estado do projeto (que pode assumir os valores: "Por fazer", "Em Andamento", "Pendente" e "Finalizado"), o esforço estimado (em minutos) e o esforço efetivo do projeto (em minutos).

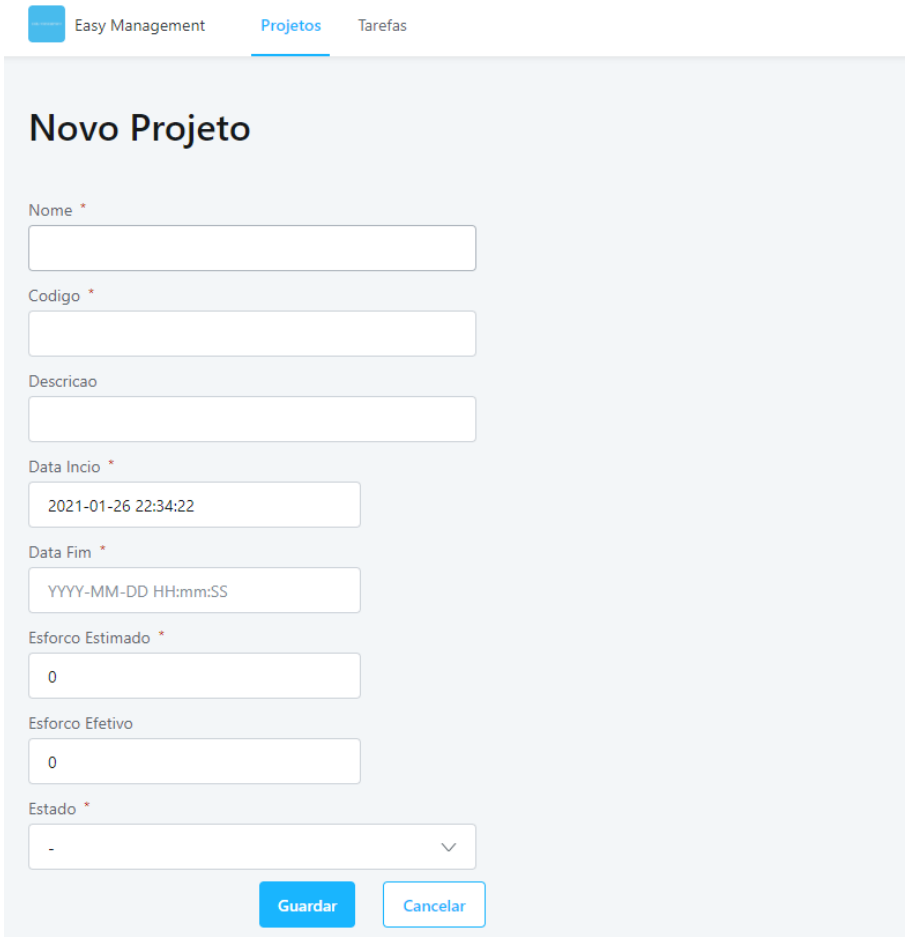


Figura 4.11. UC3.1. Criar Projeto (Outsystems)

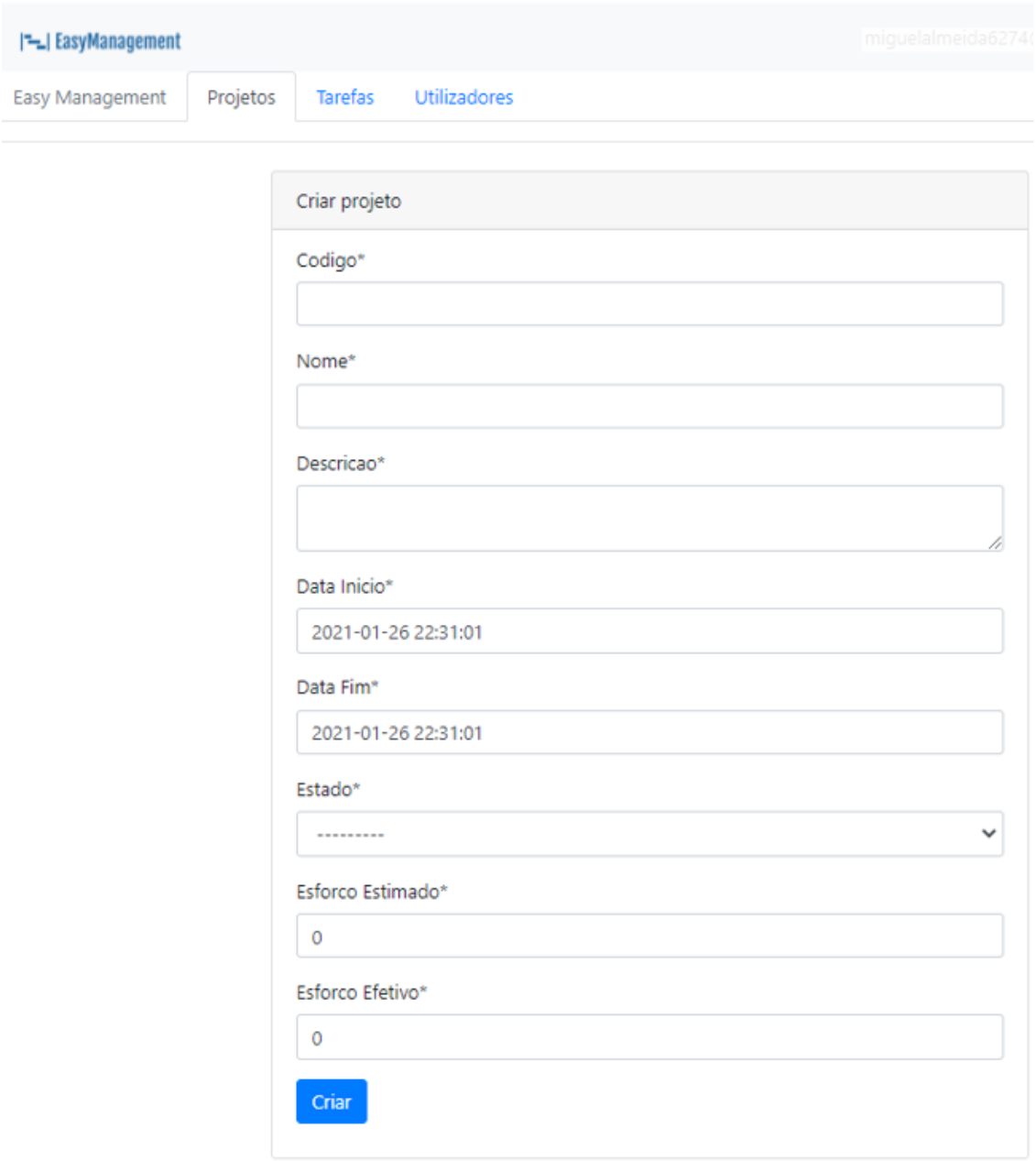


Figura 4.12. UC3.1. Criar Projeto (Django/Python)

2) UC3.2. Listar Projetos

Para aceder à listagem dos projetos, o utilizador necessita de clicar no separador "Projetos" presente na barra de tarefas, sendo assim direcionado para a página apresentada na Figura 4.13 e Figura 4.14. Para efetuar uma pesquisa na lista de projetos, o utilizador deve inserir uma expressão (referente a pelo menos parte do nome do projeto que pretende encontrar) na caixa de texto "Procurar" e clicar no botão "Procurar", sendo então apresentados apenas os projetos que contenham no seu nome a expressão inserida. O botão "Limpar" elimina o texto inserido na caixa de texto "Procurar", sendo listados todos os projetos novamente.

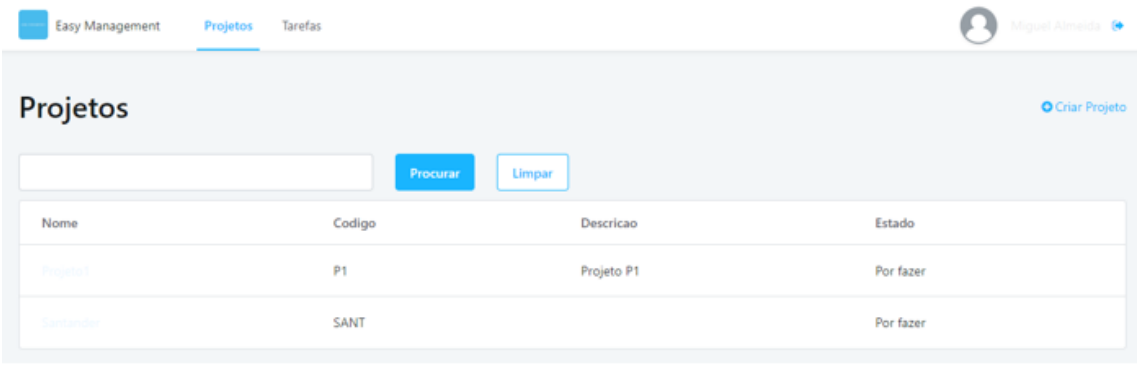


Figura 4.13. UC3.2. Listar Projetos (Outsystems)

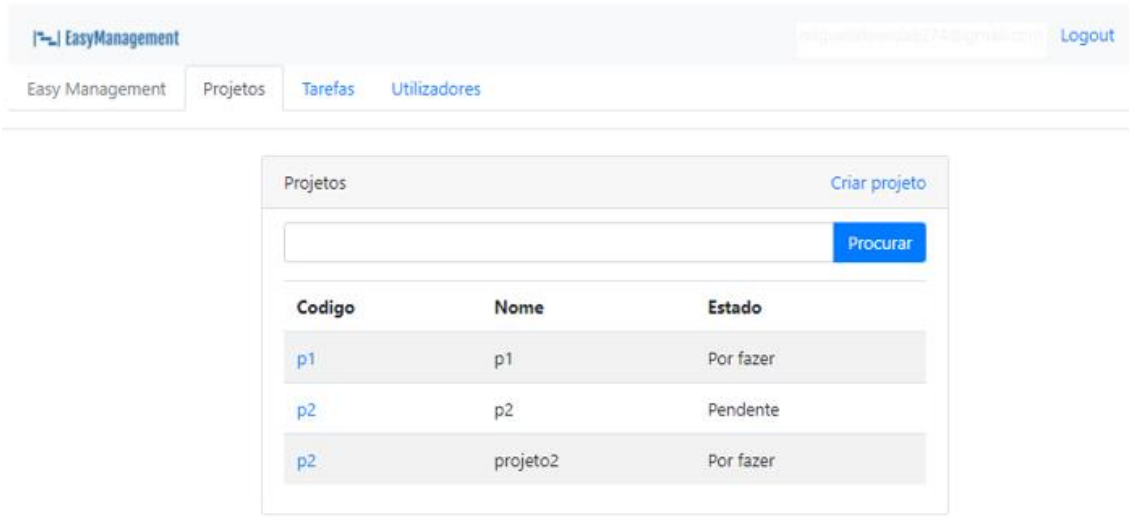


Figura 4.14. UC3.2. Listar Projetos (Django/Python)

3) UC3.3. Editar Projeto

Para aceder à página de edição de projeto (ver Figura 4.15 e Figura 4.16) o utilizador terá de aceder primeiro à página de listagem dos projetos (ver Figura 4.13 e Figura 4.14) e selecionar o projeto que quer editar. Esta página possui duas áreas distintas: a primeira, do lado esquerdo, que lista todos os projetos a que o utilizador está associado, permitindo ao utilizador selecionar o projeto deseja visualizar (a opção gráfica foi a *checkbox* mas pode-se utilizar outra que identifique o projeto selecionado como *radiobuttons* ou destacar o projeto selecionado colorindo o nome do mesmo); e a segunda, do lado direito da primeira, que possui três separadores, que permitem o acesso a três áreas distintas da aplicação: informações, tarefas e colaboradores.

No separador "Informações", é apresentado um formulário que permite visualizar/editar os detalhes do projeto, com um botão no canto superior direito com o ícone "Disquete", que permite guardar as alterações efetuadas ao projeto.

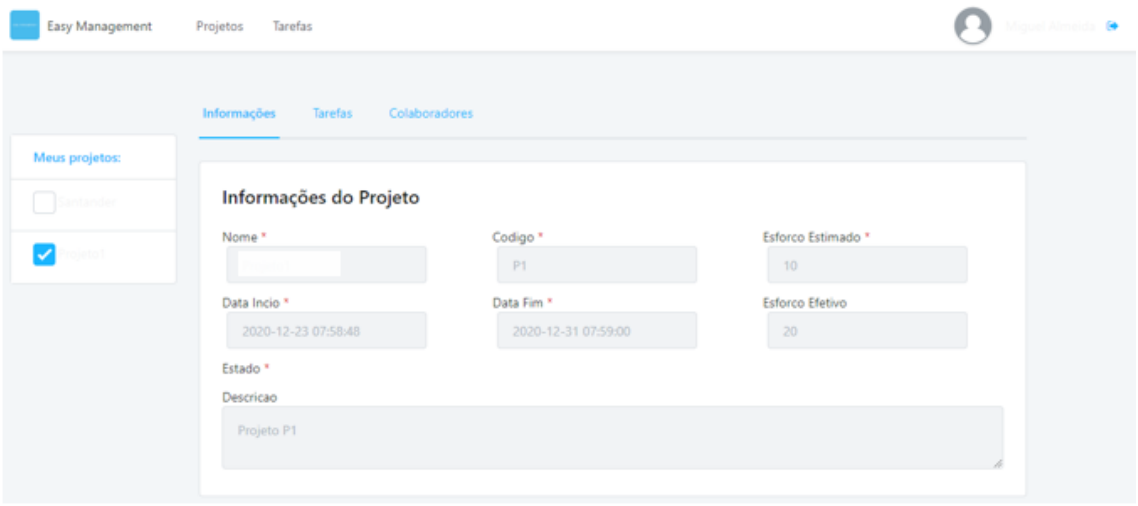


Figura 4.15. UC3.3. Editar Projeto (Outsystems)

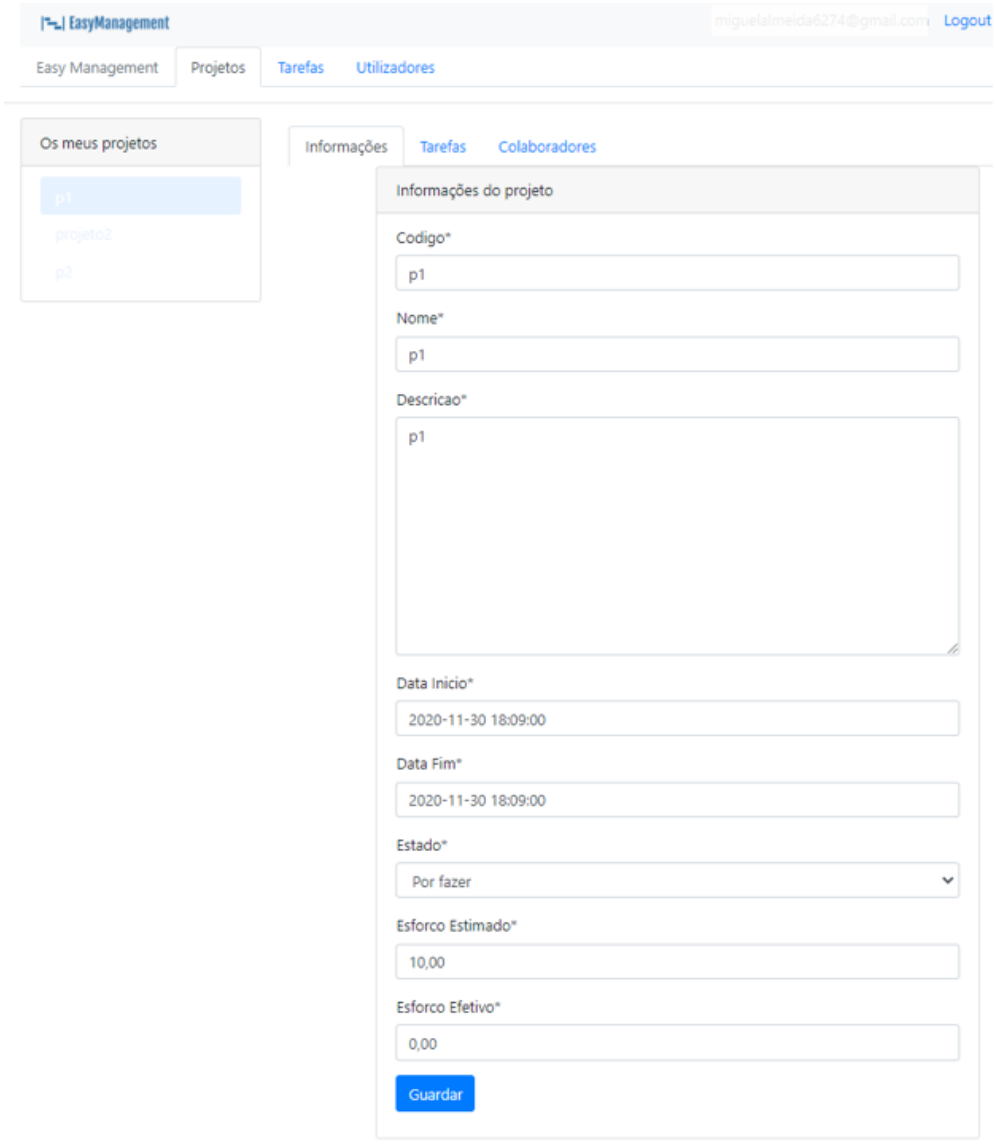


Figura 4.16. UC3.3. Editar Projeto (Django/Python)

4) UC3.5. Adicionar Colaborador ao Projeto

A página da Figura 4.17 e Figura 4.18 possibilita ao Gestor do Projeto selecionar colaboradores ao projeto, possibilitando que posteriormente sejam atribuídas tarefas para serem executadas.

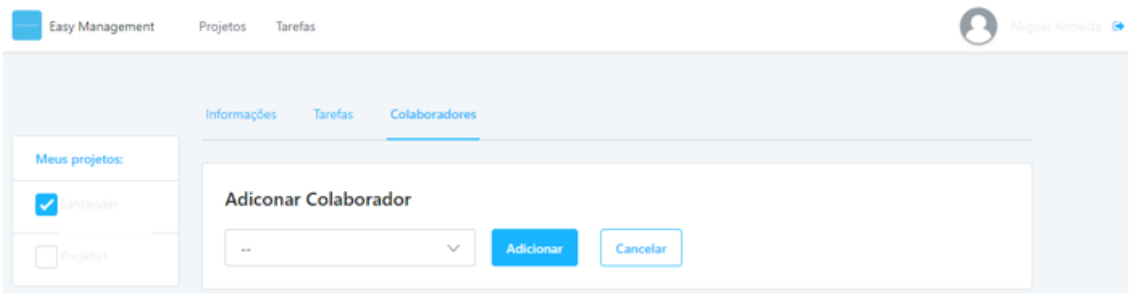


Figura 4.17. UC3.5. Adicionar Colaborador ao Projeto (Outsystems)

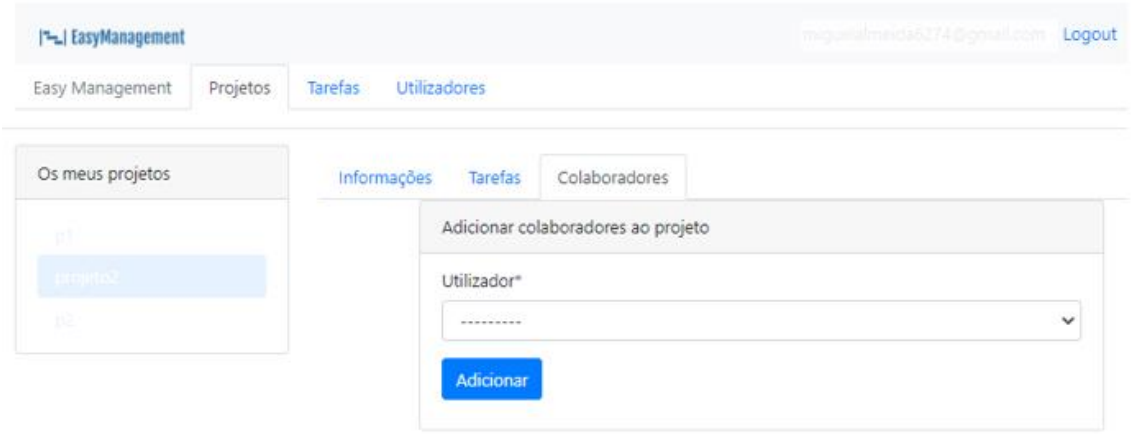


Figura 4.18. UC3.5. Adicionar Colaborador ao Projeto (Django/Python)

5) UC3.6. Listar Colaboradores do Projeto

Para visualizar os colaboradores associados ao projeto, o utilizar deverá clicar no separador "Colaboradores", sendo-lhe apresentada uma página com a lista dos colaboradores associados ao projeto (ver Figura 4.19 e Figura 4.20). No topo da lista do lado direito existe a opção adicionar colaborador, a qual permite adicionar um novo colaborador ao projeto selecionado.

