PM

Desenvolvimento do Sistema de Informação de Suporte à Gestão da Factory

Centro de desenvolvimento SAP

PROJETO DE MESTRADO

José Filipe Trindade Silva

MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

maio | 2019

Desenvolvimento do Sistema de Informação de Suporte à Gestão da Factory

Centro de desenvolvimento SAP

PROJETO DE MESTRADO

José Filipe Trindade Silva
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Leonel Domingos Telo Nóbrega

CO-ORIENTAÇÃO
Carlos Manuel Saragoça de Castro

UNIVERSIDADE da MADEIRA

Desenvolvimento do Sistema de Informação de Suporte à Gestão da Factory

José Filipe Trindade Silva

Constituição do júri de provas públicas:

- Presidente do júri: David Sardinha Andrade de Aveiro (Professor Auxiliar da Universidade da Madeira);
- Arguente: Frederica Margarida Camacho Gonçalves (Professora Auxiliar da Universidade da Madeira);
- Orientador: Leonel Domingos Telo Nóbrega (Professor Auxiliar da Universidade da Madeira);

Maio 2019

Funchal – Portugal

RESUMO

Apesar dos processos de desenvolvimento de s*oftware* terem surgido há cerca de 40 anos atrás, a sua gestão ainda é crítica e pouco previsível. Atendendo a esta evidência, tem havido um grande esforço, por parte da comunidade de desenvolvimento de *software,* para criar ferramentas que suportem a referida atividade de uma forma mais simplificada e mais colaborativa. Estes tipos de ferramentas têm-se mostrado eficazes na obtenção de um produto de qualidade, no tempo estimado e trazendo lucros para as empresas.

O âmbito deste projeto centra-se numa proposta apresentada pela empresa Odkas (empresa do sector dos serviços de implementação e consultadoria SAP) que visa a criação duma plataforma que permitisse suportar a sua gestão de projetos. Esta plataforma, tem como objetivo de centralizar e uniformizar a gestão de projetos através de uma ferramenta única, que permita a todos os intervenientes de um projeto a consulta de toda a informação associada ao mesmo. Embora haja conhecimento da existência de ferramentas deste tipo no mercado, o objetivo é obter uma ferramenta própria da empresa.

A implementação deste projeto dividiu-se em duas partes. Uma parte para o cliente FOMS (Factory Odkas Management System) e outra para o servidor FOMS. Para o cliente FOMS foi utilizada a *framework* OpenUI5, em JavaScript e para o servidor FOMS foi utilizado uma API REST utilizando o Node.js. Para finalizar foi utilizado o Microsoft Azure para criar um serviço *online* do sistema e prosseguiu-se com as suas avaliações, nomeadamente, testes de usabilidade e desempenho.

PALAVRAS-CHAVE

Gestão de projetos

Processo de desenvolvimento de *software*

Aplicação Web

OpenUI5

Node.js

API REST

ABSTRACT

Although software development processes came about 40 years ago, the management of these processes are still critical and unpredictable. Due to this, there has been a great deal of effort, over time, by the software development community, to develop tools that support this process in a more simplified and unified way. These types of tools have proven to be effective in obtaining a quality product in the estimated time, bringing profits to the companies.

The scope of this project focuses on the proposal presented by the company Odkas (a company of the sector of implementation and consulting SAP) that aims to create a platform that allows to support its project management. This platform purpose is to centralize and standardize project management through a unique tool that allows all project stakeholders to consult all the information associated with it. Although it is known of the existence of such tools in the market, the goal is to obtain a company's own tool.

The implementation of this project was divided into two parts. One part for the FOMS (Factory Odkas Management System) client and one for the FOMS server. For the FOMS client, it was used the OpenUI5 framework in JavaScript and for the FOMS server was used a REST API in Node.js. Finally, the Microsoft Azure was used to create an online system service and the system evaluations, such as usability and performance tests, were followed up.

KEYWORDS

Project management

Software Development Process

Web Application

OpenUI5

Node.js

REST API

AGRADECIMENTOS

Esta secção é dedicada a todos os que ajudaram a tornar esta etapa menos árdua e simplificada.

Agradeço ao professor Leonel Nóbrega por todo o acompanhamento ao longo do projeto, pela sua dedicação e paciência em identificar os meus lapsos e pelos seus conselhos que me orientaram na direção certa.

Agradeço ao Carlos Castro, como também a toda a equipa Odkas Factory, pela confiança depositada em mim e a disponibilidade por participarem nos testes finais da ferramenta.

Por fim, agradeço aos meus pais, irmão e amigos por todo o apoio incondicional e por proporcionarem um ambiente de motivação que me levou ao desfecho desta etapa.

ÍNDICE

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

ACRÓNIMOS

JS – JavaScript

MVC – Modelo, Vista e Controlador

UI – User Interface

JSON – Java Script Object Notation

OData – Open Data Protocol

SAP – Systems, Applications and Products

URL – Uniform Resource Locator

XML – eXtensible Markup Language

HTML – Hypertext Markup Language

DOM – Document Object Model

API – Application Programming Interface

REST – Representational State Transfer

PHP – Hypertext Preprocessor

UML – Unified Modeling Language

CRUD – Create Read Update Delete

HTTP – HyperText Transfer Protocol

CORS – Cross-Origin Resource Sharing

RF – Requisito Funcional

RNF – Requisito Não Funcional

1 INTRODUÇÃO

1.1 Motivação/Contexto

A construção de um produto de software é alcançada a partir de um esforço praticado em conjunto por uma equipa, capaz de empregar uma boa comunicação entre os membros, a fim de diminuir o tempo de execução de tarefas individuais [1]. O colapso de muitos projetos de software deriva de um mau planeamento do tempo de implementação e da gestão do processo que leva à sua construção. Este mau planeamento resulta da utilização de técnicas de estimação de esforço pouco precisas e no recurso a suposições de estimação muito distintas do tempo que é, realmente, necessário. Confundir esforço com progresso, efetuar uma má monitorização do progresso do planeamento de um projeto e adicionar mais programadores perto do prazo de entrega são outras causas, que se pode enumerar, capazes de provocar o fracasso de um projeto de *software* [1].

A gestão de projetos baseada na determinação de requisitos é uma tarefa complexa [2]. Consequentemente, é necessário recorrer à utilização de processos sistematizados no desenvolvimento de *software*, a fim de melhorar a sua eficácia. Um processo de *software* identifica-se pelo conjunto de atividades que são executadas durante o desenvolvimento de um projeto [3]. Este processo envolve um fluxo de tarefas, que podem ser resumidas nas quatro atividades principais representadas na Figura 1.

Figura 1: Atividades de um processo de desenvolvimento de *software*.

Os estudos na área de processos de *software* promove a ocorrência de novos processos, ou a ocorrência de uma evolução dos métodos e técnicas que são capazes de assimilar e qualificar processos, que são suportados pelas suas atividades [4].

O âmbito deste projeto centra-se na necessidade, da empresa Odkas [5] (empresa do sector dos serviços de implementação e consultadoria SAP), de desenvolver uma plataforma que automatize, em certa medida, a gestão dos seus

projetos. Esta empresa pretende centralizar e uniformizar esta gestão, através de uma ferramenta única que permita, a todos os intervenientes de um projeto, a consulta de toda a informação associada ao mesmo.

Uma vez, que a empresa Odkas possui uma sede na cidade do Porto e outra na cidade do Funchal (equipa denominada por Odkas Factory), uma aplicação que suporte o processo de desenvolvimento de *software* entre estas é algo essencial para a empresa, eliminando, deste modo, a limitação de uma equipa ser constituída apenas por elementos locais e criando, assim, uma maior diversidade de escolha de membros aquando da formação das equipas. Além disso, promove melhorias na monitorização de tarefas e uma maior facilidade na troca de informação. Embora haja conhecimento da disponibilidade de ferramentas deste tipo no mercado, o objetivo é obter uma ferramenta própria da empresa.

1.2 Problema

O problema detetado, pela empresa Odkas, surge da necessidade de obter toda a informação relacionada com um projeto, que seja essencial para o seu desenvolvimento (como a sua descrição, requisitos e tarefas), numa única plataforma. Era, ainda, indispensável haver uma disposição uniformizada e centralizada da gestão dos projetos, com a capacidade de criar equipas juntamente com a respetiva adição de membros e atribuição de tarefas.