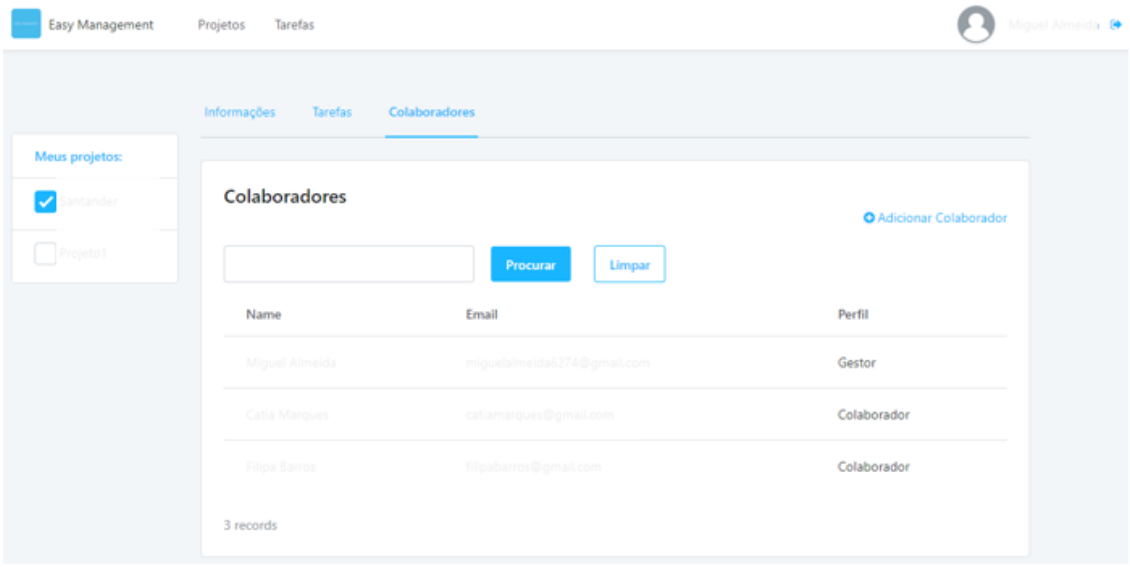


Figura 4.19. UC3.6. Listar Colaboradores do Projeto (Outsystems)

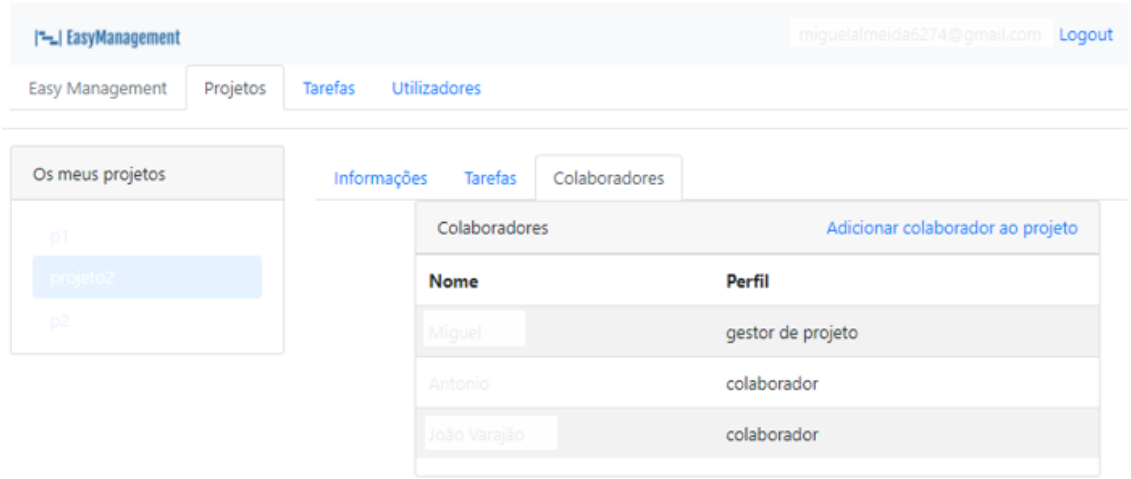


Figura 4.20. UC3.6. Listar Colaboradores do Projeto (Django/Python)

6) UC3.7. Criar Tarefa do Projeto

Para aceder à funcionalidade Criar Tarefa, o utilizador deverá clicar no botão que se encontra no canto superior do lado direito da página Listar Tarefas (ver Figura 4.23 e Figura 4.24). Será então direcionado para um formulário que permite a criação de tarefas. Esse formulário possui vários campos, nomeadamente, código, nome, estado ("Por fazer", "Em Andamento", "Pendente" e "Finalizado"), o colaborador que irá realizar a tarefa, uma breve descrição, a data de início e data de fim prevista de conclusão da tarefa, e ainda os esforços estimado e efetivo de execução da tarefa (em minutos). Alguns dos valores destes campos, como, por exemplo, o estado da tarefa, serão posteriormente atualizados durante a execução da tarefa.

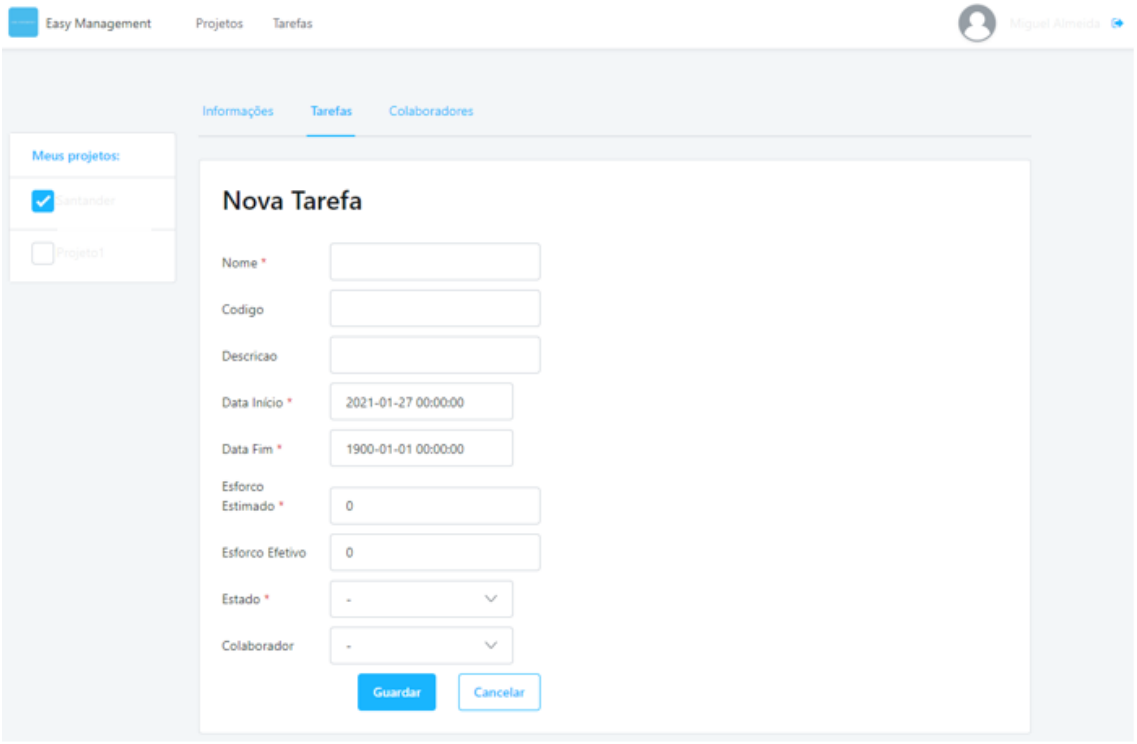


Figura 4.21. UC3.7. Criar Tarefa do Projeto (Outsystems)

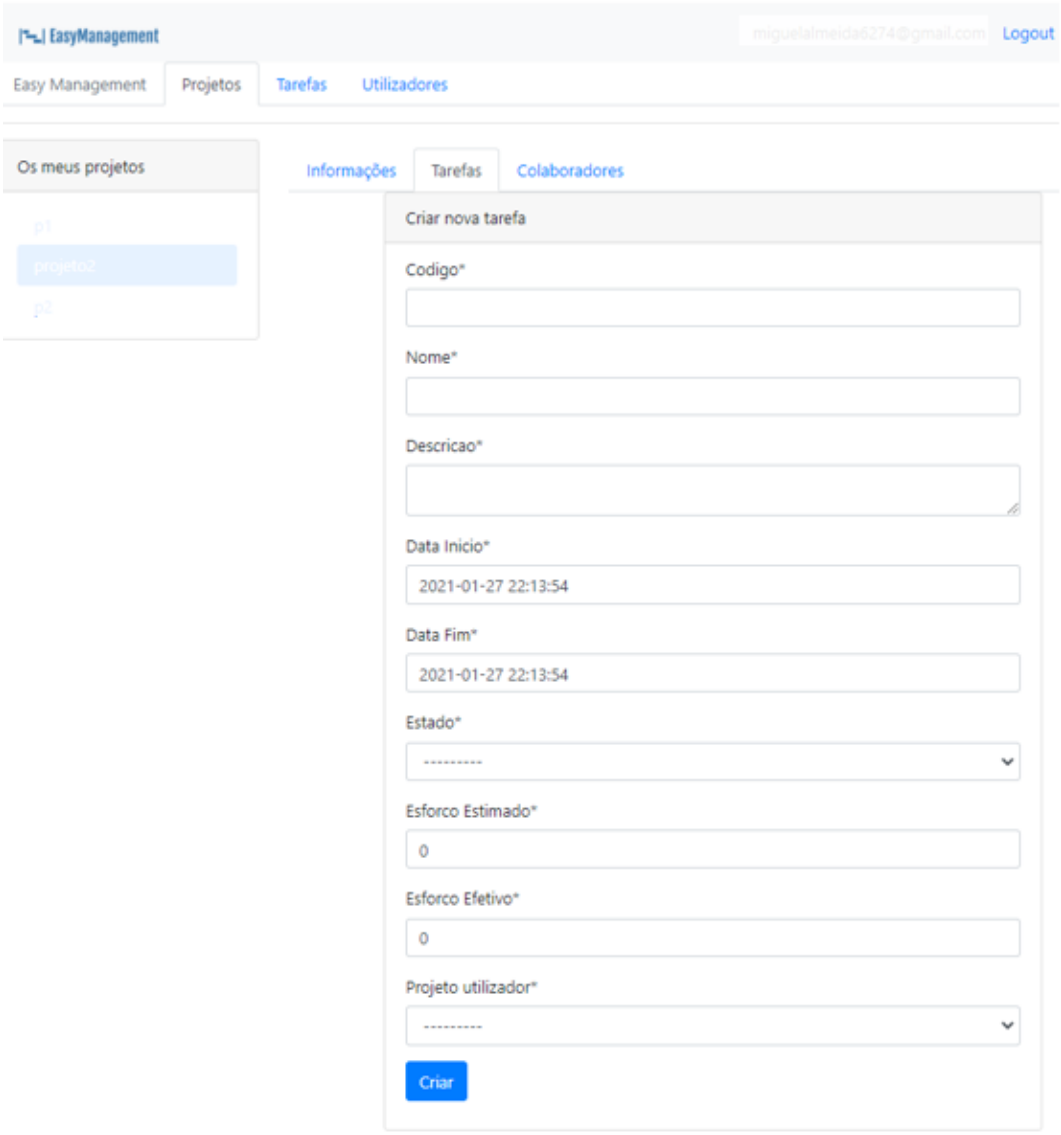


Figura 4.22. UC3.7. Criar Tarefa do Projeto (Django/Python)

7) UC3.8. Listar Tarefas do Projeto

Clicando no separador relativo às tarefas, o utilizador acede à lista de tarefas associadas ao projeto selecionado (ver Figura 4.23 e Figura 4.24). A partir desta lista, o utilizador tem as opções de criar tarefa (link no canto superior direito) e editar tarefa, clicando na tarefa desejada.

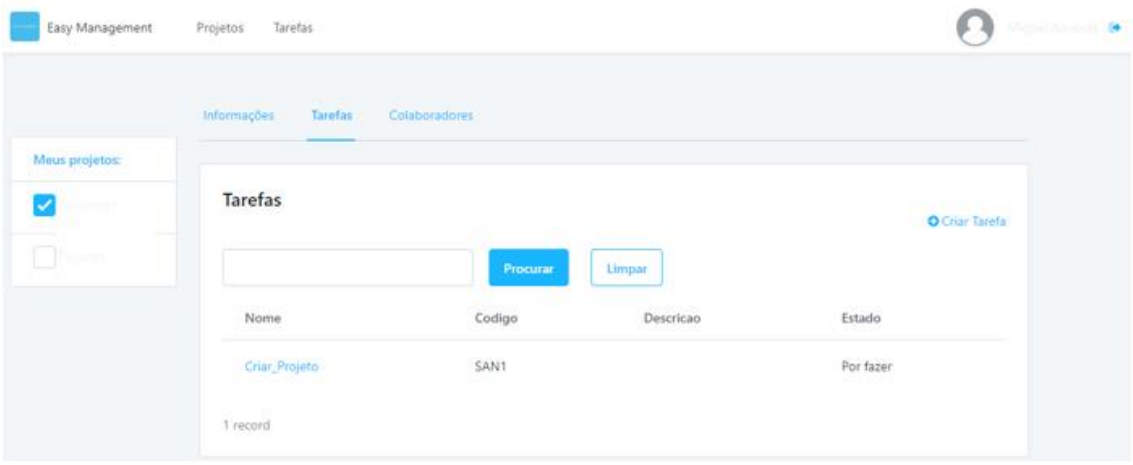


Figura 4.23. UC3.8. Listar Tarefas do Projeto (Outsystems)

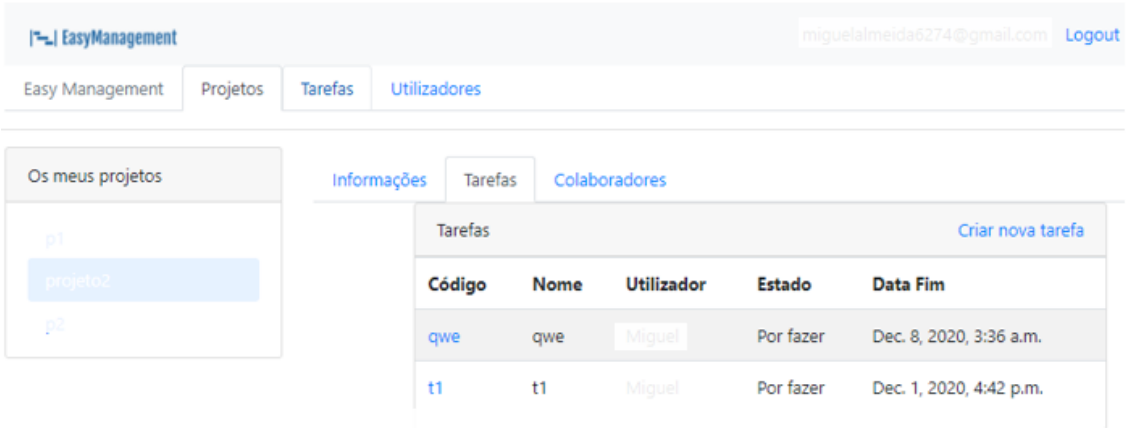


Figura 4.24. UC3.8. Listar Tarefas do Projeto (Django/Python)

8) UC3.9. Listar Tarefas do Colaborador

O separador "Tarefas" permite ao colaborador aceder à lista das tarefas (ver Figura 4.25 e Figura 4.26) que lhe estão associadas e que não estão no estado "Finalizado". Funciona como um *dashboard* das tarefas a realizar pelo utilizador em questão, existindo também a hipótese de filtrar as tarefas apresentadas por nome. É possível clicar numa das tarefas da lista e ir para o formulário de edição da mesma (ver Figura 4.27 e Figura 4.28).

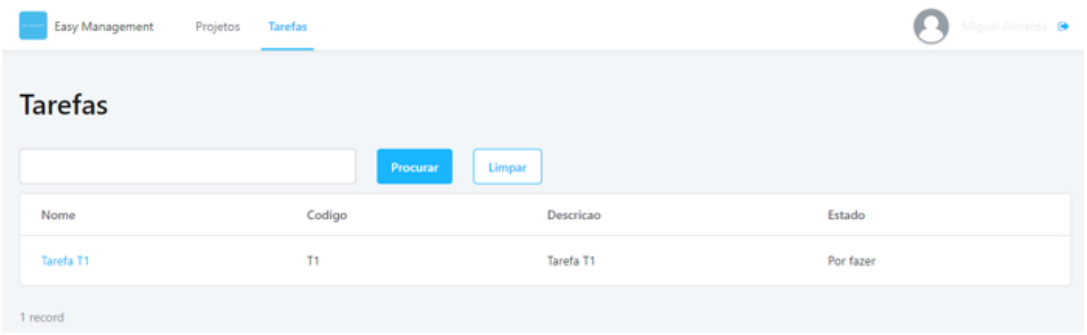


Figura 4.25. UC3.9. Listar Tarefas do Colaborador (Outsystems)

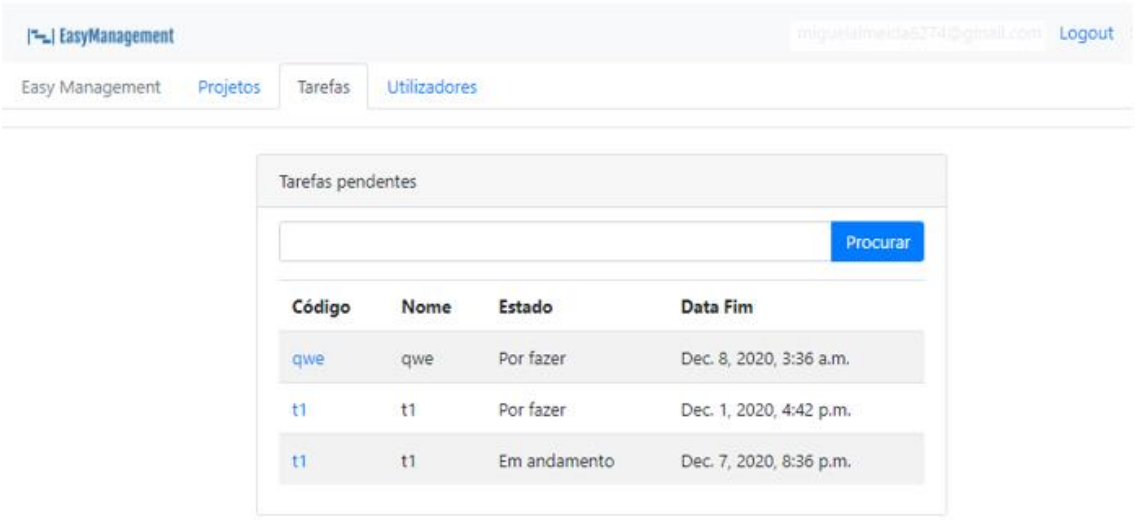


Figura 4.26. UC3.9. Listar Tarefas do Colaborador (Django/Python)

9) UC3.10. Editar Tarefa do Projeto

Para aceder à funcionalidade de Editar Tarefa, o utilizador deverá selecionar, na lista de tarefas, a tarefa desejada, sendo então redirecionado para a página de visualização/edição de tarefa. Após alterar os campos desejados, o utilizador deverá clicar no botão guardar para submeter as alterações efetuadas. Esta ação só será possível se o utilizador for gestor do projeto.