Os objetivos da empresa que levaram ao surgimento deste problema foram:

- Registar os requisitos/ideias apresentados pelo cliente;
- Notificar o líder de equipa sobre a criação de novas tarefas;
- Visualização, por parte do líder de equipa, de toda a informação referente a nova tarefa (requisitos funcionais e técnicos, justificação, prazo espectável, âmbitos e outros);
- A possibilidade de o líder poder obter o estado atual da equipa (tarefas associadas a cada elemento e percentagem de conclusão dos projetos), em qualquer momento;

- Notificação para os membros de equipa, aquando na atribuição de tarefas;
- Conclusão e aprovação das tarefas pelo líder de equipa.

1.3 Contribuições

Este projeto permite obter conhecimento sobre o desenvolvimento de uma aplicação *web* de gestão de projetos, baseada em processos de desenvolvimento de *software,* com finalidade de uso empresarial. Referente ao desenvolvimento do lado do cliente, é possível obter práticas de desenvolvimento ao utilizar a livraria em JavaScript OpenUI5 [6]. Relativo ao lado do servidor foi utilizado o Node.js juntamente com a *framework* Express, para a elaboração de uma API REST. Foi, também, desenhado e implementado um sistema de permissões, que apresenta uma estrutura genérica e adaptável a implementação noutros sistemas. Além disto, é possível obter práticas no desenho e implementação de um padrão arquitetural de produtores e consumidores, para um sistema de notificações utilizando a tecnologia Socket.io. O sistema de autenticação do sistema utiliza a biblioteca Passport.js e MySQLStore para o Node.js. Por fim, foi utilizado o Microsoft Azure para criar o serviço em *cloud* do sistema

Este documento engloba todo o procedimento de desenvolvimento da aplicação, como também decisões tomadas e devidas justificações.

1.4 Estrutura da dissertação

Este relatório está estruturado em 7 capítulos, que se descrevem na seguinte forma:

1. **Introdução**: onde está exposta a motivação e o contexto deste projeto, qual o problema identificado pela empresa ODKAS e quais são as contribuições esperadas deste projeto;
2. **Contexto tecnológico**: inicia-se com um pequeno estudo sobre os processos de desenvolvimento de *software*. Logo em seguida é apresentado as tecnologias que foram utilizadas ao longo do desenvolvimento do projeto, como o

OpenUI5 e o Node.js (poderá saltar esta secção, cajo já esteja familiarizado com estas tecnologias);

3. **Ferramentas existentes**: secção onde está elaborado um estudo sobre ferramentas semelhantes à que foi desenvolvida neste projeto, a fim de as comparar;
4. **Análise**: representa a fase de desenho de *software* efetuado antes do início do desenvolvimento da aplicação, onde estão definidos os requisitos e o desenho de *software*;
5. **Desenvolvimento**: apresenta a descrição do desenvolvimento do *software* da aplicação *web*, como também as decisões que foram tomadas ao longo deste;
6. **Avaliação**: é demonstrado os testes efetuados à aplicação, nomeadamente testes de usabilidade e de desempenho;
7. **Conclusão e trabalho futuro**: é apresentada a conclusão referindo o trabalho efetuado, como também, é indicado algum do trabalho futuro que poderá ser realizado neste projeto.

2 CONTEXTO TECNOLÓGICO

2.1 Processos de desenvolvimento de *software*

O conceito de desenvolvimento de *software* surgiu na década de 1970 [7] originando, assim, os primeiros métodos formais de criação de *software*. Estes sofreram, mais tarde, algumas alterações até surgir os modelos de ciclo de vida de desenvolvimento de sistemas. Estes modelos são, atualmente, considerados como modelos muito limitados, pois, restringem-se principalmente na produção de código e não recorrem a elaboração de muitos testes. Em 1982 [8], iniciou-se uma evolução no desenvolvimento de software centrada no objetivo de entregar sistemas de qualidade. Desde então vários trabalhos e estudos têm sido desenvolvidos a fim de avaliar e melhorar os processos de software, pois, a maturidade de um processo é indicada pela sua melhoria contínua e quão perto a sua evolução está da sua conclusão.

A principal razão pela qual o desenvolvimento de *software* tem estado numa evolução continua deve-se a complexidade do próprio desenvolvimento. Este requer um grande nível de coordenação por parte dos elementos de uma equipa, uma recolha de requisitos que sejam consistentes e a uma adaptação constante a evolução de tecnologias e ferramentas. Tudo isto cria um conjunto complexo envolvido de incertezas.

"*A incerteza é um dos factos fundamentais da vida. É tão inerradicável das decisões de negócios quanto daquelas de qualquer campo.*"

F.H. Knight [9]

Segundo Bach [10], os problemas no desenvolvimento de *software* baseiam-se nos três P's, denominados de pessoas, problema e processo. Um projeto não deve ser construído numa abordagem de coleção de tarefas e produtos, mas sim numa abordagem adaptativa. Embora já se tenha passado cerca de 40 anos após o surgimento dos processos de desenvolvimento de *software*, muitos projetos e produtos falham devido a uma deficiente definição, gestão e controle do seu processo de desenvolvimento. Contudo, existem as seguintes abordagens que podem ser seguidas de modo a haver uma minimização de falhas:

- **Definir e controlar o processo de desenvolvimento de *software*.** Só é possível obter *software* de qualidade a partir de um processo bem qualificado;
- **Definir o foco nas pessoas que criam o produto e gerem o projeto.** Cada produto de *software* resulta da colaboração de um conjunto de pessoas, logo, estas necessitam de compreender as habilidades de colaboração, a forma correta de pensar, quais os métodos de desenvolvimento e requerem um ambiente de desenvolvimento onde estas habilidades possam ser manifestadas.

Os processos de *software* não devem ser vistos como programas porque são executados por pessoas, não por computadores, que possuem comportamentos deterministas [4]. As pessoas, são influenciados por objetivos e pelo ambiente de desenvolvimento, sendo necessário ter em consideração o tipo relação criada entre os processos e as entidades que os realizam.

Uma equipa que utilize uma ferramenta de desenvolvimento de processos de *software* permite a criação de um trabalho cooperativo associado com partilhas de conhecimento, técnicas e métodos que sejam úteis ao projeto [11]. O processo de desenvolvimento que permite a definição da equipa e que presta suporte à gestão de todas as tarefas, traz melhorias significativas a uma organização. Esta abordagem foi extremamente notória na Divisão de Engenharia de Software de Hughes [12], em que se obteve uma redução no orçamento de cerca de 2 milhões de dólares anuais. Embora, seja de referir que a aprendizagem do processo ao longo do tempo, por parte dos programadores, poderá ser um dos fatores que influenciou este melhoramento.

No entanto, existe algum desinteresse por parte de algumas equipas, em utilizar processos de desenvolvimento de *software*, pois, estes frequentemente estão associados com documentação que inibem a criatividade. Outras vezes, os processos são constituídos por tarefas que estão fora dos objetivos do projeto em si e são, portanto, vistos como processos difíceis e consumidores de tempo. Devido a isto, algumas equipas têm a tendência de recorrer a práticas ágeis, que se focam em tarefas de

desenvolvimento e práticas de acompanhamento do desenvolvimento de *software*. Atualmente cerca de 85% das companhias de *software* utilizam metodologias ágeis [13], pois, estas permitem uma entrega periódica de versões do produto ao cliente e desta forma alcançar o objetivo deste.

O que difere as ferramentas de gestão de projetos ágeis das ferramentas tradicionais é o facto de apresentarem uma abordagem adaptativa e dinâmica baseada numa gestão contínua do projeto [14]. Sendo, assim, boas ferramentas de gestão de projeto devem de possuir as seguintes características:

- **Métricas e relatórios ágeis**:
  - Monitorização de tempo;
  - Garantia de qualidade;
  - Relatórios que apresentem progressos de fácil compreensão;
  - Possibilidade de integração com o sistema existente na companhia;
- **Comunicação**:
  - Comunicar atualizações entre as equipas;
  - Partilhar listas de tarefas, atribuições de tarefas e comentários;
- **Avaliação de projetos**:
  - Identificar e resolver problemas no projeto;
  - Avaliação constante do desempenho;
  - Estimativa financeira.