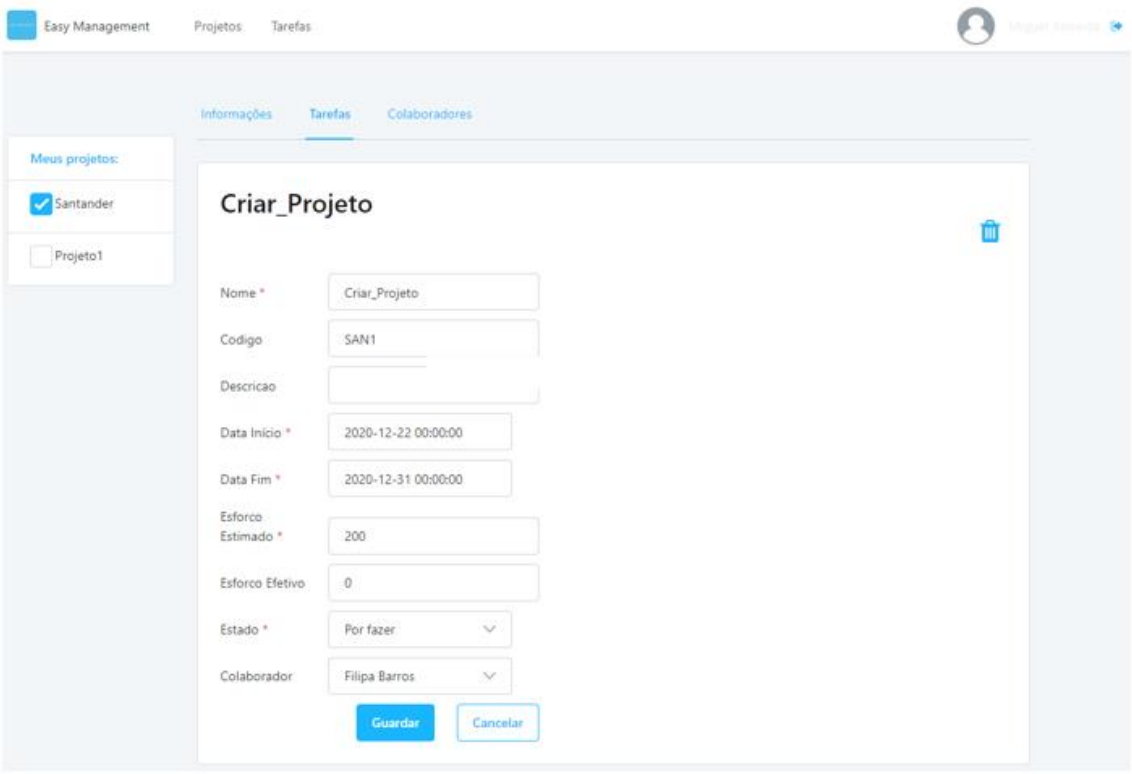


Figura 4.27. UC3.10. Editar Tarefa do Projeto (Outsystems)

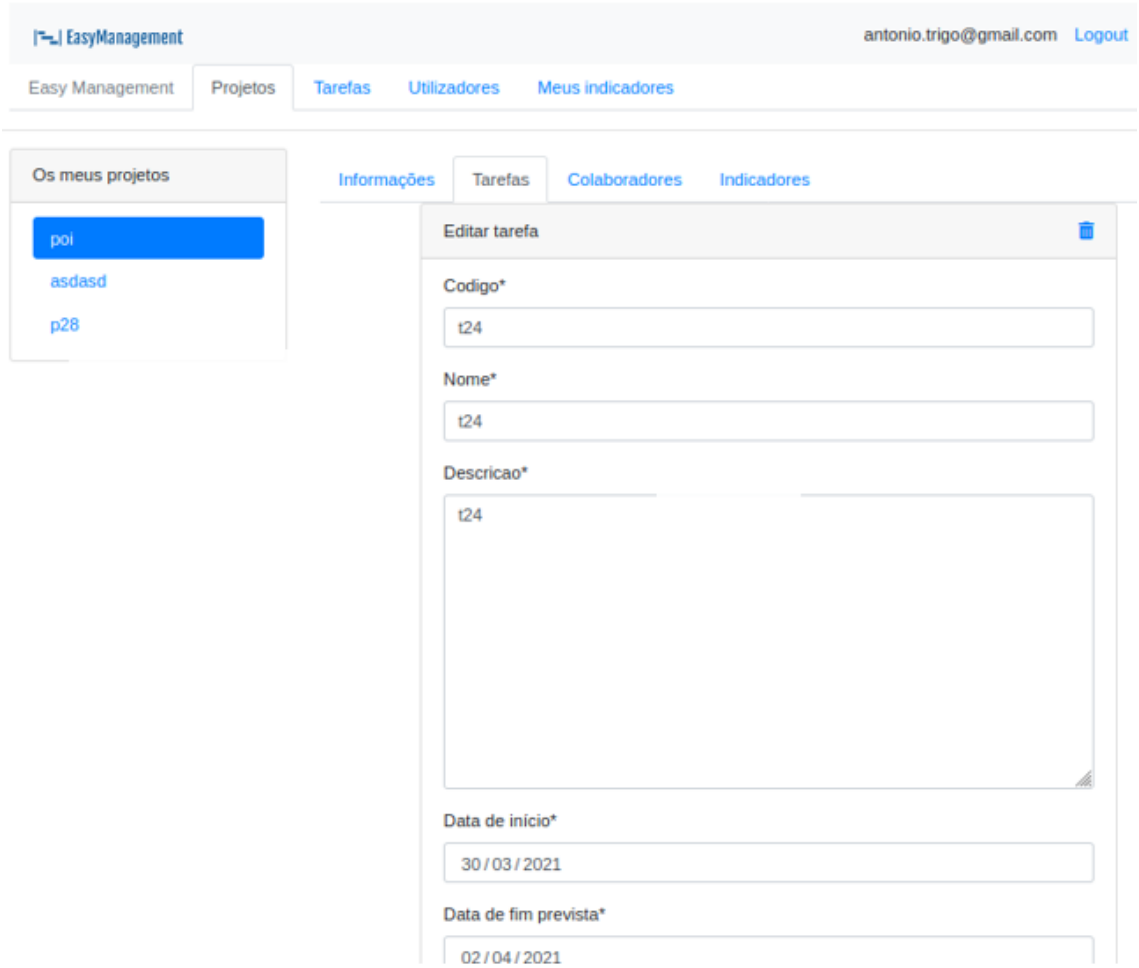


Figura 4.28. UC3.10. Editar Tarefa do Projeto (Django/Python)

10) UC3.11. Eliminar Tarefa do Projeto

Para eliminar uma tarefa do projeto, o utilizador deve clicar no botão com o ícone de "Caixote do Lixo" disponível no canto superior direito (ver Figura 4.27 e Figura 4.28). Esta ação só será possível se o utilizador for gestor do projeto

4.1.2 Fase II – Manutenção de *software*

Nesta etapa desenvolveram-se as funcionalidades descritas no "Protocolo para Realização da Experiência Laboratorial – Fase II" presente no *Apêndice 2*.

4.1.2.1 *Módulo Gestão de Projetos e Tarefas*

1) UC3.7 Criar Tarefa do Projeto

Para aceder à funcionalidade Criar Tarefa, o utilizador deverá clicar no botão que se encontra no canto superior do lado direito da página Listar Tarefas (ver Figura 4.23 e Figura 4.24). Será então direcionado para um formulário que permite a criação de tarefas apresentado na Figura 4.29 e na Figura 4.30.

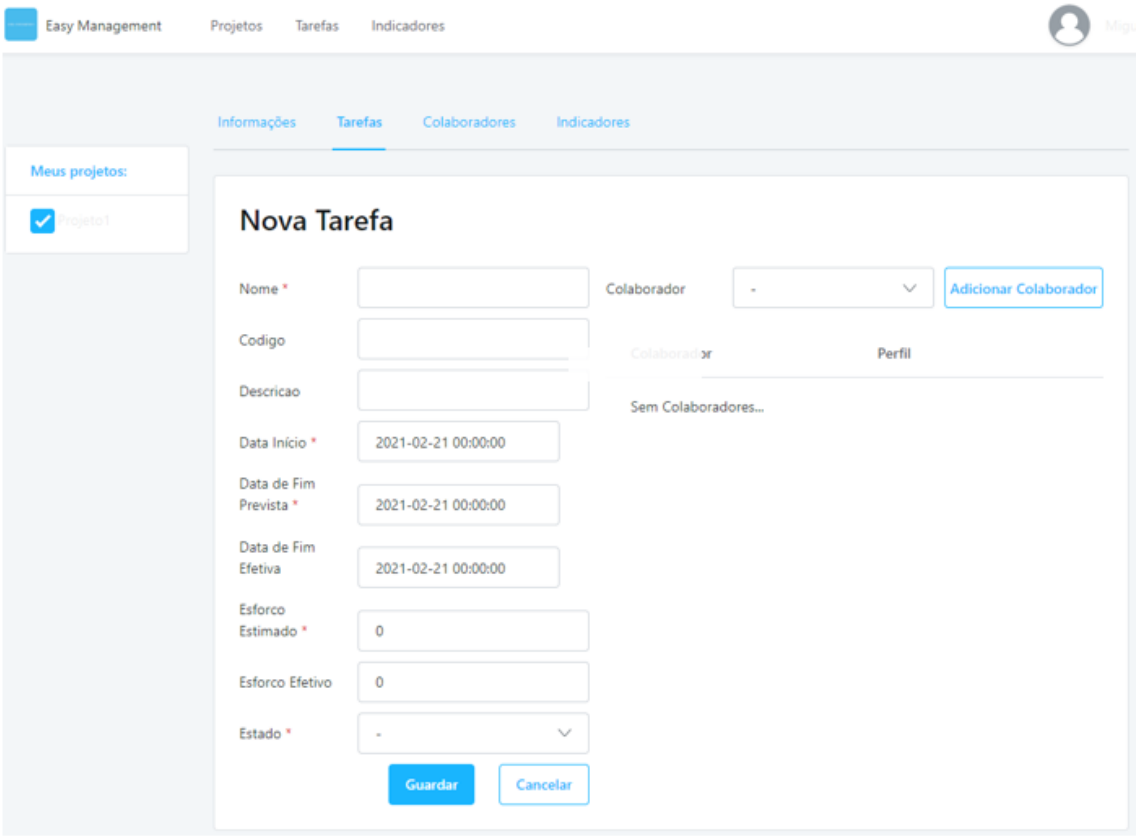


Figura 4.29. UC3.7 Criar Tarefa do Projeto (Outsystems)

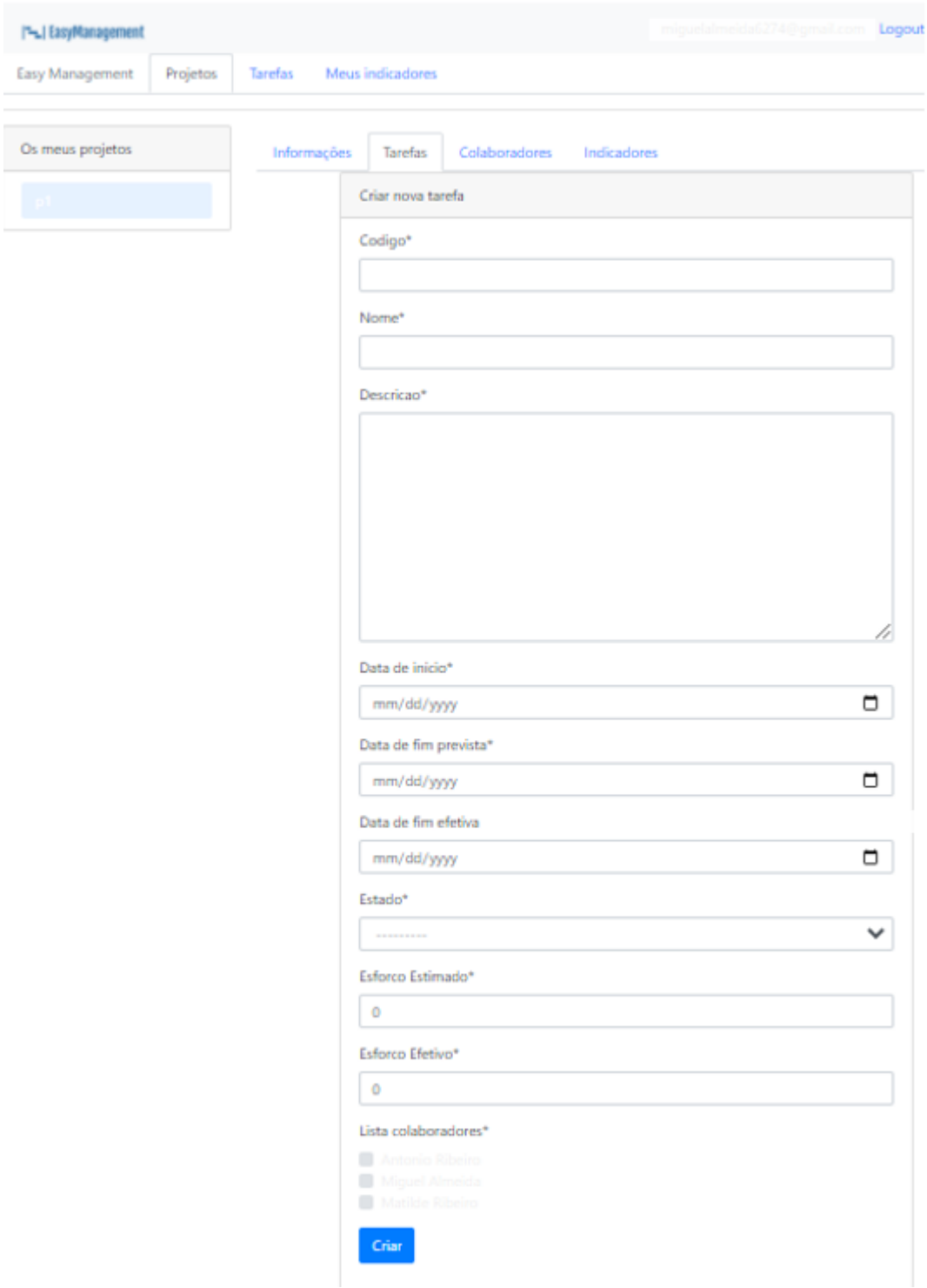


Figura 4.30. UC3.7 Criar Tarefa do Projeto (Django/Python)

O formulário apresentado nas duas figuras anteriores (Figura 4.29 e Figura 4.30) possui vários campos, nomeadamente, código, nome, estado (“Por fazer”, “Em Andamento”, “Pendente” e “Finalizado”), os colaboradores que vão realizar a tarefa, uma breve descrição, a data de início e data de fim prevista e a data de fim efetiva da tarefa, e ainda os esforços estimado e efetivo de execução da tarefa (em minutos). Alguns dos valores

destes campos, como, por exemplo, o estado da tarefa, serão posteriormente atualizados durante a execução da tarefa.

2) UC3.10 Editar Tarefa do Projeto

Para aceder à funcionalidade de Editar Tarefa, o utilizador deverá selecionar, na lista de tarefas, a tarefa desejada, sendo então redirecionado para a página de visualização/edição de tarefa. Após alterar os campos desejados, o utilizador deverá clicar no botão guardar para submeter as alterações efetuadas. Para além da edição da tarefa o utilizador pode eliminar uma tarefa clicando no botão com o ícone de "Caixote do Lixo" disponível no canto superior direito.

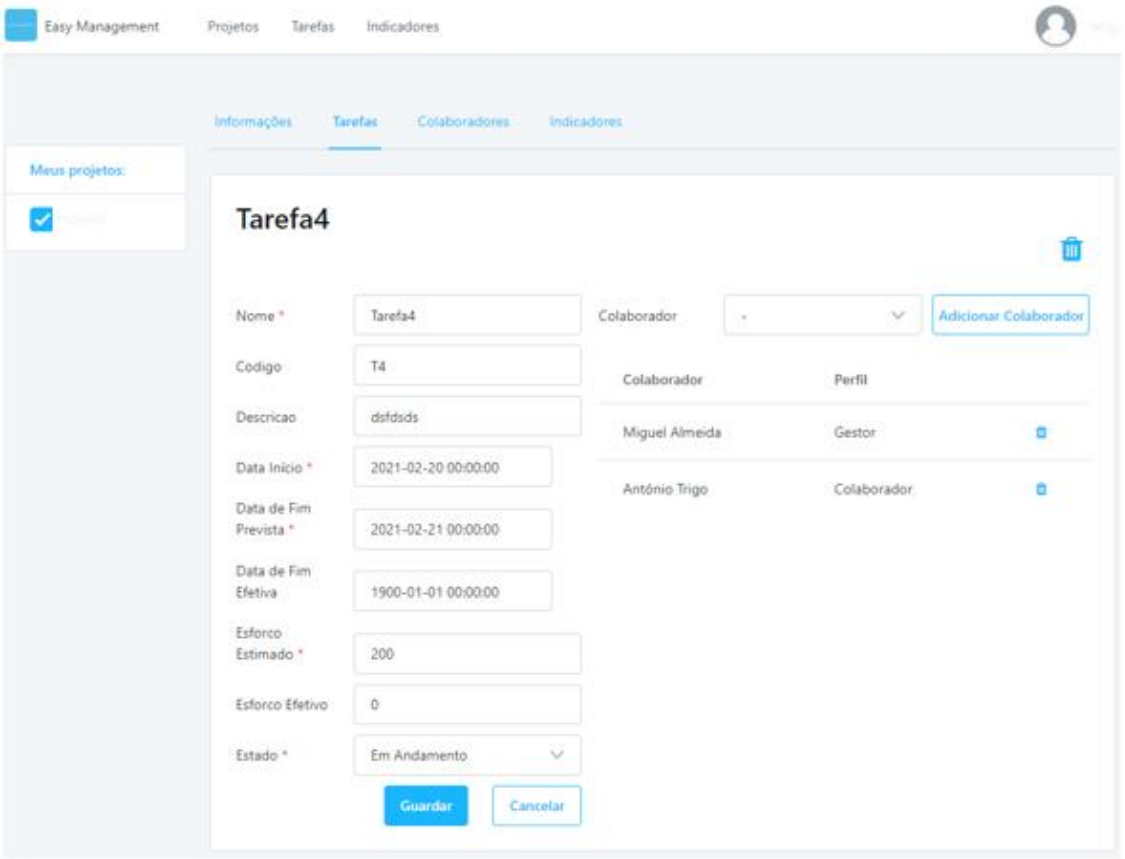


Figura 4.31. UC3.10 Editar Tarefa do Projeto (Outsystems)

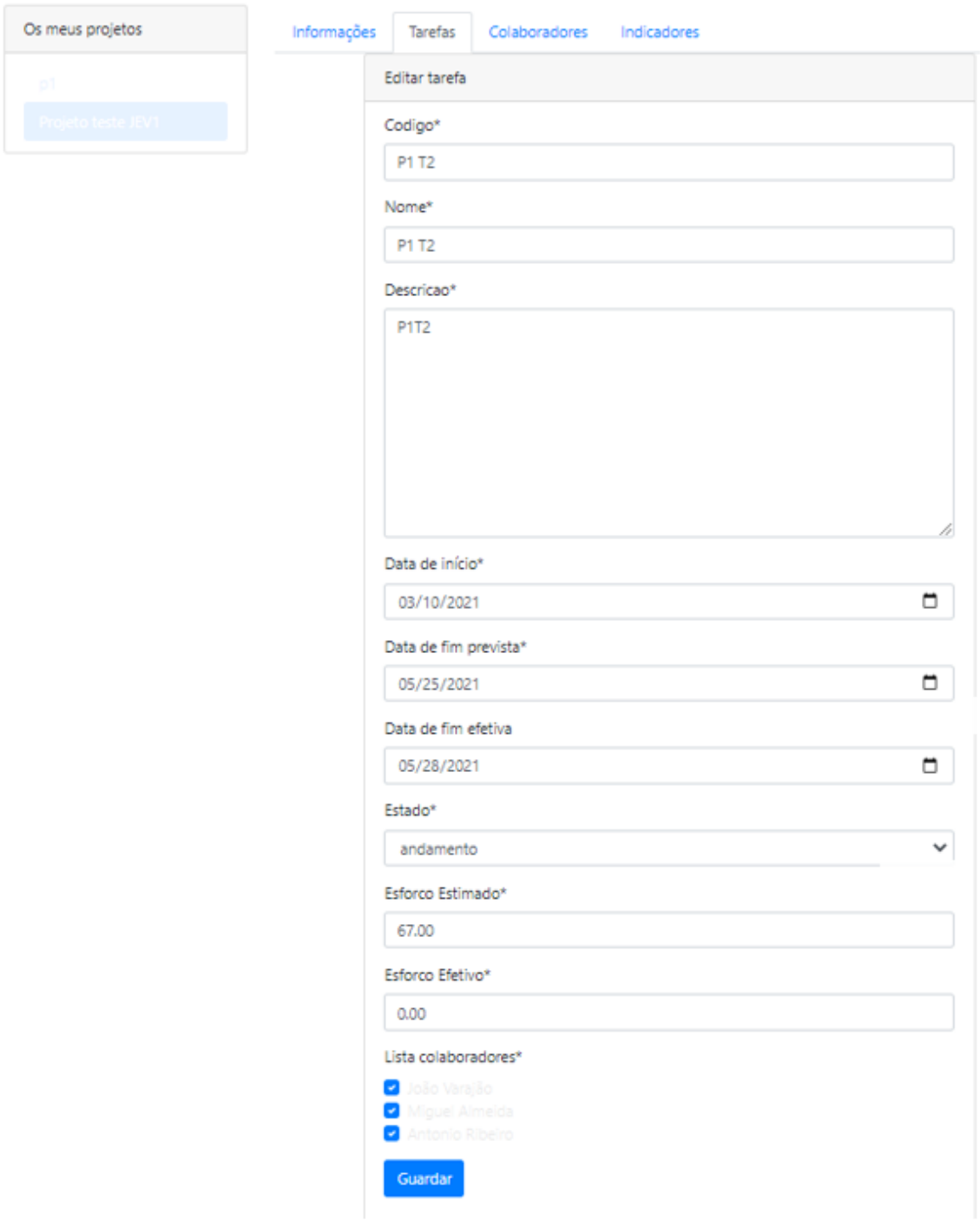


Figura 4.32. UC3.10 Editar Tarefa do Projeto (Django/Python)

3) UC3.12 Consultar Indicadores do Projeto

Para aceder à página de indicadores de projeto o utilizador terá de aceder primeiro à página de listagem dos projetos e selecionar o projeto. Posteriormente, deve clicar no separador "Indicadores" onde são apresentados os indicadores do projeto selecionado.