2.2 Tecnologias abordadas

Visto que este projeto ter sido proposto pela empresa ODKAS, fora sugerido a utilização da biblioteca JS OpenUI5, uma vez que esta tecnologia tem sido utilizada pela própria empresa para o desenvolvimento de projetos.

2.2.1 OpenUI5

O OpenUI5 é uma biblioteca JS *open source*, mantida pela SAP e disponível sob a licença Apache 2.0 [6]. Esta *framework* tem como finalidade o desenvolvimento de aplicações *web,* fornecendo um conjunto de controles de *interface* baseado no padrão

arquitetural MVC [15]. O OpenUI5 ao se influenciar pela utilização de JS, possibilita a criação de aplicações *web,* sobremodo, interativas e com uma velocidade de carregamento de dados eficiente, no entanto, compromete a velocidade de carregamento de gráficos e imagens.

2.2.1.1 MVC

O conceito MVC é utilizado no OpenUI5 a fim de apresentar uma representação separada entre a informação e a interação do utilizador [16]. O funcionamento da arquitetura MVC no OpenUI5 está esquematizado na Figura 2.

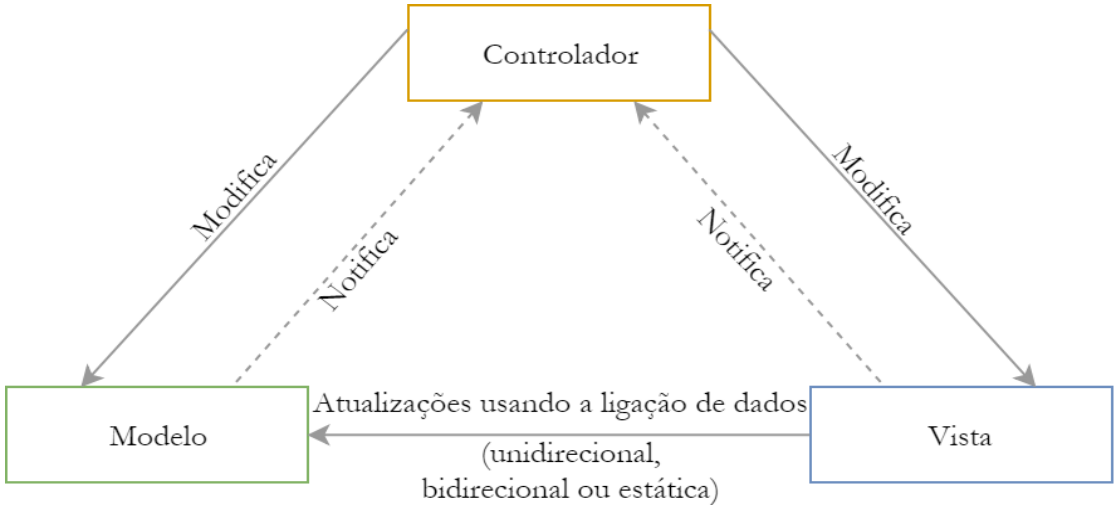


Figura 2: Funcionamento da arquitetura do MVC.

Esta separação é utilizada para criar uma melhor legibilidade, manutenção, extensibilidade e facilidade de modificação da vista sem ter que efetuar alterações na lógica subjacente. Permitindo, ainda, definir várias exibições dos mesmos dados.

As vistas e os controladores formam, normalmente, relações 1:1 mas é possível ter controladores que não estejam associados a uma vista, que se denominam de controladores da aplicação. O mesmo acontece para as vistas, sendo possível criar vistas sem o seu controlador permitindo criar fragmentos que sejam comuns a várias vistas, tudo isto com a finalidade de reutilização de código. Os tipos de vista suportados pelo OpenUI5 são: **XML, JSON, JS, HTML** [17]. O tipo de vista recomendado é o XML, devido a aplicar uma separação entre a UI e a lógica da aplicação implementada no controlador.