Figura 4.33. UC3.12 Consultar Indicadores do Projeto (Outsystems)

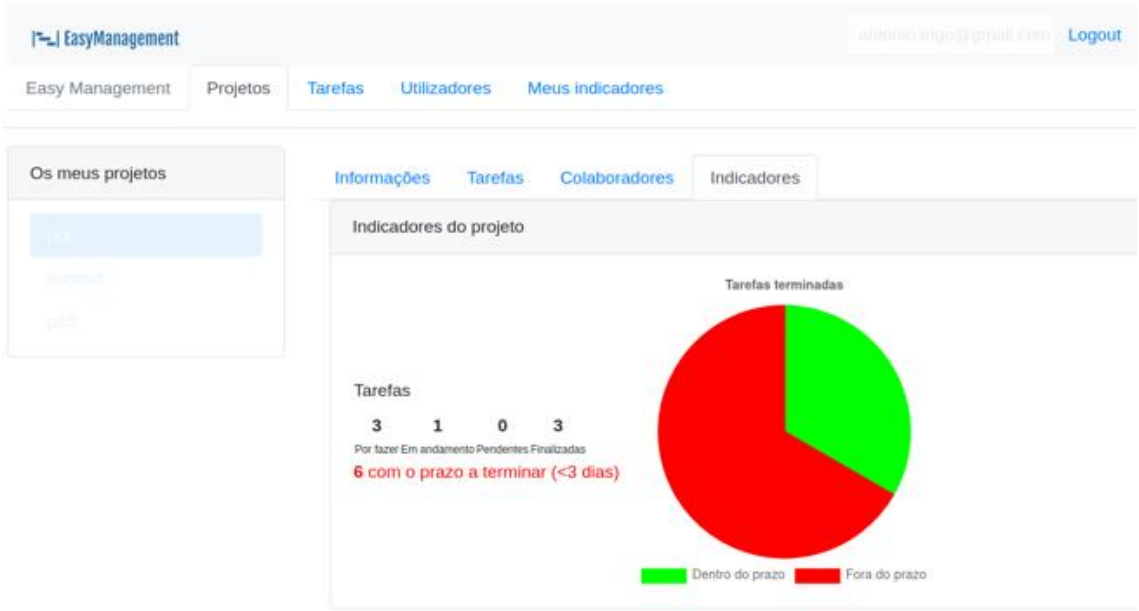


Figura 4.34. UC3.12 Consultar Indicadores do Projeto (Django/Python)

4) UC3.13 Consultar Indicadores do Colaborador

Para aceder à listagem de utilizadores, o administrador terá de clicar no separador "Indicadores" presente na barra de tarefas. De seguida, será direcionado para a página onde são apresentados os indicadores do colaborador.

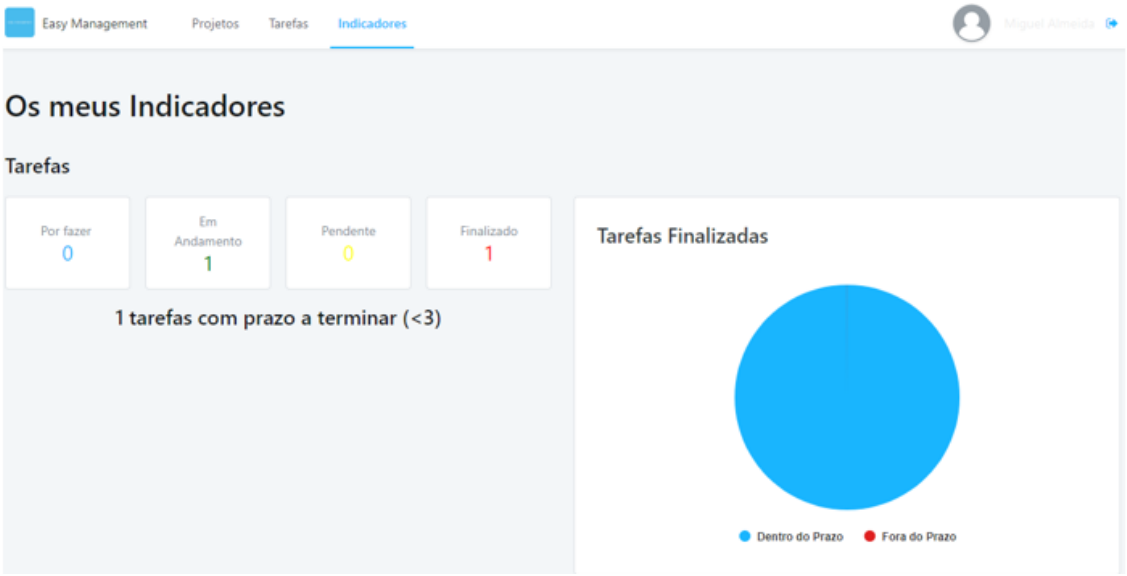


Figura 4.35. UC3.13 Consultar Indicadores do Colaborador (Outsystems)

Figura 4.36. UC3.13 Consultar Indicadores do Colaborador (Django/Python)

4.2 Avaliação das soluções

Nesta secção avalia-se as duas soluções implementadas em termos do tempo de implementação de cada uma das funcionalidades.

4.2.1 Fase I – Desenvolvimento de *software*

No decorrer do desenvolvimento foi registado o tempo de desenvolvimento de cada caso de uso em cada tecnologia, como se apresenta na Figura 4.37. Os tempos são apresentados em minutos e depois convertidos para horas para uma mais fácil comparação. O desenvolvimento foi registado em vídeo e acompanhado de forma a garantir uma avaliação independente.

Outsystems	
Casos de Uso	Tempo real (minutos)
UC1.1. Efetuar Login	0
UC1.2.Recuperar Password	131
UC1.3. Alterar Password	57
UC1.4. Efetuar Logout	0
UC2.1. Criar Utilizador	19
UC2.2. Listar Utilizadores	3
UC2.3. Editar Utilizador	5
UC3.1. Criar Projeto	22
UC3.2. Listar Projetos	3
UC3.3. Editar Projeto	88
UC3.5. Adicionar Colaborador ao Projeto	17
UC3.6. Listar Colaboradores do Projeto	8
UC3.7. Criar Tarefa do Projeto	83
UC3.8. Listar Tarefas do Projeto	9
UC3.9. Listar Tarefas do Colaborador	4
UC3.10. Editar Tarefa do Projeto	49
UC3.11. Eliminar Tarefa do Projeto	6
Total	**504**
Em horas	**8,4**

Django/Phyton	
Casos de Uso	Tempo real (minutos)
UC1.1. Efetuar Login	21
UC1.2.Recuperar Password	326
UC1.3. Alterar Password	81
UC1.4. Efetuar Logout	8
UC2.1. Criar Utilizador	74
UC2.2. Listar Utilizadores	86
UC2.3. Editar Utilizador	45
UC3.1. Criar Projeto	78
UC3.2. Listar Projetos	74
UC3.3. Editar Projeto	63
UC3.5. Adicionar Colaborador ao Projeto	76
UC3.6. Listar Colaboradores do Projeto	71
UC3.7. Criar Tarefa do Projeto	92
UC3.8. Listar Tarefas do Projeto	71
UC3.9. Listar Tarefas do Colaborador	66
UC3.10. Editar Tarefa do Projeto	96
UC3.11. Eliminar Tarefa do Projeto	41
Total	**1369**
Em horas	**22,8**

Figura 4.37. Tempo real de desenvolvimento por Caso de Uso – Fase I (Outsystems & Django/Phyton)

Começando pelo tempo total, conseguimos observar que na tecnologia *Outsystems* o desenvolvimento aplicacional teve uma duração de 504 minutos (8,4 horas), já na tecnologia *Django/Phyton*, o tempo despendido foi de 1369 minutos (22,8 horas). Destes tempos pode-se concluir que a tecnologia *low-code* com base em *Outsystems* nesta experiência revelou-se 3x mais rápida que *source code*, baseada em *Django/Phyton*, o que evidencia ganhos de produtividade para a tecnologia *low-code*, permitindo responder à questão de investigação deste trabalho, ou seja, conclui-se desta primeira experiência que a tecnologia *low-code* permite maior produtividade que a tecnologia *source code* no que toca ao desenvolvimento de *software*. Em termos absolutos a diferença foi de 14,4 horas.

Ao analisar especificamente os casos de uso, UC1.1. Efetuar *Login* e UC1.4. Efetuar *Logout*, na tecnologia *Outsystems*, reparamos que o tempo real é igual a zero, isto deve-se ao facto destas funcionalidades serem implementadas por omissão quando o projeto é criado, o que facilitam bastante o desenvolvimento. Relativamente à gestão de utilizadores, na tecnologia *Django/Phyton*, embora já existam modelos de gestão de utilizadores, é necessário fazer algumas alterações aos mesmos, ou seja, despender tempo em parametrizações.

Havendo agora uma visão geral da implementação dos casos de uso de cada tecnologia, é evidente que praticamente todos foram finalizados em menor tempo na tecnologia *Outsystems*, sendo isso refletido no tempo total anteriormente analisado.

Complementando o desenvolvimento por caso de uso, existiu também tempo despendido noutras ações diretamente relacionadas com o desenvolvimento do projeto, como se pode observar na Figura 4.38.

Outsystems	
Outras configurações/testes referentes ao projeto	**Tempo real (minutos)**
Criação do Projeto	2
Criação da Base de Dados	14
Criação das ações de Criar/Editar/Eliminar de cada entidade	16
Testes aplicacionais	180
Total	**212**
Em horas	**3,53**
Total de Tempo Gasto (em horas)	**11,93**
Django/Phyton	
Outras configurações/testes referentes ao projeto	**Tempo real (minutos)**
Configuração inicial do projeto (ambiente virtual, base de dados, servidor de email, github, etc.)	190
Criação do modelo de dados inicial	126
Criação dos templates base	428
Configuração do ambiente de deploy (heroku, github, etc.)	70
Testes aplicacionais	120
Total	**934**
Em horas	**15,57**
Total de Tempo Gasto (em horas)	**38,38**

Figura 4.38. Tempo despendido noutras configurações/ações – Fase I (Outsystems & Django/Phyton)

É visível que as configurações/ações apresentadas nas duas tecnologias são diferentes. Tanto a criação do projeto como a da base de dados em *Outsystems* é bastante simples e não é necessária nenhuma configuração externa à aplicação. É observável que, para efetuar estas ações, foram necessários apenas 32 minutos, um tempo bastante razoável tendo em comparação a tecnologia *Django/Phyton*, que necessitou de 934 minutos (15,57 horas) para configurações/ações inerentes à criação do projeto.

Em suma, o tempo total gasto em *Outsystems* foi de 11,93 horas, sendo a estas acrescentado testes efetuados à aplicação. Quanto ao *Django/Phyton*, obteve um tempo total gasto de 38,38 horas, que perfaz uma diferença de 26,45 horas em comparação à tecnologia *Outsystems*.

Utilizando a equação do esforço total ($Esforço\,Total = UCP \times PF$) apresentada na revisão de literatura (ver secção 2.4.3) e considerando apenas os tempos de desenvolvimento dos casos de uso obtemos os seguintes valores para o *Productivity Factor (PF)*:

- $PF_{Outsystems} = 8{,}4/134 = 0{,}063$
- $PF_{Django/Python} = 22{,}8/134 = 0{,}17$

Considerando os tempos de configuração obtemos os seguintes *PF:*

- $PF_{Outsystems} = 11{,}93/134 = 0{,}089$
- $PF_{Django/Python} = 38{,}38/134 = 0{,}286$

Verifica-se que, considerando os tempos de configuração dos projetos, a diferença de produtividade (*PF*) ainda se agrava mais.

4.2.2 Fase II – Manutenção de *software*

Em conformidade com a primeira fase do desenvolvimento, nesta fase também foram apontados os tempos de desenvolvimento de cada caso de uso em cada tecnologia que se apresentam na Figura 4.39.

Outsystems	
Casos de Uso	Tempo real (minutos)
UC3.7. Criar Tarefa do Projeto	73
UC3.10. Editar Tarefa do Projeto	44
UC3.12. Consultar Indicadores do Projeto	13
UC3.13. Consultar Indicadores do Colaborador	63
Total	**193**
Em horas	**3,2**

Django/Phyton	
Casos de Uso	Tempo real (minutos)
UC3.7. Criar Tarefa do Projeto	271
UC3.10. Editar Tarefa do Projeto	116
UC3.12. Consultar Indicadores do Projeto	151
UC3.13. Consultar Indicadores do Colaborador	15
Total	**553**
Em horas	**9,2**

Figura 4.39. Tempo real de desenvolvimento por Caso de Uso – Fase II (Outsystems & Django/Phyton)

Observando o tempo total em horas de cada uma das tecnologias, conseguimos concluir que o tempo gasto no desenvolvimento em *Django/Phyton* é cerca de três vezes maior que o em *Outsystems*. O desenvolvimento aplicacional de *Outsystems* foi de 193 minutos / 3,2 horas, já em *Django/Phyton* foi de 553 minutos / 9,2 horas. Existe então uma

diferença de 6 horas entre cada desenvolvimento, provando uma vez mais favorece a tecnologia *low-code.*

Analisando agora cada caso de uso, é evidente que *Outsystems* obteve um tempo menor na maior parte. Analisando especificamente o caso de uso UC3.12. (Consultar Indicadores do Projeto) e UC3.13. (Consultar Indicadores do Colaborador), observamos que um deles, em ambas as tecnologias, apresenta um valor muito superior ao outro. Isto acontece porque, após um deles ser desenvolvido, dado que o outro requer o mesmo layout/funções, o código é aproveitado, facilitando bastante o desenvolvimento do segundo caso de uso.

Para além dos tempos representados na Figura 4.39, foi também despendido algum tempo para testes à aplicação, como se pode observar na Figura 4.40.

Outsystems	
Outras configurações/testes referentes ao projeto	Tempo real (minutos)
Testes aplicacionais	40
Total	**40**
Em horas	**0,67**
Total de Tempo Gasto (em horas)	**3,88**

Django/Phyton	
Outras configurações/testes referentes ao projeto	Tempo real (minutos)
Testes aplicacionais	20
Total	**20**
Em horas	**0,33**
Total de Tempo Gasto (em horas)	**9,55**

Figura 4.40. Tempo despendido noutras configurações/ações – Fase II (Outsystems & Django/Phyton)

Em suma, o total de tempo gasto é o somatório do tempo despendido no desenvolvimento de cada caso de uso mais o tempo gasto em testes aplicacionais. Dessa forma, o tempo total gasto na tecnologia *Outsystems* é de 3,88 horas e o de *Django/Phyton* é de 9,55 horas. Com isto, concluímos também que a diferença de desenvolvimento entre ambas as tecnologias é de exatamente 5,67 horas.

Utilizando a equação do esforço total ($Esforço\ Total = UCP \times PF$) apresentada na revisão de literatura (ver secção 2.4.4) e considerando apenas os tempos de desenvolvimento dos casos de uso obtemos os seguintes valores para o *Productivity Factor (PF)*:

- $PF_{Outsystems} = 3{,}2/49 = 0{,}065$
- $PF_{Django/Python} = 9{,}2/49 = 0{,}019$

Considerando os tempos de configuração obtemos os seguintes *PF:*

- $PF_{Outsystems} = 3{,}88/49 = 0{,}079$
- $PF_{Django/Python} = 9{,}55/49 = 0{,}195$

Na segunda parte da experiência, de manutenção do software desenvolvido, algumas das configurações necessárias ao início do projeto não foram necessárias pelo que a diferença de produtividade (*PF*) dos projetos com e sem os tempos de configuração é menor.

4.3 Discussão de resultados

A plataforma *Outsystems* utiliza o modelo *Platform as a Service* (PaaS), em português Plataforma como Serviço, para o desenvolvimento de aplicações (Sahay et al., 2020). As plataformas *PaaS* encontram-se na *Cloud* e oferecem um vasto leque de funcionalidades que beneficiam o desenvolvimento aplicacional. A redução do tempo gasto no desenvolvimento é uma dessas vantagens, como foi visível nos tempos apresentados na Figura 4.37 e Figura 4.39, pelo facto de, por exemplo, existem componentes aplicacionais pré-programados que são incorporados na plataformas (Azure, 2021). Segundo uma pesquisa realizada pela *Forrester*, as plataformas *low-code* aceleram o desenvolvimento cerca de cinco a dez vezes (Sanchis et al., 2020), sendo este um valor bastante significativo.

Outra vantagem da utilização desde modelo é a existência de uma base de dados integrada, que facilita a criação e gestão da mesma, pelo facto de todos os micro serviços serem criados e geridos sem intervenção do utilizador (Sahay et al., 2020). Em *Outsystems* a criação da base de dados, especialmente na primeira fase, foi bastante rápida e intuitiva, assim como a sua conexão ao projeto.

Vários *experts* na área de TI reconhecem que as plataformas *low-code* vão crescer nos próximos anos e destacam alguns dos seus principais benefícios como a privacidade, tendo em conta que as aplicações podem ser desenvolvidas por utilizadores sem conhecimento técnico aprofundado, a redução da complexidade e a fácil manutenção, sendo esta uma fase vital é necessária a rápida alteração do que já foi desenvolvido garantindo o alinhamento de todos os serviços aplicacionais como também dos requisitos de negócio (Sanchis et al., 2020), sem comprometer a qualidade dos artefactos (Varajão, 2021). Na segunda fase de desenvolvimento na plataforma *Outsystems* foi notória a rápida alteração de algumas funcionalidades (Figura 4.39) sem comprometer quaisquer outros serviços anteriormente desenvolvidos, nem a qualidade dos mesmos.

Dahlberg (2020), com base em entrevistas a profissionais de desenvolvimento de *software* onde foi abordado o tema produtividade no desenvolvimento *low-code*, conclui que os visados se sentiam mais produtivos ao utilizarem plataformas *low-code*. Alguns

desenvolvedores expressaram satisfação por serem capazes de produzirem resultados de forma mais rápida e eficiente (Dahlberg, 2020). Nesse estudo, foi também abordado a experiência dos desenvolvedores entre o desenvolvimento convencional *code based* e *low-code*. Alguns afirmaram que, com o desenvolvimento *low-code,* pode ser feito muito mais num dia do que com o desenvolvimento convencional o que indica que a produtividade no desenvolvimento *low-code* é bastante superior. Pode-se, dessa forma, comparar esses resultados com os da experiência realizada no presente trabalho, em que foram desenvolvidas as mesmas funcionalidades em duas tecnologias diferentes, uma convencional com *source code* (*Django/Python*) e outra com *low-code* (*Outsystems*). Também nesta experiência os tempos de desenvolvimento com *low-code* foram muito inferiores ao desenvolvimento em *source code*.

5 CONCLUSÃO

Neste capítulo apresentam-se as conclusões do trabalho realizado, com uma síntese do trabalho realizado, dos principais contributos, limitações e sugestões de trabalhos futuros.

5.1 Síntese do trabalho desenvolvido

Este trabalho teve como objetivo principal realizar um estudo comparativo de produtividade das tecnologias de desenvolvimento de *software*, s*ource code* ou *low-code*

Começou-se por fazer uma revisão de literatura, onde foram abordados métodos, tecnologias, produtividade e estimação de esforço no desenvolvimento de *software*, com o intuito de definir melhor de que forma poderíamos responder à questão de investigação inicialmente formulada.

De seguida, foi elaborado o protocolo da experiência laboratorial a realizar para responder à questão de investigação. A experiência laboratorial foi dividida em duas fases, que traduzem dois momentos distintos no ciclo de vida de desenvolvimento de software, importantes para a avaliação da produtividade no desenvolvimento de software com as diferentes tecnologias: implementação e manutenção de software. Assim, o protocolo da primeira fase consiste num SRS com os requisitos funcionais para o desenvolvimento da aplicação (ver apêndice 1), enquanto o protocolo da segunda fase contém os requisitos das alterações e novas funcionalidades a implementar (ver apêndice 2). Neste protocolo, foi também apresentada uma estimativa para o esforço de desenvolvimento da aplicação, utilizando o método de análise de pontos de caso de uso.

Após o desenvolvimento do protocolo, seguiu-se o desenvolvimento da aplicação nos dois tipos de tecnologia: *source code*, com a utilização do *Django/Python;* e *low-code*, com a utilização do *Outsystems*. Foram registados os tempos despendidos no desenvolvimento de cada caso de uso. Com base nestes tempos foi avaliada a produtividade no desenvolvimento de *software*. Por fim, foram discutidos esses resultados tendo como base alguns dos benefícios esperadas no desenvolvimento *low-code*.

5.2 Principais contributos

Como principais contributos deste trabalho tem-se a revisão de literatura efetuada, em que se apresentou e comparou as diferentes metodologias (ágeis e tradicionais) e

tecnologias (*source code, low-code e no-code*), apresentando as suas vantagens e desvantagens.

Sendo a produtividade um ponto fulcral neste trabalho, abordou-se também na revisão de literatura este aspeto, tendo-se identificado alguns dos fatores que influenciam o desenvolvimento de software ligados a pessoas, produtos e organizações, e analisando-se as diferentes metodologias de estimação de esforço de desenvolvimento de software, das quais se escolheu a de análise de pontos de caso de uso.

Outro contributo foi a elaboração do protocolo da experiência laboratorial, dividido em duas fases, que pretendem caracterizar o desenvolvimento e manutenção de software respetivamente. Esta sistematização permite a que outros no futuro possam desenvolver a mesma experiência e medir os resultados. Não foi encontrada na literatura uma abordagem semelhante.

Como contributo final, e mais importante, os resultados obtidos com a experiência laboratorial realizada permitiram concluir que a produtividade com recurso à utilização de tecnologias *low-code* é superior à obtida com as tecnologias *source code*. Na literatura esta diferença já é referida, mas tipicamente apenas em publicações de natureza comercial, notando-se uma escassez significativa de estudos científicos e independentes sobre este fenómeno. Na primeira fase, o desenvolvimento em *Outsystems* obteve uma produtividade bastante superior, revelando-se, cerca de três vezes mais rápido que o desenvolvimento em *Django/Python*. O mesmo aconteceu na segunda fase.

5.3 Limitações

Não obstante, a pertinência e o rigor dos resultados obtidos, identificam-se algumas limitações neste trabalho. A primeira prende-se com o protocolo das experiências, o qual foi definido tendo por referência características típicas das aplicações de gestão. Para tipos de aplicações distintos, obrigando a interfaces mais elaborados, a produtividade pode ser distinta, mas é algo que só em trabalhos futuros ou em novas experiências será possível verificar efetivamente. A segunda limitação, prende-se com o perfil dos programadores envolvidos na experiência. Apesar de se ter procurado envolver programadores com elevada experiência nas tecnologias selecionadas, os diferentes perfis podem causar algum enviesamento nos resultados. Finalmente, nos cálculos de produtividade (FP) deve ser considerada a qualidade das aplicações desenvolvidas, algo que para efeitos de simplificação não foi considerado neste trabalho.

5.4 Trabalhos futuros

O tema da produtividade de base científica e software é um tema atual e alvo de vários estudos. No entanto, não se encontram estudos que se debrucem sobre a questão de investigação apresentada neste relatório pelo que, em termos de trabalhos futuros, propõe-se a realização de experiências similares com outras tecnologias (*source code, low-code* ou *no-code*) por forma a avaliar a produtividade na realização das diferentes atividades relativas à implementação, teste e manutenção de software.

Este trabalho corrobora a ideia original dos autores de que a tecnologia *low-code* é mais produtiva que a tecnologia *source code*, pelo que as organizações que desenvolvem *software* devem ter em conta este tipo de tecnologia.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2017.11.045

Alahyari, H., Berntsson Svensson, R., & Gorschek, T. (2017). A study of value in agile software development organizations. *Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2016.12.007

Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *International Journal of Computer Science Issues*, *12*(1), 106–111.

Alvarenga, E. M. de. (2012). *Metodologia da Investigação - quantitativa e qualitativa*.

Arlow, J., & Neustadt, I. (2002). *UML and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley.

Azure, M. (2021). *O que é PaaS?*

Balijepally, Mahapatra, Nerur, & Price. (2009). Are Two Heads Better than One for Software Development? The Productivity Paradox of Pair Programming. *MIS Quarterly*, *33*(1), 91. https://doi.org/10.2307/20650280

Boehm, B. (2007). A Spiral model of software development and enhancement. In *Software Management, Seventh Edition*. https://doi.org/10.1109/9780470049167.ch2

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61–72. https://doi.org/10.1109/2.59

Bubble. (2021). *Bubble*.

Canedo, E. D., & Santos, G. A. (2019). Factors affecting software development productivity: An empirical study. *ACM International Conference Proceeding Series*, 307–316. https://doi.org/10.1145/3350768.3352491

Carroll, E. R. (2005). *Estimating software based on use case points*. https://doi.org/10.1145/1094855.1094960

Chowdary, V., & Krishna, V. (2016). Software Effort Estimation: A Comparative Analysis. *International Journal of Progressive Sciences and Technologies*, *2*(2), 48–

60.

Clemmons, R. K. (2006). Project estimation with use case points. *The Journal of Defense Software Engineering*, *19*(2), 18–22.

Dahlberg, D. (2020). *Developer Experience of a Low-Code Platform: An exploratory study*.

Ferrucci, F., Gravino, C., & Lavazza, L. (2016). Simple function points for effort estimation: A further assessment. *Proceedings of the ACM Symposium on Applied Computing*, *04-08-Apri*, 1428–1433. https://doi.org/10.1145/2851613.2851779

FRED, F. R. B. of S. L. (2021). *U.S. Bureau of Labor Statistics, All Employees, Computer Systems Design and Related Services [CES6054150001]*. https://fred.stlouisfed.org/series/CES6054150001

Hjalmarsson, A., Korman, M., & Lagerström, R. (2013). Software migration project cost estimation using COCOMO II and enterprise architecture modeling. *CEUR Workshop Proceedings*, *1023*, 39–48.

Ifpug. (2010). *Manual de Práticas de Contagem de Pontos de Função*.

Laranjeira, M., Trigo, A., & Varajão, J. (2019). Success of Software Development Projects in Portugal-preliminary results. *CAPSI 2019 Proceedings*. https://aisel.aisnet.org/capsi2019/45

Lavazza, L., Morasca, S., & Tosi, D. (2018). An empirical study on the factors affecting software development productivity. *E-Informatica Software Engineering Journal*, *12*(1), 27–49. https://doi.org/10.5277/e-Inf180102

Leau, Y., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches*. *37*(Icint), 162–167.

Lokan, C. J. (2005). Function Points. In *Advances in Computers* (Vol. 65, pp. 297–347). https://doi.org/10.1016/S0065-2458(05)65007-3

McLeod, S. A. (2012). Experimental method. *Simply Psychology*.

Mendix. (2021). *What Is Mendix?*

Nageswaran, S. (2001). *Quality Week*.

Ochodek, M., Nawrocki, J., & Kwarciak, K. (2010). *Simplifying effort estimation based on Use Case Points q*. https://doi.org/10.1016/j.infsof.2010.10.005

Oliveira, E. C. C. de. (2017). *Fatores de influência da produtividade dos desenvolvedores de organizações de software*. 182.

Outsystems. (2021). *Outsystems*.

Pimparkhede, K. (2018). Software Development Life Cycle. In *Computer Programming with C++*. https://doi.org/10.1017/9781316534489.021

Pressman, R. S. (2009). Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman. In *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*. https://doi.org/10.1017/CBO9781107415324.004

Python.org. (2021). *What is Python? Executive Summary*.

Rambe, B. H., Pane, R., Irmayani, D., Nasution, M., Munthe, I. R., Ekonomi, F., & Bisnis, D. (2020). UML Modeling and Black Box Testing Methods in the School Payment Information System. In *Jurnal Mantik*.

Richardson, C., & Rymer, J. R. (2016). Vendor landscape: The fractured, fertile terrain of low-code application platforms. In *FORRESTER, Janeiro*. https://informationsecurity.report/Resources/Whitepapers/0eb07c59-b01c-4399-9022-dfc297487060_Forrester Vendor Landscape The Fractured, Fertile Terrain.pdf

Sadowski, C., & Zimmermann, T. (2019). Rethinking Productivity in Software Engineering. In C. Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering*. Apress. https://doi.org/10.1007/978-1-4842-4221-6

Sahay, A., Indamutsa, A., Di Ruscio, D., Pierantonio, A., Ruscio, D. Di, & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences (Switzerland)*, *10*(1). https://doi.org/10.3390/app10010012

Sanner, M. F. (1999). Python: A programming language for software integration and development. In *Journal of Molecular Graphics and Modelling*.

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide: The Definitive The Rules of the Game. *Scrum.Org and ScrumInc*.

Shylesh, S. (2017). A Study of Software Development Life Cycle Process Models. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2988291

Singal, P., Kumari, A. C., & Sharma, P. (2020). ScienceDirect ScienceDirect Estimation of Software Development Effort : A Differential Estimation of Software Development Effort : A Differential Evolution Approach Evolution Approach. *Procedia Computer Science*, *167*(2019), 2643–2652. https://doi.org/10.1016/j.procs.2020.03.343

Sommerville, I. (2011). Engenharia de Software. In *Pearson Brasil*.

Sousa, A., Machado, R. J., & Ribeiro, P. (2012). Estimação do esforço de desenvolvimento de um sistema de software com Use Case Points : Análise de um Caso de Aplicação Resumo. *12.ª Conferência Da Associação Portuguesa de Sistemas de Informação (CAPSI'2012)*, *1*, 175–185. http://capsi.apsi.pt/index.php/capsi/article/view/67

Sousa, L., Trigo, A., & Varajão, J. (2019). DevOps – Foundations and perspectives. *Atas Da Conferencia Da Associacao Portuguesa de Sistemas de Informacao*.

STH. (2019). *Agile vs. Waterfall: Which is the Best Methodology for Your Project?* Software Testing Help. https://www.softwaretestinghelp.com/agile-vs-waterfall/

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

Varajão, J., Trigo, A., & Barroso, J. (2009). Motivations and trends for IT/IS adoption: Insights from Portuguese companies. In *Social, Managerial, and Organizational Dimensions of Enterprise Information Systems*. https://doi.org/10.4018/978-1-60566-856-7.ch024

Varajao, J., Trigo, A., Figueiredo, N., Barroso, J., Cruz, J. B., Varajão, J., Trigo, A., Figueiredo, N., Barroso, J., & Bulas-Cruz, J. (2009). Information systems services outsourcing reality in large Portuguese organisations. *International Journal of Business Information Systems*, *4*(1), 125. https://doi.org/10.1504/IJBIS.2009.021606

Varajão, João. (2021). Software Development in Disruptive Times. *Queue*, *19*(1), 94–103. https://doi.org/10.1145/3454122.3458743

Varajão, João, Trigo, A., & Barroso, J. (2009). Motivations and Trends for IT/IS Adoption. *International Journal of Enterprise Information Systems*, *5*(4), 34–52. https://doi.org/10.4018/jeis.2009090203

Vazquez, C. E., Simões, G. S., & Albert, R. M. (2013). *Análise de pontos de função: Medição, estimativas e gerenciamento de projetos de software* (23ª). Editora Érica.

Vincent, P., Natis, Y., Iijima, K., Wong, J., Ray, S., Jain, A., & Leow, A. (2020). *The 2020 Gartner Magic Quadrant for Enterprise Low-Code Application Platforms*.

Wagner, S., & Ruhe, M. (2018). *A Systematic Review of Productivity Factors in Software Development*.

Wenz, A. (2021). Do Distractions During Web Survey Completion Affect Data Quality? Findings From a Laboratory Experiment. *Social Science Computer Review*, *39*(1), 148–161. https://doi.org/10.1177/0894439319851503

APÊNDICES

APÊNDICE 1. Protocolo da experiência laboratorial – Fase I – Desenvolvimento de *software*

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE CONTABILIDADE E ADMINISTRAÇÃO DE COIMBRA

MESTRADO EM

SISTEMAS DE INFORMAÇÃO DE GESTÃO

Protocolo para a Realização de Experiência Laboratorial

MESTRANDO:

Nome: Miguel Alves Almeida N.º 17945 (IPC-ISCAC)

ORIENTADORES:

Prof. Doutor António Trigo (IPC-ISCAC)

Prof. Doutor João Varajão (UMINHO)

Índice Geral

Índice de Tabelas

Índice de Figuras

1 Sumário

A aplicação informática a desenvolver, denominada *Easy Management*, tem por objetivo suportar a gestão das atividades de um projeto, desde a sua criação até à finalização.

Esta aplicação inclui a gestão de Utilizadores, Projetos e Tarefas. A aplicação foi projetada para funcionar em *Web* e deverá ser compatível com diferentes *browsers*.

A aplicação contempla três perfis distintos de utilizadores, que têm acesso a diferentes funcionalidades na aplicação:

- O **Administrador** tem, somente, a possibilidade de gerir os utilizadores da plataforma;
- O **Gestor de Projetos**, pode criar Projetos e as respetivas Tarefas. Com este perfil, também é possível listar os (próprios) Projetos e Tarefas, sendo ainda possível editá-los e arquivá-los;
- No que respeita ao **Colaborador**, este tem a possibilidade de consultar e editar as suas Tarefas, as quais são previamente atribuídas pelo **Gestor de Projetos**. O **Colaborador** tem, também, a possibilidade de visualizar os Projetos que estão associados às suas Tarefas.

O sistema é composto por três módulos que agrupam as funcionalidades do sistema, encontrando-se identificados na Figura *1*: Módulo Autenticação (P1); Módulo Gestão de Utilizadores (P2); Módulo Gestão de Projetos e Tarefas (P3).

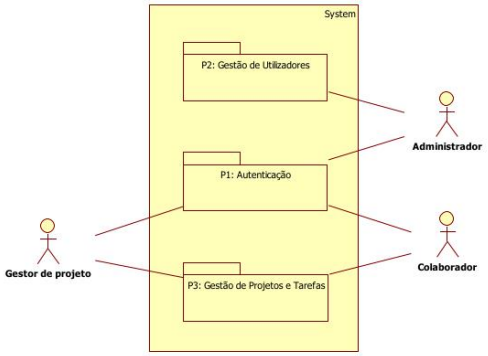


Figura 1. *Módulos do Sistema*

6

2 Modelo de Casos de Uso

2.1 Módulo Autenticação

A Figura 3 apresenta os casos de uso referentes ao módulo Autenticação (*package* P1).

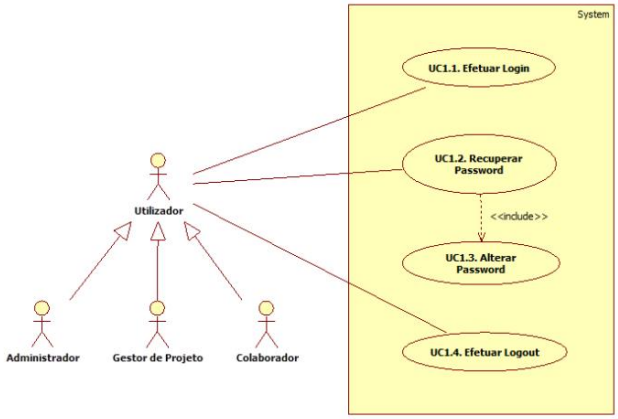


Figura 2. Diagrama de Casos de Uso (Autenticação)

2.2 Módulo Gestão de Utilizadores

A Figura 3 apresenta os casos de uso referentes ao módulo Gestão de Utilizadores (*package* P2).

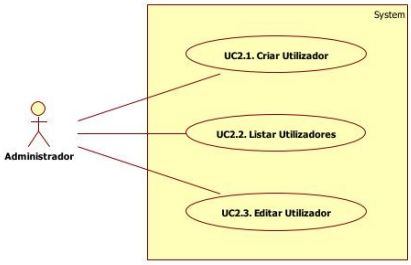


Figura 3. Diagrama de Casos de Uso (Gestão de Utilizadores)

2.4 Módulo Gestão de Projetos e Tarefas

A Figura 4 apresenta os casos de uso refentes ao módulo Gestão de Projetos e Tarefas (*package* P3).

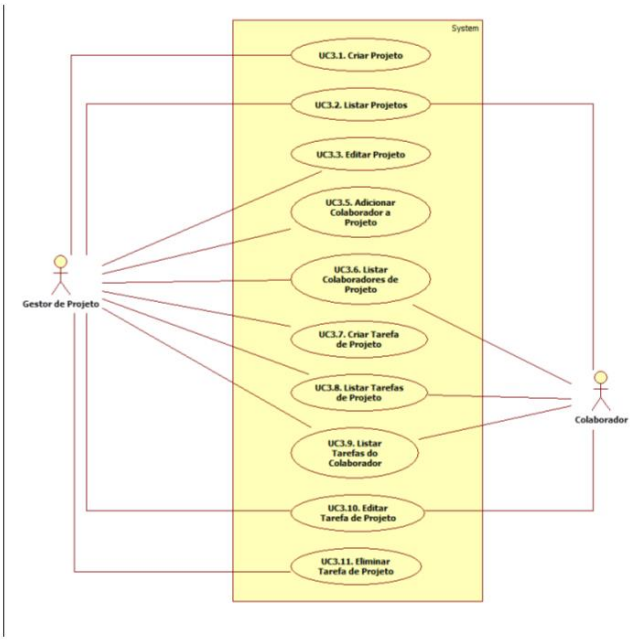


Figura 4. Diagrama de Casos de Uso (Gestão de Projetos e Tarefas)

8

3 Especificação dos casos de uso

3.1 Módulo: Autenticação

3.1.1 Efetuar Login

O formulário da Figura 5 deve ser o primeiro a ser apresentado (quando um utilizador acede à aplicação). Neste formulário, o utilizador deverá inserir as suas credenciais de acesso. De notar que os utilizadores são criados pelo Administrador, não havendo uma funcionalidade para se auto registarem. A especificação do caso de uso encontra-se na Tabela 1.

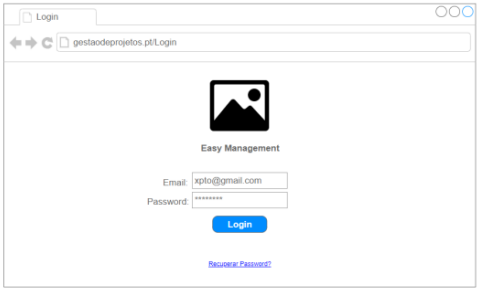


Figura 5. Página de Login

Tabela 1. Efetuar Login

UC1.1	**Efetuar Login**
Descrição: Este caso de uso permite que um utilizador se autentique na aplicação.	
Pré-Condições: -	
Fluxo Primário: 1. É apresentado o formulário de *Login*. 2. O utilizador insere as suas credenciais. 3. O utilizador clica no botão "*Login*". A1. O utilizador insere as permissões erradas. A2. O utilizador está não está "Ativo"	
Fluxos Secundários: A1. O utilizador insere as permissões erradas. 1. É apresentada a mensagem de erro "Email ou password inválidos". 2. O fluxo volta ao passo 2 do fluxo primário. A2. O utilizador não está "Ativo" 1. É apresentada a mensagem de erro "Utilizador inativo". 2. O fluxo volta ao passo 2 do fluxo primário.	
Exceções: -	
Pós-Condições: O utilizador fica autenticado e é atualizado o seu registo de utilizador (entidade "Utilizador", coluna "ultimoLogin" = data/hora de login).	

9

3.1.2 Recuperar *Password*

Caso o utilizador se esqueça das credenciais, terá a possibilidade de recuperar a sua *password*. Para isso, terá de clicar no link "Recuperar Password" que se encontra na mesma página do *Login* (ver Figura 5) e, em seguida, irá ser encaminhado para o formulário apresentado na Figura 6. Após inserir o seu endereço de e-mail e submeter o formulário, irá receber um e-mail com instruções e um link para efetuar a recuperação da *password*. A especificação do caso de uso encontra-se na Tabela 2.