Referente aos modelos, o OpenUI5 permite a utilização dos seguintes tipos de modelos [18]:

- **Modelo OData**: possibilita a ligação a serviços de dados do tipo OData, suportando modos de ligação bidirecionais, unidirecionais e estáticos. Atualmente suporta as versões: OData V2 e OData V4;
- **Modelo JSON**: utilizado, normalmente, para a ligação entre os controladores e objetos de dados em JS, que são serializados no formato JSON. Este é um modelo do lado do cliente e, portanto, é indicado para pequenos conjuntos de dados. Os modos de ligação possíveis são bidirecionais, unidirecionais e estáticos.
- **Modelo XML:** Tal como o modelo JSON é indicado para pequenos conjuntos de dados para o lado do cliente. Este modelo não contém mecanismos para paginação baseada em servidor ou carregamento de deltas. O modelo XML suporta modos de ligação bidirecionais, unidirecionais e estáticos;
- **Modelo de recurso:** projetado para manipular dados em pacotes de recursos, principalmente para fornecer textos em diferentes idiomas. Este modelo apenas suporta ligações estáticas, uma vez que lida com texto estático.

Os modelos do lado do cliente como o modelo JSON, XML e recurso, carregam os dados do servidor e só depois ficam disponíveis para o cliente. Isto significa que operações como ordenação e filtragem são executadas no lado do cliente, não havendo necessidade de recorrer a mais pedidos ao servidor. O modelo OData, no lado do servidor, apenas carrega os dados que foram pedidos ao servidor pela vista.

O OpenUI5 permite definir vários modelos em diferentes áreas da aplicação e atribuir outros apenas a um controlador. Permite, ainda, definir combinações de modelos como, por exemplo, definir um modelo JSON para a aplicação e definir o modelo OData para uma tabela contida na aplicação.

2.2.1.2 Ligação de dados

Como já foi referido o OpenUI5 utiliza a arquitetura MVC, que permite criar uma abstração entre o modelo, vista e controlador. A ligação de dados irá, portanto, definir como os modelos e as vistas comunicam entre si e qual o tipo de modelo ou modelos utilizados na aplicação [19]. A escolha dos modelos irá depender da fonte de dados externa utilizada. É possível, ainda, utilizar uma fonte de dados interna que pode ser usada para fins específicos, tal como o modelo de recursos e modelos das vistas, e ainda o modelo de dispositivos que pode ser utilizado para definições especificas de dispositivos. A representação da ligação de dados de uma aplicação está representada na Figura 3.

A partir da Figura 3 verifica-se, ainda, que a maior parte dos modelos estão localizados no lado do cliente, portanto, todos os dados são carregados para os modelos no momento da inicialização da aplicação. Todas as ações aplicadas sobre os dados apenas são executadas no lado do cliente, sendo necessário enviar essas mesmas ações para o lado do servidor.