Figura 6. Página Recuperar Password

Tabela 2. Recuperar Password

UC1.2	**Recuperar Password**
Descrição: Este caso de uso permite ao utilizador recuperar a sua *password* de acesso ao sistema.	
Pré-Condições: O utilizador encontra-se na página de *Login*.	
Fluxo Primário: 1. Na página de *Login*, o utilizador seleciona a opção "Recuperar *password*". 2. É apresentado o formulário de recuperação de *password*. 3. O utilizador insere o seu email. 4. O utilizador clica no botão "Enviar". A1. O utilizador engana-se no email. 5. O utilizador recebe um email para a recuperação da *password*. 6. O utilizador clica no *link* apresentado no email, o qual respeita a um formulário que permite a definição de uma nova *password*. A2. O utilizador não recebe o email ou não clica no *link* apresentado. 7. INCLUDE (UC1.3. Alterar Password)	
Fluxos Secundários: A1. O utilizador engana-se no email. 1. É apresentada a mensagem de erro "Email inválido". 2. O fluxo volta ao passo 3 do fluxo primário. A2. O utilizador não recebe o email ou não clica no *link* apresentado. 1. A password não é alterada.	
Exceções: -	
Pós-Condições: O utilizador fica com uma nova password definida.	

10

3.1.3 Alterar *Password*

Caso o utilizador pretenda alterar a sua *password*, terá de selecionar a opção "Recuperar password" (ver secção 3.1.2), sendo na sequência enviada uma mensagem por *e-mail* com um *link* para o formulário apresentado na Figura 7. A especificação do caso de uso encontra-se na Tabela 3.

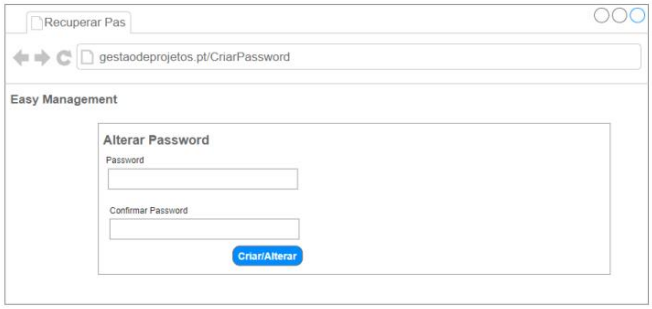


Figura 7. Página Alterar Password

Tabela 3. Alterar Password

UC1.3	**Alterar *Password***
Descrição: Este caso de uso permite a um utilizador alterar a sua *password*.	
Pré-Condições: O utilizador tem de estar autenticado e ter selecionado previamente a opção "Recuperação de *Password*".	
Fluxo Primário: 1. O utilizador clica no *link* de recuperação de *password*, que recebeu por *e-mail*. 2. É apresentado o formulário para alteração da *password*. 3. O utilizador introduz a nova *password*, nos campos "Password" e "Confirmar Password". 4. O utilizador clica no botão "Criar/Alterar". A1. O utilizador insere *passwords* diferentes nos campos.	
Fluxos Secundários: A1. O utilizador insere *passwords* diferentes nos campos. 1. É apresentada a mensagem de erro "As *passwords* inseridas não são iguais". 2. O fluxo volta ao passo 3 do fluxo primário.	
Exceções: -	
Pós-Condições: A *password* do utilizador é alterada.	

11

3.1.4 Efetuar *Logout*

Para que seja possível ao utilizador terminar uma sessão ativa na aplicação, terá disponível um ícone referente ao *Logout* (este deverá estar localizado no canto superior direito da barra de tarefas). Este ícone estará presente em todas as páginas, com a exceção da página de *Login*. A especificação do caso de uso encontra-se na Tabela 4.

Figura 8. Ícone de Logout

Tabela 4. Efetuar Logout

UC1.4	**Efetuar *Logout***
Descrição: Este caso de uso permite ao utilizador encerrar a sua sessão na aplicação.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador clica no botão *Logout*. 2. É terminada a autenticação na aplicação.	
Exceções: -	
Pós-Condições: A sessão do utilizador é encerrada.	

3.2 Módulo: Gestão de Utilizadores

Apenas o Administrador da aplicação terá acesso ao módulo de Gestão de Utilizadores.

3.2.1 Listar Utilizadores

Para aceder à listagem de utilizadores, o administrador terá de clicar no separador "Utilizadores" presente na barra de tarefas. De seguida, será direcionado para a página que apresenta a lista dos utilizadores registados na aplicação, que se apresenta na Figura 9. A especificação do caso de uso encontra-se na Tabela 5.

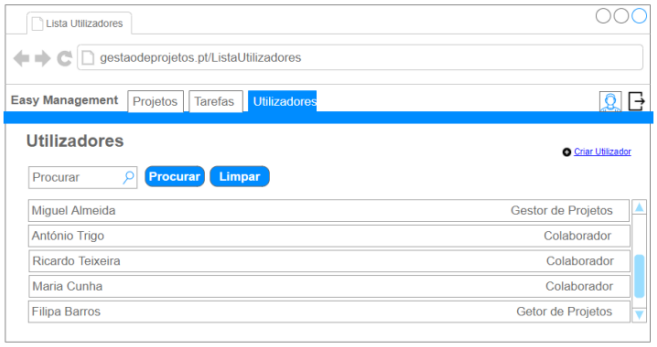


Figura 9. Página Listar Utilizadores

Tabela 5. Listar Utilizadores

UC2.2	**Listar Utilizadores**
Descrição: Este caso de uso permite a visualização de todos os utilizadores registados na aplicação.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Administrador.	
Fluxo Primário: 1. O utilizador (Administrador) acede ao separador "Utilizadores". 2. É apresentada a lista de utilizadores.	
Exceções: -	
Pós-Condições: -	

3.2.2 Criar Utilizador

Para criar um novo utilizador, o administrador terá primeiro de clicar no separador "Utilizadores", sendo então direcionado para a listagem de Utilizadores. De seguida, terá de clicar no *link* "Criar Utilizador". Posteriormente, será direcionado para o formulário apresentado na Figura 10. Neste formulário, o administrador insere o nome e email do novo utilizador. Pode, ainda, indicar se o mesmo tem permissões para criar novos projetos (*checkbox* Criar Projetos) e se está ativo para aceder à aplicação (*checkbox* Ativo). A especificação do caso de uso encontra-se na Tabela 6.

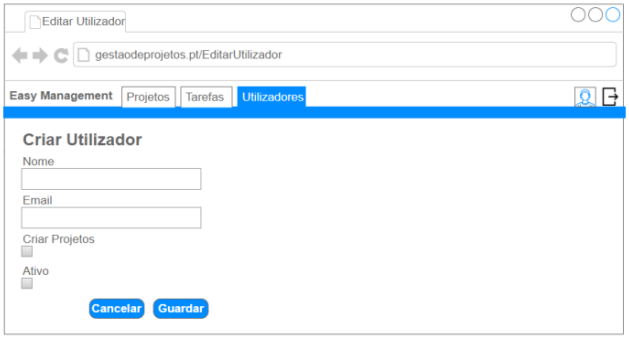


Figura 10. Página Criar Utilizador

Tabela 6. Criar Utilizador

UC2.1	**Criar Utilizador**
Descrição: Este caso de uso permite criar um utilizador.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Administrador.	
Fluxo Primário: 1. O utilizador (Administrador) acede ao separador "Utilizadores", onde seleciona a opção "Criar Utilizador". 2. É apresentado um formulário para a criação de novo utilizador. 3. O utilizador (Administrador) preenche o formulário. 4. O utilizador (Administrador) clica no botão "Guardar". A1. O utilizador (Administrador) clica no botão "Cancelar". A2. O formulário contém erros.	
Fluxos Secundários: A1. O utilizador (Administrador) clica no botão "Cancelar". 1. É fechado o formulário referente à criação de utilizador e apresentada a página listar utilizadores (sem gravar os dados do formulário). A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário (ex. "nome não preenchido", "nome já existente", "email não preenchido", "email já existente", "email inválido"). **2.** O fluxo volta ao passo 3 do fluxo primário.	
Exceções: E1. O utilizador (Administrador) pode abandonar o formulário sem registar o novo utilizador.	
Pós-Condições: É registado o novo utilizador. É apresentada a página listar utilizadores.	

14

3.2.3 Editar Utilizador

Para editar os dados de um utilizador já existente, o administrador terá primeiro de clicar no separador "Utilizadores", sendo então direcionado para a listagem de Utilizadores. De seguida, terá de clicar no nome do utilizador que pretende editar. Posteriormente, será direcionado para o formulário apresentado na Figura 11. A especificação do caso de uso encontra-se na Tabela 6.

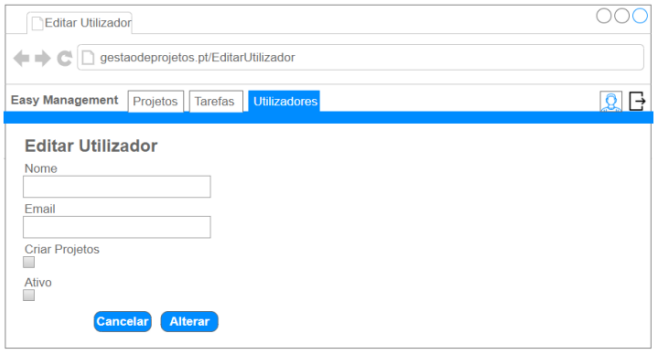


Figura 11. Página Editar Utilizador

Tabela 7. Editar Utilizador

UC2.3	**Editar Utilizador**
Descrição: Este caso de uso permite editar os dados de um utilizador.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Administrador.	
Fluxo Primário: 1. O utilizador (Administrador) acede ao separador "Utilizadores", onde seleciona o utilizador que pretende editar clicando para tal no respetivo nome. 2. É apresentado um formulário para a edição dos dados do utilizador, pré-preenchido com os dados que se encontram registados na aplicação. 3. O utilizador (Administrador) faz as alterações que pretende. 4. O utilizador (Administrador) clica no botão "Alterar". A1. O utilizador (Administrador) clica no botão "Cancelar". A2. O formulário contém erros.	
Fluxos Secundários: A1. O utilizador (Administrador) clica no botão "Cancelar". 1. É fechado o formulário referente à edição de utilizador e apresentada a página listar utilizadores (sem gravar os dados do formulário).	

15

A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário (ex. "nome não preenchido", "nome já existente", "email não preenchido", "email já existente", "email inválido"). 2. O fluxo volta ao passo 3 do fluxo primário.
Exceções: -
Pós-Condições: Os dados do utilizador são alterados. É apresentada a página listar utilizadores.

3.3 Módulo: Gestão de Projetos (Projetos)

3.3.1 Listar Projetos

Para aceder à listagem dos projetos, o utilizador necessita de clicar no separador "Projetos" presente na barra de tarefas, sendo assim direcionado para a página apresentada na Figura 12. Para efetuar uma pesquisa na lista de projetos, o utilizador deve inserir uma expressão (referente a pelo menos parte do nome do projeto que pretende encontrar) na caixa de texto "Procurar" e clicar no botão "Procurar", sendo então apresentados apenas os projetos que contenham no seu nome a expressão inserida. O botão "Limpar" elimina o texto inserido na caixa de texto "Procurar", sendo listados todos os projetos novamente. A especificação do caso de uso encontra-se na Tabela 8.

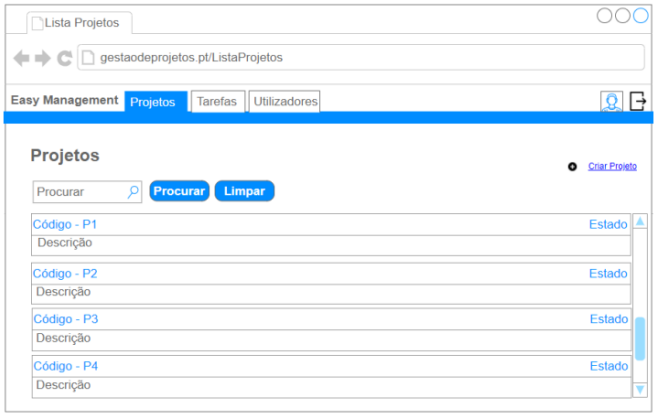


Figura 12. Página Listar Projetos

Tabela 8. Listar Projetos

UC3.2	**Listar Projetos**
Descrição: Este caso de uso permite listar e pesquisar os projetos existentes na aplicação associados ao utilizador.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Projetos". A1. Filtrar projetos. 2. É apresentada a listagem dos projetos associados ao utilizador.	
Fluxos Secundários: A1. Filtrar projetos 1. O utilizador insere uma expressão referente a pelo menos parte do nome do projeto, na caixa de texto "Procurar". 2. O utilizador clica no botão "Procurar". 3. É aplicada a expressão inserida como filtro na listagem de projetos. 4. O fluxo volta ao passo 2 do fluxo primário.	
Exceções: -	
Pós-Condições: -	

3.3.2 Criar Projeto

Para criar um novo projeto, cujo formulário se encontra na Figura 13, o utilizador deverá clicar no link "Criar projeto" disponibilizado na página "Listar Projetos" (ver Figura 12). Para criar um projeto, o utilizador necessita de inserir o código (identificador) e o nome do projeto, uma descrição, as datas de início e término do projeto, o estado do projeto (que pode assumir os valores: "Por fazer", "Em Andamento", "Pendente" e "Finalizado"), o esforço estimado (em minutos) e o esforço efetivo do projeto (em minutos). A especificação do caso de uso encontra-se na Tabela 9.

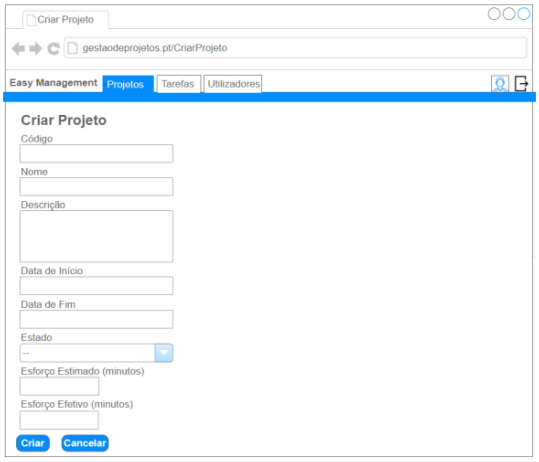


Figura 13. Página Criar Projeto

Tabela 9. Criar Projeto

UC3.1	**Criar Projeto**
Descrição: Este caso de uso permite criar um novo projeto.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Gestor de Projeto.	
Fluxo Primário: 1. O utilizador acede ao separador "Projetos". 2. Clica em "Criar Projeto". 3. É apresentado um formulário para preenchimento. 4. O utilizador preenche e submete o formulário. A1. O utilizador clica no botão "Cancelar". A2. O formulário contém erros.	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar projetos (sem gravar os dados do formulário). A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário (ex. "código não preenchido", "código já existente", "nome não preenchido", "nome já existente"). Todos os campos à exceção da descrição e do esforço efetivo são obrigatórios. 2. O fluxo volta ao passo 4 do fluxo primário.	
Exceções: -	
Pós-Condições: É registado o novo projeto. É apresentada a página listar projetos.	

18

3.3.3 Editar Projeto

Para aceder à página de edição de projeto (ver Figura 14) o utilizador terá de aceder primeiro à página de listagem dos projetos (ver Figura 12) e selecionar o projeto que quer editar. Esta página possui duas áreas distintas: a primeira, do lado esquerdo, que lista todos os projetos a que o utilizador está associado, permitindo ao utilizador selecionar o projeto deseja visualizar (a opção gráfica foi a *checkbox* mas pode-se utilizar outra que identifique o projeto selecionado como *radiobuttons* ou destacar o projeto selecionado colorindo o nome do mesmo); e a segunda, do lado direito da primeira, que possui três separadores, que permitem o acesso a três áreas distintas da aplicação: informações, tarefas e colaboradores.

No separador "Informações", é apresentado um formulário que permite visualizar/editar os detalhes do projeto, com um botão no canto superior direito com o ícone "Disquete", que permite guardar as alterações efetuadas ao projeto. A especificação do caso de uso encontra-se na Tabela 10.

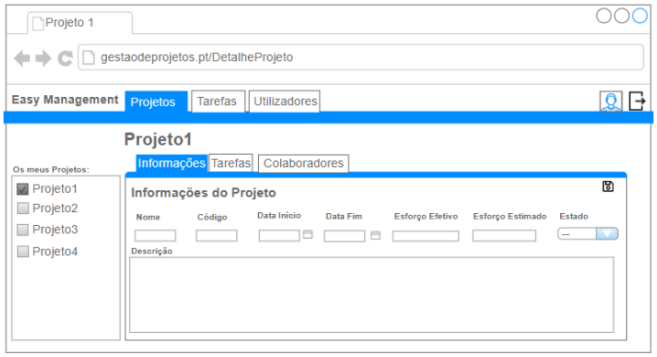


Figura 14. Página Detalhe do Projeto, separador Informações

Tabela 10. Editar Projeto

UC3.3	**Editar Projeto**
Descrição: Este caso de uso permite alterar os dados de um projeto.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Gestor de Projeto.	
Fluxo Primário: 1. O utilizador acede ao separador "Projetos". 2. O utilizador seleciona o projeto que pretende editar. 3. O utilizador carrega no botão Editar. 4. O sistema disponibiliza o formulário para edição. 5. O utilizador faz as alterações pretendidas. 6. O utilizador clica no botão guardar. A1. O formulário contém erros. A2. Guarda formulário com o estado "Arquivar".	
Fluxos Secundários: A1. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário. 2. O fluxo volta ao passo 5 do fluxo primário. A2. Guarda formulário com o estado "Arquivar". 1. É apresentada uma questão ao utilizador questionado se tem a certeza de que quer guardar esta alteração no projeto. 2. Se confirmar guarda as alterações ao projeto. 3. Se não confirmar cancela alterações e volta ao passo 4.	
Exceções: -	
Pós-Condições: Os dados do projeto são alterados. É apresentada a página listar projetos.	

3.3.4 Listar Tarefas do Projeto

Clicando no separador relativo às tarefas, o utilizador acede à lista de tarefas associadas ao projeto selecionado em "Os meus projetos" (ver Figura 15). A partir desta lista, o utilizador tem as opções de criar tarefa (link no canto superior direito) e editar tarefa, clicando na tarefa desejada. A especificação do caso de uso encontra-se na Tabela 11.

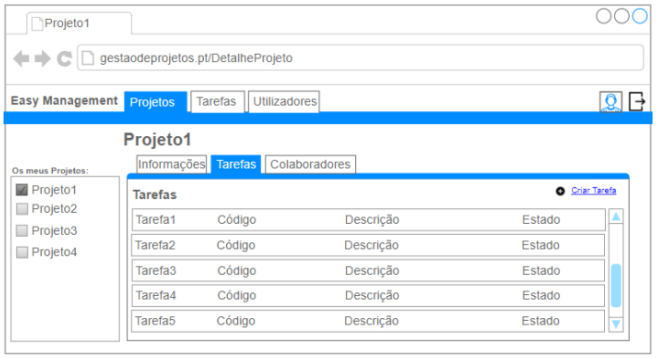


Figura 15. Página Detalhe do Projeto, separador Tarefas

Tabela 11. Listar Tarefas do Projeto

UC3.8	**Listar Tarefas do Projeto**
Descrição: Este caso de uso permite listar as tarefas existentes na aplicação associadas ao projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Tarefas". 2. Se o utilizador for o Gestor de Projetos: 2.1. É apresentada a lista das tarefas associadas ao projeto. 3. Caso contrário (colaborador): 3.1. É apresentada a lista das tarefas do utilizador associadas ao projeto.	
Fluxos Secundários:	
Exceções:	
Pós-Condições:	

3.3.5 Criar Tarefa do Projeto

Para aceder à funcionalidade Criar Tarefa, o utilizador deverá clicar no botão que se encontra no canto superior do lado direito da página Listar Tarefas (ver Figura 15). Será então direcionado para um formulário que permite a criação de tarefas. Esse formulário possui vários campos, nomeadamente, código, nome, estado ("Por fazer", "Em Andamento", "Pendente" e "Finalizado"), o colaborador que irá realizar a tarefa, uma breve descrição, a data de início e data de fim prevista de conclusão da tarefa, e ainda os esforços estimado e efetivo de execução da tarefa (em minutos). Alguns dos valores

destes campos, como, por exemplo, o estado da tarefa, serão posteriormente atualizados durante a execução da tarefa. A especificação do caso de uso encontra-se na Tabela 12.

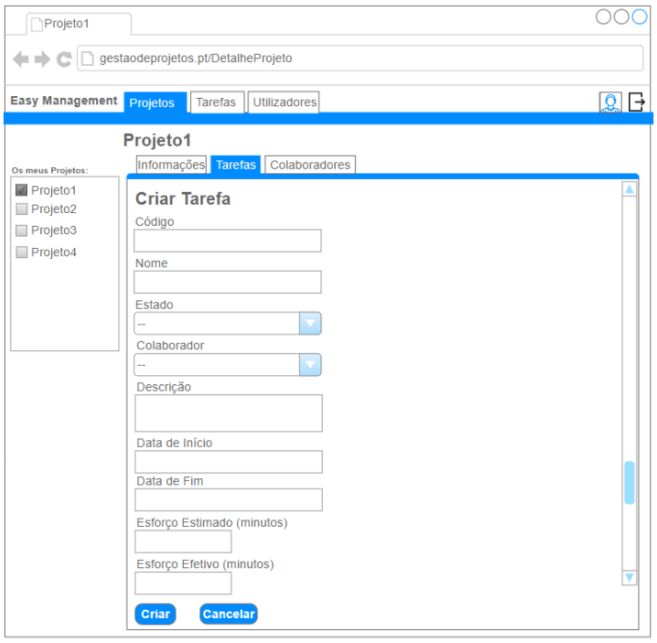


Figura 16. Página Criar Tarefa do Projeto

Tabela 12. Criar Tarefa do Projeto

UC3.7	**Criar Tarefa do Projeto**
Descrição: Este caso de uso permite criar uma tarefa associada a um determinado projeto.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissões de Gestor de projeto.	
Fluxo Primário: 1. Clica em "Criar Tarefa". 2. É apresentado um formulário para preenchimento. 3. O utilizador preenche e submete o formulário. A1. O utilizador clica no botão cancelar. A2. O formulário contém erros.	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar tarefas (sem gravar os dados do formulário). A2. O formulário contém erros.	

22

1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário. 2. O fluxo volta ao passo 3 do fluxo primário.
Exceções:
Pós-Condições: É registado uma nova tarefa. É apresentada a página listar tarefas.

3.3.6 Editar Tarefa do Projeto

Para aceder à funcionalidade de Editar Tarefa, o utilizador deverá selecionar, na lista de tarefas, a tarefa desejada, sendo então redirecionado para a página de visualização/edição de tarefa. Após alterar os campos desejados, o utilizador deverá clicar no botão guardar para submeter as alterações efetuadas. Para além da edição da tarefa o utilizador pode eliminar uma tarefa clicando no botão com o ícone de "Caixote do Lixo" disponível no canto superior direito. A especificação do caso de uso encontra-se na Tabela 13.

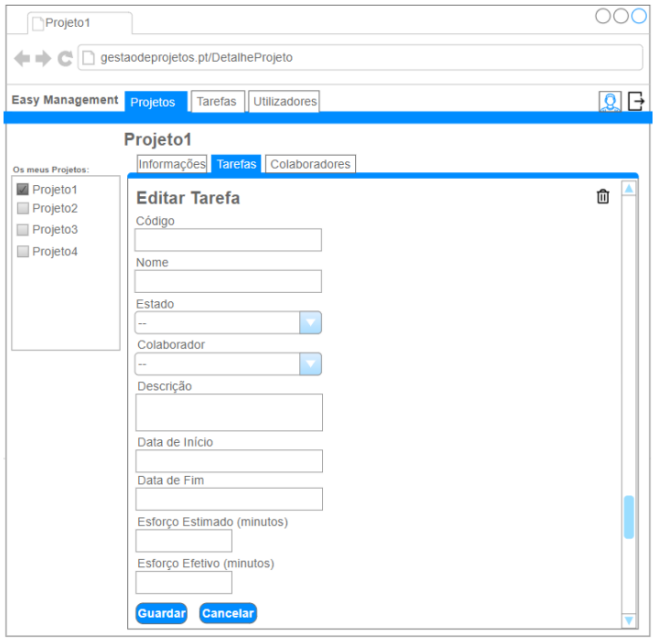


Figura 17. Página Editar Tarefa do Projeto

23

Tabela 13. Editar Tarefa do Projeto

UC3.10	**Editar Tarefa do Projeto**
Descrição: Este caso de uso permite editar uma tarefa associada a um determinado projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador seleciona, a partir da lista de tarefas, a tarefa que deseja editar. 2. É apresentado um formulário com os detalhes da tarefa. 3. O utilizador preenche e submete o formulário. A1. O utilizador clica no botão cancelar. A2. O formulário contém erros. A3. O utilizador clica no botão com o ícone do "Caixote do Lixo"	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar tarefas do projeto (sem gravar os dados do formulário). A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário (ex. "código não preenchido", "código já existente", "nome não preenchido", "nome já existente"). Todos os campos à exceção da descrição, esforço efetivo e colaborador são obrigatórios. 2. O fluxo volta ao passo 3 do fluxo primário. A3. O utilizador clica no botão com o ícone do "Caixote do Lixo" 1. INCLUDE (UC 3.11 Eliminar Tarefa)	
Exceções: -	
Pós-Condições: Os dados da tarefa são alterados. É apresentada a página listar tarefas do projeto.	

3.3.7 Eliminar Tarefa do Projeto

Para eliminar uma tarefa do projeto, o utilizador deverá carregar no botão que encontra no topo do lado direito da página de Visualizar/Editar Tarefa. A especificação do caso de uso encontra-se na Tabela 14.

Tabela 14. Eliminar Tarefa do Projeto

UC3.11	**Eliminar Tarefa do Projeto**
Descrição: Este caso de uso permite eliminar uma tarefa.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissão de Gestor de Projetos.	
Fluxo Primário: 1. O utilizador clica no ícone "Eliminar Tarefa". 2. O sistema pergunta se realmente quer fazer essa ação. 3. O utilizador confirma a eliminação. A1. O utilizador cancela a eliminação.	
Fluxos Secundários: A1. O utilizador clica na opção "Cancelar". 1. É fechado o formulário e apresentada a página listar tarefas do projeto (sem gravar os dados do formulário).	
Exceções: -	
Pós-Condições: É eliminada a tarefa. É apresentada a página listar tarefas do utilizador.	

3.3.8 Listar Colaboradores do Projeto

Para visualizar os colaboradores associados ao projeto, o utilizar deverá clicar no separador "Colaboradores", sendo-lhe apresentada uma página com a lista dos colaboradores associados ao projeto (ver Figura 18). No topo da lista do lado direito existe a opção adicionar colaborador, a qual permite adicionar um novo colaborador ao projeto selecionado. A especificação do caso de uso encontra-se na Tabela 15.

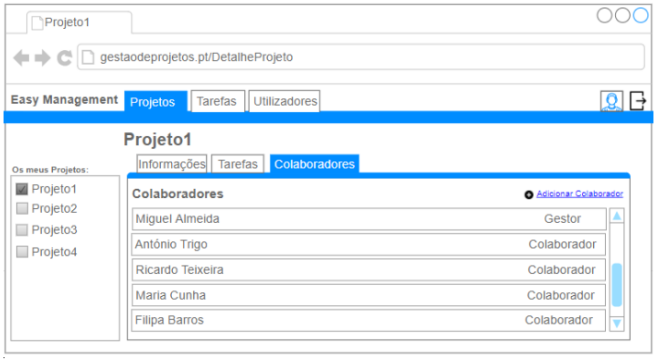


Figura 18. Página Detalhe do Projeto, separador Colaboradores

Tabela 15. Listar Colaboradores do Projeto

UC3.6	**Listar Colaboradores do Projeto**
Descrição: Este caso de uso só está disponível para o Gestor do Projeto e permite a visualização dos colaboradores associados ao projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Colaboradores". 2. É apresentada a lista dos colaboradores/utilizadores associados ao projeto.	
Exceções: -	
Pós-Condições: -	

3.3.9 Adicionar Colaborador ao Projeto

Este caso de uso possibilita selecionar colaboradores a adicionar ao projeto, possibilitando que posteriormente sejam atribuídas tarefas para serem executadas. A especificação do caso de uso encontra-se na Tabela 16.

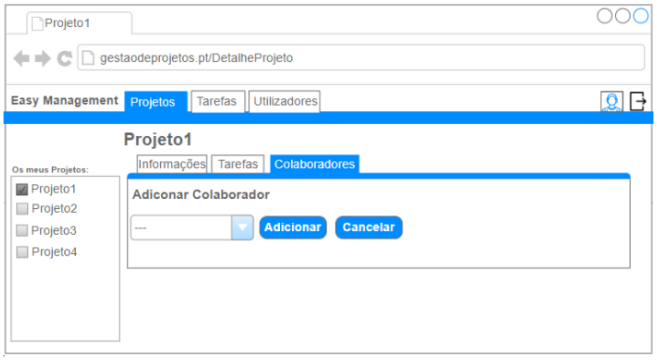


Figura 19. Página Adicionar Colaborador ao Projeto

Tabela 16. Adicionar Colaborador ao Projeto

UC3.5	**Adicionar Colaborador ao Projeto**
Descrição: Este caso de uso só está disponível para o Gestor do Projeto e permite adicionar colaboradores ao projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador clica no botão "Adicionar Colaborador", a partir da lista dos colaboradores. 2. O utilizador seleciona o colaborador a adicionar e clica no botão "Adicionar". A1. O utilizador clica no botão "Cancelar". 3. O utilizador é redirecionado para a página de listagem dos colaboradores.	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar colaboradores do projeto (sem gravar os dados do formulário).	
Exceções:	
Pós-Condições: O colaborador é adicionado ao projeto. É apresentada a página listar colaboradores.	

3.4 Módulo: Gestão de Projetos (Tarefas)

3.4.1 Listar Tarefas do Colaborador

O separador "Tarefas" permite ao colaborador aceder à lista das tarefas (ver Figura 20) que lhe estão associadas e que não estão no estado "Finalizado". Funciona como um

27

dashboard das tarefas a realizar pelo utilizador em questão, existindo também a hipótese de filtrar as tarefas apresentadas por nome. A partir desta página, o utilizador pode criar uma tarefa (desde que tenha permissões de gestor de projeto) ou clicar numa das tarefas da lista e ir para o formulário de edição da mesma (ver Figura 20). A especificação do caso de uso encontra-se na Tabela 17.

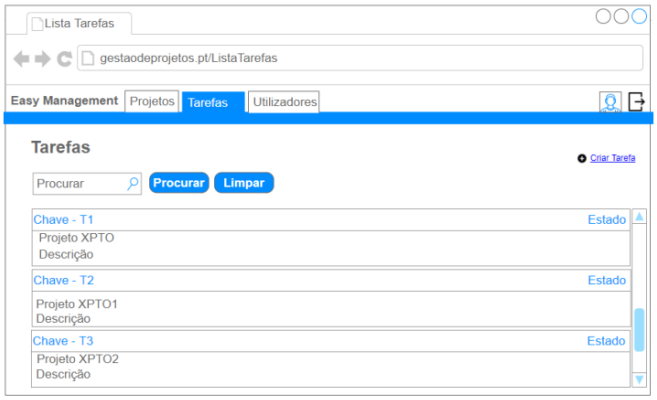


Figura 20. Página Listar Tarefas do Utilizador

Tabela 17. Listar Tarefas do Utilizador

UC3.9	**Listar Tarefas do Colaborador**
Descrição: Este caso de uso permite listar e filtrar as tarefas existentes na aplicação associadas ao utilizador em sessão.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Tarefas". A1. Filtrar tarefas. 2. É apresentada a lista das tarefas do utilizador.	
Fluxos Secundários: A1. Filtrar tarefas 1. O utilizador insere parte do nome da tarefa. 2. O sistema devolve a lista de tarefas filtrada.	
Exceções:	
Pós-Condições:	

4 Modelo de dados

O diagrama de entidade e relacionamentos ilustra a estrutura das entidades relevantes para o funcionamento da aplicação e a forma como estas se relacionam.

Como se apresenta na Figura 21 existem seis entidades: a entidade Utilizador, que representa os utilizadores da aplicação; a entidade Projeto, que guarda a informação relativa aos projetos criados na aplicação; a entidade Tarefa, que guarda a informação relativa às tarefas dos projetos; a entidade Estado, que permite definir os estados para os projetos e tarefas (convencionou-se que seriam os mesmos estados para projetos e tarefas); a entidade Perfil, que guarda a informação dos utilizadores na aplicação (atualmente "Gestor de Projeto" e "Colaborador"); e a entidade Projeto_Utilizadores, que associa os utilizadores aos projetos, com determinado perfis que os utilizadores assumem nos diferentes projetos.

De um modo geral, os atributos das entidades presentes na Figura 21 são autoexplicativos. As exceções são os seguintes atributos: na entidade Utilizador, o atributo CriarProjeto, indica se o utilizador pode ou não criar projetos; na entidade Utilizador, o atributo Ativo que indica se o utilizador está ou não ativo; na entidade Utilizador", o atributo SuperUser indica se o utilizador é do tipo Administrador; nas entidades Projeto e Tarefa, os atributos EsforcoEstimado e EsforcoEfetivo, que representam respetivamente o tempo previsto e o tempo gasto na execução do projeto/tarefa em minutos.

Tem, ainda, de ser verificada a seguinte regra de negócio: uma tarefa é executada por um e um único colaborador.

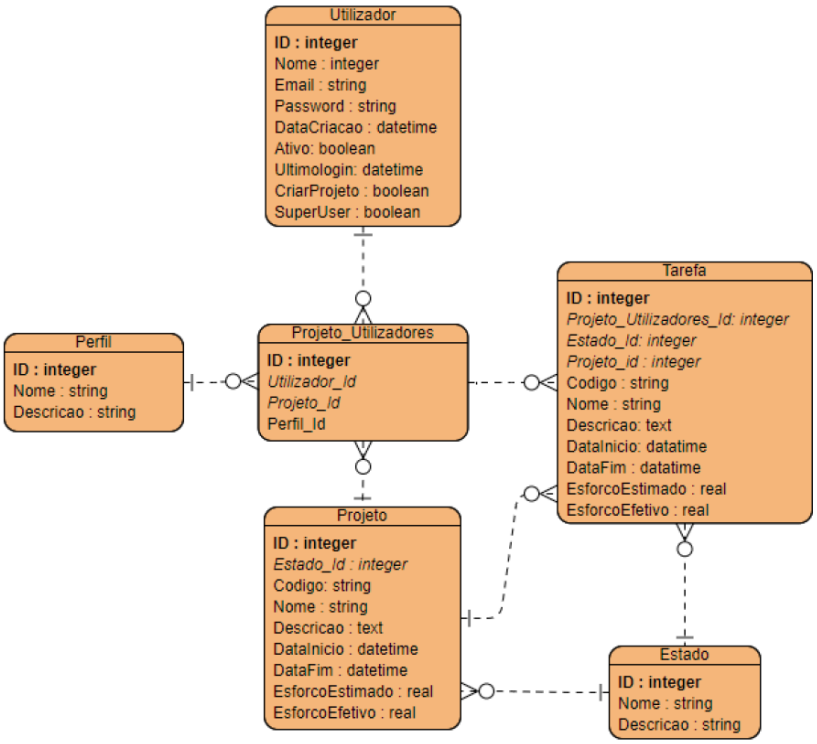


Figura 21. Diagrama de Entidade e Relacionamentos

30

APÊNDICE 2. Protocolo da experiência laboratorial – Fase II – Manutenção de *Software*

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE CONTABILIDADE E ADMINISTRAÇÃO DE COIMBRA

MESTRADO EM

SISTEMAS DE INFORMAÇÃO DE GESTÃO

Protocolo para a Realização de Experiência Laboratorial

Fase II

MESTRANDO:

Nome: Miguel Alves Almeida N.º 17945 (IPC-ISCAC)

ORIENTADORES:

Prof. Doutor António Trigo (IPC-ISCAC)

Prof. Doutor João Varajão (UMINHO)

Índice Geral

Índice de Tabelas

Índice de Figuras

1 Sumário

A segunda fase do projeto surge devido à necessidade de algumas mudanças funcionais na aplicação.

A nova versão tem como principal foco as tarefas e os dados registados. Em concreto:

- O campo "Data de Fim" da tabela "Tarefa", anteriormente definido (Protocolo I), deve ser alterado para "Data de Fim Prevista" (*DeadLine*). Também deve ser adicionado a essa tabela um novo atributo designado por "Data de Fim Efetiva". Deve, ainda, existir a possibilidade de adicionar mais do que um colaborador a cada tarefa, podendo assim a tarefa ser realizada por vários colaboradores.

- Focando os detalhes do projeto, deve ser criado um novo separador, denominado "Indicadores", possibilitando ao gestor do projeto obter uma visão geral das tarefas. Cada colaborador deverá, também, ter disponível no menu principal um separador denominado "Desempenho", que permita obter uma visão geral das tarefas em que intervém.

5

2 Modelo de Caso de Uso

O sistema é composto por três módulos que agrupam as funcionalidades do sistema, encontrando-se identificados na Figura *1*: Módulo Autenticação (P1); Módulo Gestão de Utilizadores (P2); Módulo Gestão de Projetos e Tarefas (P3).

Nesta nova versão não há alteração no Módulo Autenticação nem no Módulo Gestão de Utilizadores.

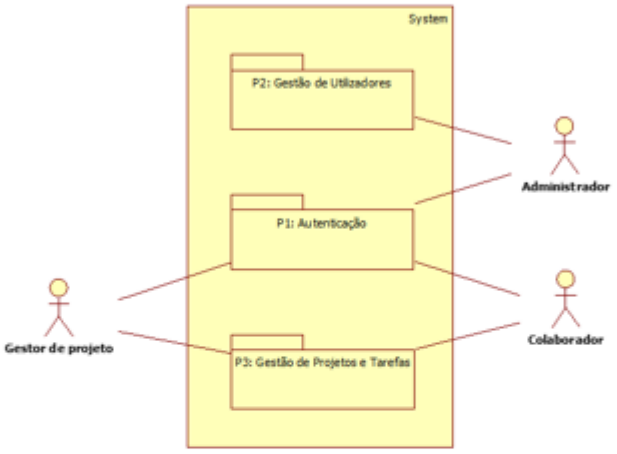


Figura 1. Módulos de Sistema

6

2.1 Módulo Gestão de Projetos e Tarefas

A Figura 2 apresenta os casos de uso refentes ao módulo Gestão de Projetos e Tarefas

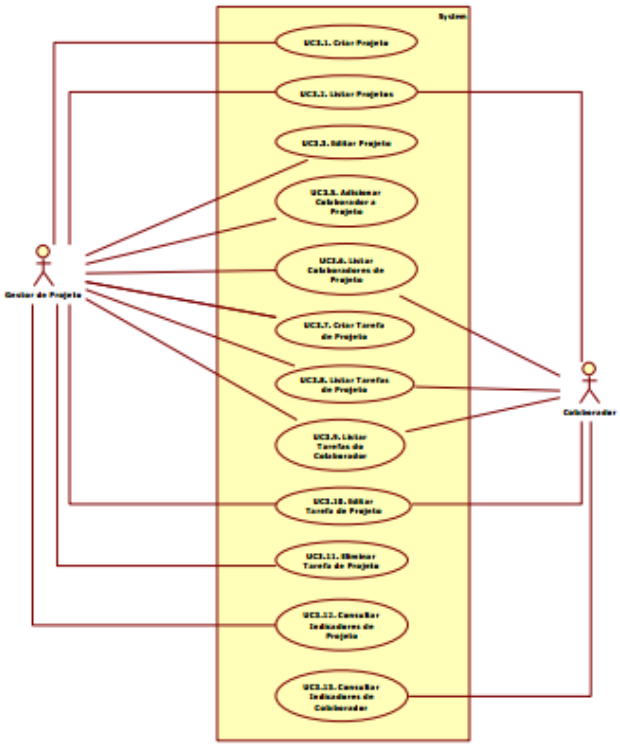


Figura 2. Diagrama de Casos de Uso (Gestão de Projetos e Tarefas)

7

3 Especificação dos casos de uso

3.1 Módulo Gestão de Projetos e Tarefas

3.1.1 Criar Tarefa do Projeto

Para aceder à funcionalidade Criar Tarefa, o utilizador deverá clicar no botão que se encontra no canto superior do lado direito da página Listar Tarefas. Será então direcionado para um formulário que permite a criação de tarefas. Esse formulário possui vários campos, nomeadamente, código, nome, estado ("Por fazer", "Em Andamento", "Pendente" e "Finalizado"), o colaborador que irá realizar a tarefa, uma breve descrição, a data de início, data de fim prevista e a data de fim efetiva da tarefa, e ainda o esforço estimado e efetivo de execução da tarefa (em minutos). Alguns dos valores destes campos, como, por exemplo, o estado da tarefa, serão posteriormente atualizados durante a execução da tarefa. A especificação do caso de uso encontra-se na Tabela 1.

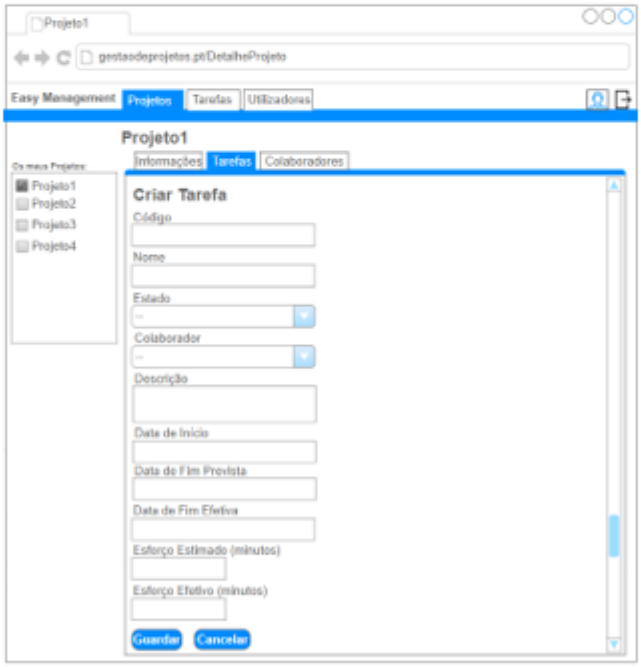


Figura 3. Página Criar Tarefa do Projeto

8

Tabela 1. Criar Tarefa do Projeto

UC3.7	**Criar Tarefa do Projeto**
Descrição: Este caso de uso permite criar uma tarefa associada a um determinado projeto.	
Pré-Condições: O utilizador tem de estar autenticado e ter permissões de Gestor de projeto.	
Fluxo Primário: 1. Clica em "Criar Tarefa". 2. É apresentado um formulário para preenchimento. 3. O utilizador preenche e submete o formulário. A1. O utilizador clica no botão cancelar. A2. O formulário contém erros.	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar tarefas (sem gravar os dados do formulário). A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário. 2. O fluxo volta ao passo 3 do fluxo primário.	
Exceções:	
Pós-Condições: É registado uma nova tarefa. É apresentada a página listar tarefas.	

9

3.1.2 Editar Tarefa do Projeto

Para aceder à funcionalidade de Editar Tarefa, o utilizador deverá selecionar, na lista de tarefas, a tarefa desejada, sendo então redirecionado para a página de visualização/edição de tarefa. Após alterar os campos desejados, o utilizador deverá clicar no botão guardar para submeter as alterações efetuadas. Para além da edição da tarefa o utilizador pode eliminar uma tarefa clicando no botão com o ícone de "Caixote do Lixo" disponível no canto superior direito. A especificação do caso de uso encontra-se na Tabela 2.

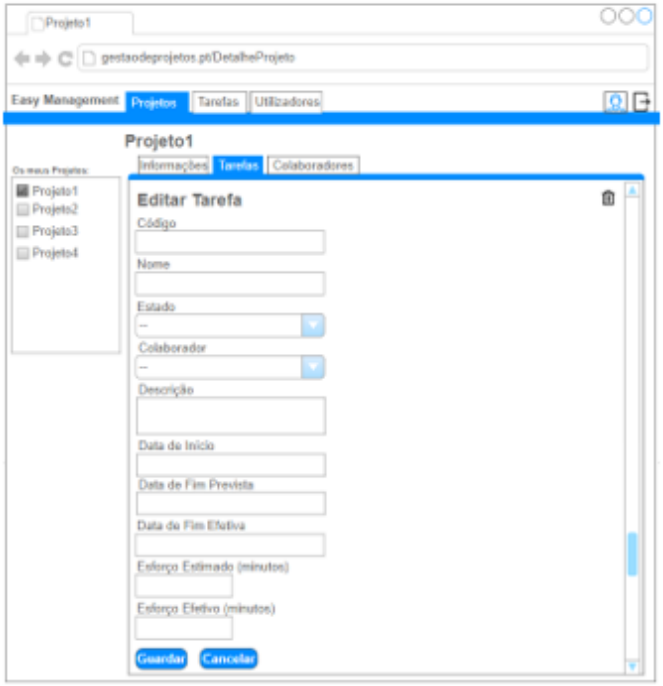


Figura 4. Página Editar Tarefa do Projeto

10

Tabela 2. Editar Tarefa do Projeto

UC3.10	**Editar Tarefa do Projeto**
Descrição: Este caso de uso permite editar uma tarefa associada a um determinado projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador seleciona, a partir da lista de tarefas, a tarefa que deseja editar. 2. É apresentado um formulário com os detalhes da tarefa. 3. O utilizador preenche e submete o formulário. A1. O utilizador clica no botão cancelar. A2. O formulário contém erros. A3. O utilizador clica no botão com o ícone do "Caixote do Lixo"	
Fluxos Secundários: A1. O utilizador clica no botão "Cancelar". 1. É fechado o formulário e apresentada a página listar tarefas do projeto (sem gravar os dados do formulário). A2. O formulário contém erros. 1. É apresentada uma mensagem que identifica os erros verificados no preenchimento do formulário (ex. "código não preenchido", "código já existente", "nome não preenchido", "nome já existente"). Todos os campos à exceção da descrição, esforço efetivo e colaborador são obrigatórios. 2. O fluxo volta ao passo 3 do fluxo primário. A3. O utilizador clica no botão com o ícone do "Caixote do Lixo" 1. INCLUDE (UC 3.11 Eliminar Tarefa)	
Exceções: -	
Pós-Condições: Os dados da tarefa são alterados. É apresentada a página listar tarefas do projeto.	

11

3.1.3 Consultar indicadores do Colaborador

Para aceder à listagem de utilizadores, o administrador terá de clicar no separador "Indicadores" presente na barra de tarefas. De seguida, será direcionado para a página onde são apresentados os indicadores do colaborador, que se apresenta na Figura 5 . A especificação do caso de uso encontra-se na Tabela 3.

Figura 5. Página Consultar indicadores do Colaborador

Tabela 3. Consultar indicadores do Colaborador

UC3.13	**Consultar Indicadores do Colaborador**
Descrição: Este caso de uso permite em consultar os indicadores referentes ao colaborador.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Indicadores". 2. São apresentados os indicadores do colaborador.	
Fluxos Secundários:	
Exceções:	
Pós-Condições:	

12

3.1.4 Consultar indicadores de Projeto

Para aceder à página de indicadores de projeto (ver Figura 6), o utilizador terá de aceder primeiro à página de listagem dos projetos e selecionar o projeto. Posteriormente, deve clicar no separador "Indicadores" onde são apresentados os indicadores do projeto selecionado. A especificação do caso de uso encontra-se na Tabela 4.

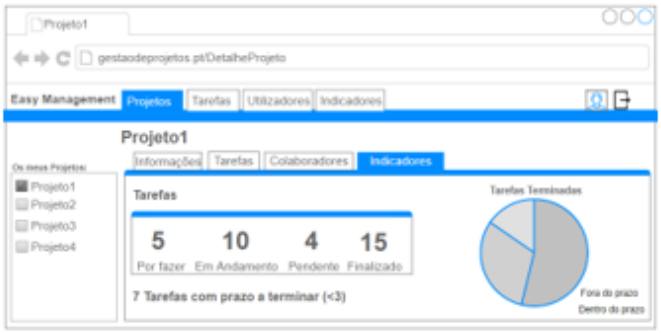


Figura 6. Página Consultar indicadores do Projeto

Tabela 4. Consultar indicadores do Projeto

UC3.12	**Consultar Indicadores do Projeto**
Descrição: Este caso de uso permite em consultar os indicadores referentes ao projeto.	
Pré-Condições: O utilizador tem de estar autenticado.	
Fluxo Primário: 1. O utilizador acede ao separador "Indicadores". 2. São apresentados os indicadores do projeto.	
Fluxos Secundários:	
Exceções:	
Pós-Condições:	

13

4 Modelo de dados

O diagrama de entidade e relacionamentos ilustra a estrutura das entidades relevantes para o funcionamento da aplicação e a forma como estas se relacionam.

Como se apresenta na Figura 4 existem sete entidades. Em comparação com o protocolo da primeira fase, foi acrescentada a entidade ProjetoUtilizadores_Tarefa, que relaciona a entidade Projeto_Utilizadores com a entidade Tarefa. Foram também introduzidos dois novos atributos na entidade Tarefa, denominado por "Data de Fim Prevista" e "Data de Fim Efetiva". Com esta mudança, o atributo "Data_Fim" foi eliminado.

Tem, ainda, de ser verificada a seguinte regra de negócio: uma tarefa é executada por um ou mais colaboradores.

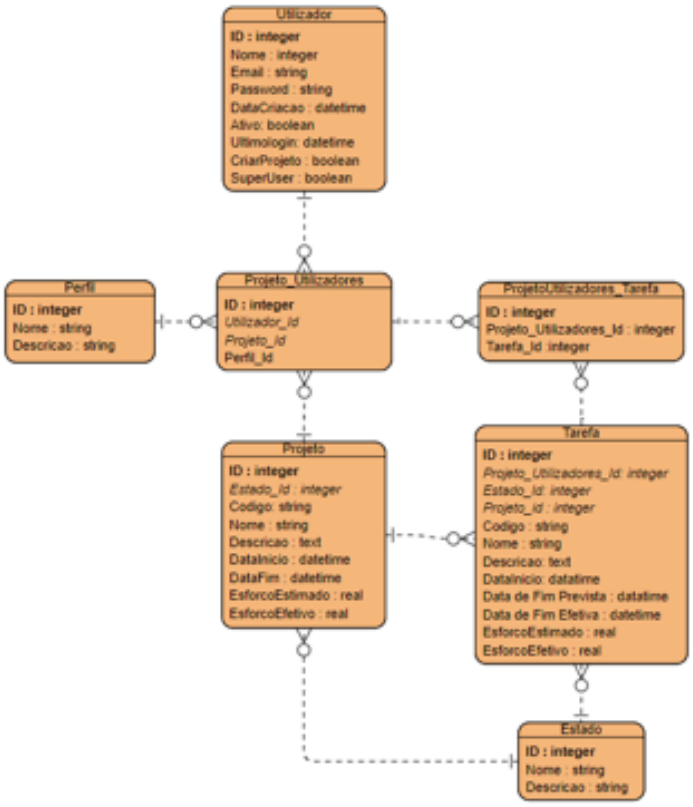
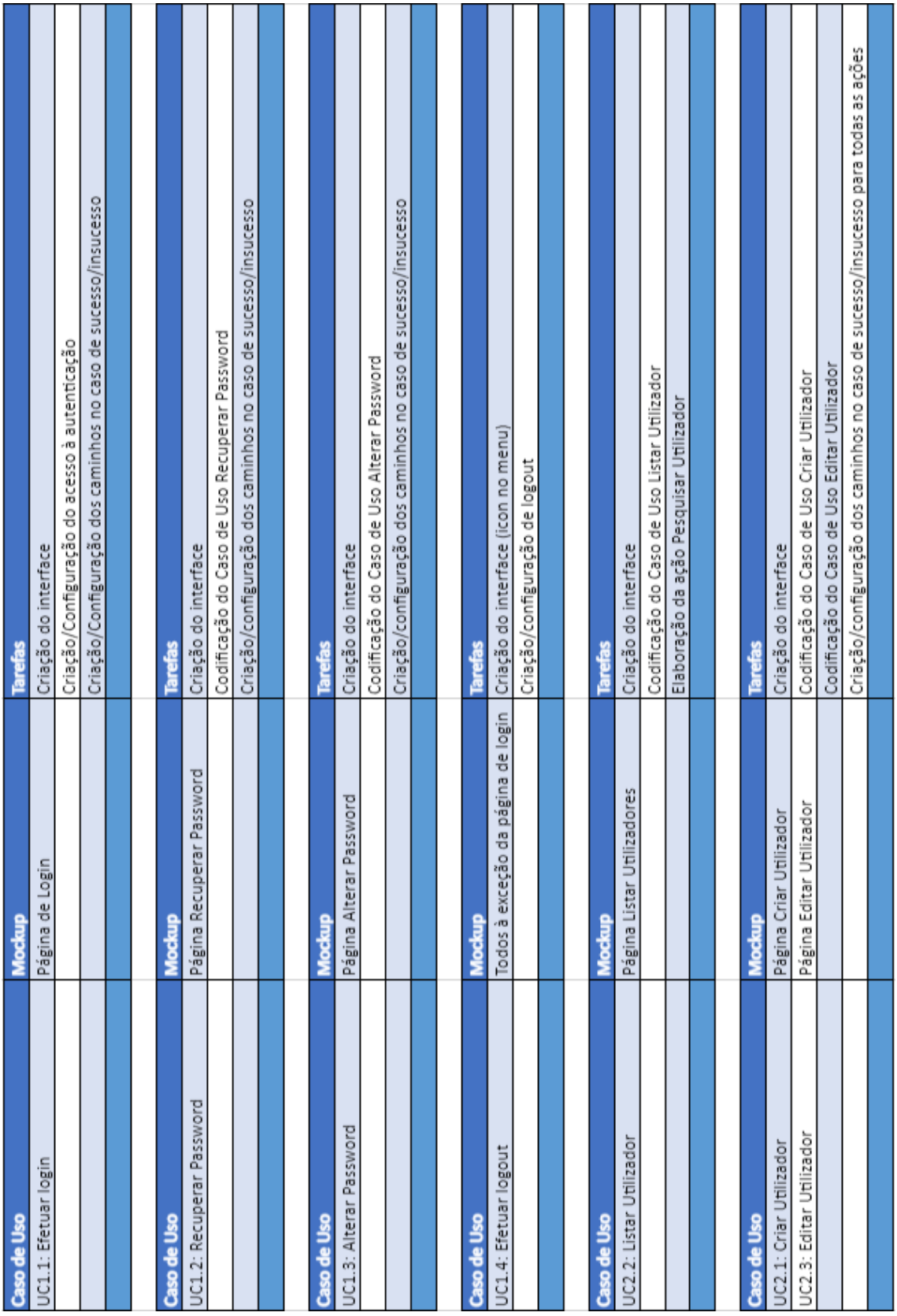


Figura 7. Diagrama de Entidade e Relacionamentos

APÊNDICE 3. Lista de tarefas - Fase I



Caso de Uso	Mockup	Tarefas
UC1.1: Efetuar login	Página de Login	Criação do interface
		Criação/Configuração do acesso à autenticação
		Criação/Configuração dos caminhos no caso de sucesso/insucesso

Caso de Uso	Mockup	Tarefas
UC1.2: Recuperar Password	Página Recuperar Password	Criação do interface
		Codificação do Caso de Uso Recuperar Password
		Criação/configuração dos caminhos no caso de sucesso/insucesso

Caso de Uso	Mockup	Tarefas
UC1.3: Alterar Password	Página Alterar Password	Criação do interface
		Codificação do Caso de Uso Alterar Password
		Criação/configuração dos caminhos no caso de sucesso/insucesso

Caso de Uso	Mockup	Tarefas
UC1.4: Efetuar logout	Todos à exceção da página de login	Criação do interface (icon no menu)
		Criação/configuração de logout

Caso de Uso	Mockup	Tarefas
UC2.2: Listar Utilizador	Página Listar Utilizadores	Criação do interface
		Codificação do Caso de Uso Listar Utilizador
		Elaboração da ação Pesquisar Utilizador

Caso de Uso	Mockup	Tarefas
UC2.1: Criar Utilizador	Página Criar Utilizador	Criação do interface
UC2.3: Editar Utilizador	Página Editar Utilizador	Codificação do Caso de Uso Criar Utilizador
		Codificação do Caso de Uso Editar Utilizador
		Criação/configuração dos caminhos no caso de sucesso/insucesso para todas as ações

Caso de Uso	Mockup	Tarefas
UC3.2: Listar Projetos	Página Listar Projetos	Criação do interface
		Codificação do Caso de Uso Listar Projetos
		Elaboração da ação Pesquisar Projetos

Caso de Uso	Mockup	Tarefas
UC3.1: Criar Projeto	Página Criar Projeto	Criação do interface Criar Projeto
UC3.3: Editar Projeto	Página Detalhe do Projeto	Criação do interface Editar Projeto
		Codificação do Caso de Uso Criar Projeto
		Codificação do Caso de Uso Editar Projeto
		Criação/configuração dos caminhos no caso de sucesso/insucesso para todas as ações

Caso de Uso	Mockup	Tarefas
UC3.8: Listar Tarefas do Projeto	Página Detalhe de Projeto	Criação do interface
		Codificação do Caso de Uso Listar Tarefas do Projeto
		Elaboração da ação Pesquisar Tarefas

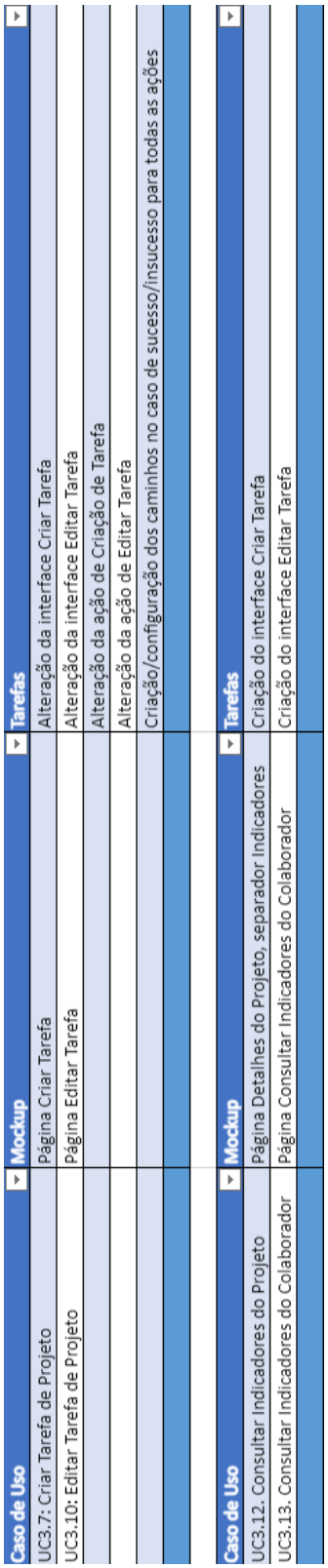
Caso de Uso	Mockup	Tarefas
UC3.6: Listar Colaboradores do Projeto	Página Detalhe do Projeto	Criação do interface
		Codificação do Caso de Uso Listar Colaboradores de Projeto
		Elaboração da ação Pesquisar Colaborador

Caso de Uso	Mockup	Tarefas
UC3.5: Adicionar Colaborador a Projeto	Página Adicionar Colaborador	Criação do interface
		Codificação do Caso de Uso Adicionar Colaborador a Projeto
		Criação/configuração dos caminhos no caso de sucesso/insucesso

Caso de Uso	Mockup	Tarefas
UC3.9: Listar Tarefas do Colaborador	Página Listar Tarefas	Criação do interface
		Codificação do Caso de Uso Listar Tarefas
		Elaboração da ação Pesquisar Projetos

Caso de Uso	Mockup	Tarefas
UC3.7: Criar Tarefa de Projeto	Página Criar Tarefa	Criação do interface Criar Tarefa
UC3.10: Editar Tarefa de Projeto	Página Editar Tarefa	Criação do interface Editar Tarefa
UC3.11: Eliminar Tarefa de Projeto		Codificação do Caso de Uso Criar de Tarefa
		Codificação do Caso de Editar Tarefa
		Criação/configuração dos caminhos no caso de sucesso/insucesso para todas as ações

APÊNDICE 4. Lista de tarefas - Fase II



Caso de Uso	Mockup	Tarefas
UC3.7: Criar Tarefa de Projeto	Página Criar Tarefa	Alteração da interface Criar Tarefa
UC3.10: Editar Tarefa de Projeto	Página Editar Tarefa	Alteração da interface Editar Tarefa
		Alteração da ação de Criação de Tarefa
		Alteração da ação de Editar Tarefa
		Criação/configuração dos caminhos no caso de sucesso/insucesso para todas as ações

Caso de Uso	Mockup	Tarefas
UC3.12. Consultar Indicadores do Projeto	Página Detalhes do Projeto, separador Indicadores	Criação do interface Criar Tarefa
UC3.13. Consultar Indicadores do Colaborador	Página Consultar Indicadores do Colaborador	Criação do interface Editar Tarefa