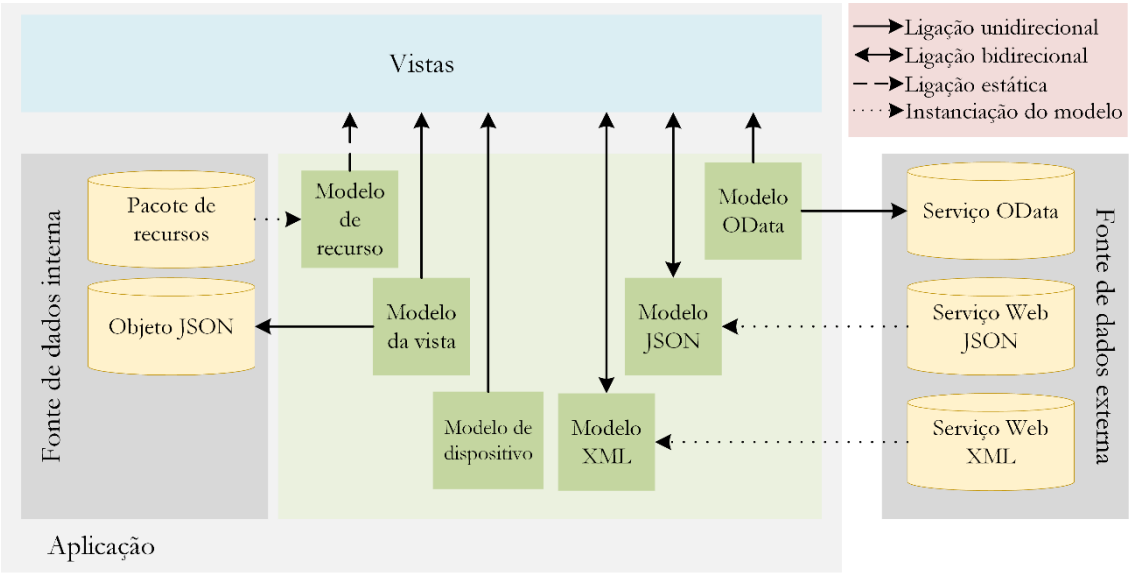


Figura 3: Representação das ligações de dados e dos modelos do OpenUI5.

Os modos de ligação suportados pelo OpenUI5 são [19]:

- Ligação **unidirecional**: apresenta uma ligação do modelo para a vista. Ao haver alterações nos dados, todas as ligações e a vista são atualizadas;

- Ligação **bidirecional**: é definido uma ligação do modelo para a vista e da vista para o modelo. A ocorrência de alterações no modelo e na vista, provoca, de forma automática, atualizações tanto para todas as ligações correspondentes como para o modelo e a vista.
- Ligação **estática**: significa do modelo para a vista apenas uma vez, não ocorrendo atualizações.

2.2.2 Node.js

O Node.js e a *framework* Express foram utilizados para a criação de uma API REST. Esta exigiu a utilização de pedidos Ajax para transportar os dados do lado do cliente para o lado do servidor.

O Node.js [20] é escrito na linguagem C++ e é executado num ambiente operacional JS, este foi desenhado para serviços de rede utilizando eventos assíncronos. Além disto, a lógica de funcionamento do Node.js requer o registo de uma função de retorno que são, também, executadas de forma assíncrona [21]. A utilização de funções assíncronas faz com que haja uma melhor utilização dos recursos do sistema criando, assim, uma melhoria a nível do desempenho.

Em [22] foi elaborado testes de desempenho às tecnologias web Node.js, PHP e Python-Web. Para estes testes foram utilizados diversos módulos teste, sendo um deles o "Hello World". As variáveis utilizadas para estes módulos foram o número médio de pedidos por segundo (Figura 4) e o tempo médio por pedido ao servidor por segundo (Figura 5), ambos em função do número de utilizadores. A partir das figuras 4 e 5, conclui-se que o Node.js possui um desempenho superior em comparação com as restantes tecnologias utilizadas neste estudo.

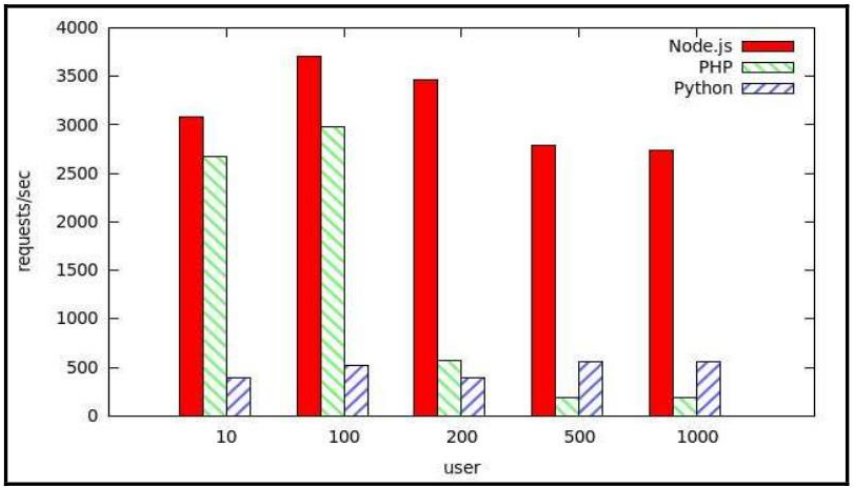


Figura 4: Resultados do número médio de pedidos por segundo.

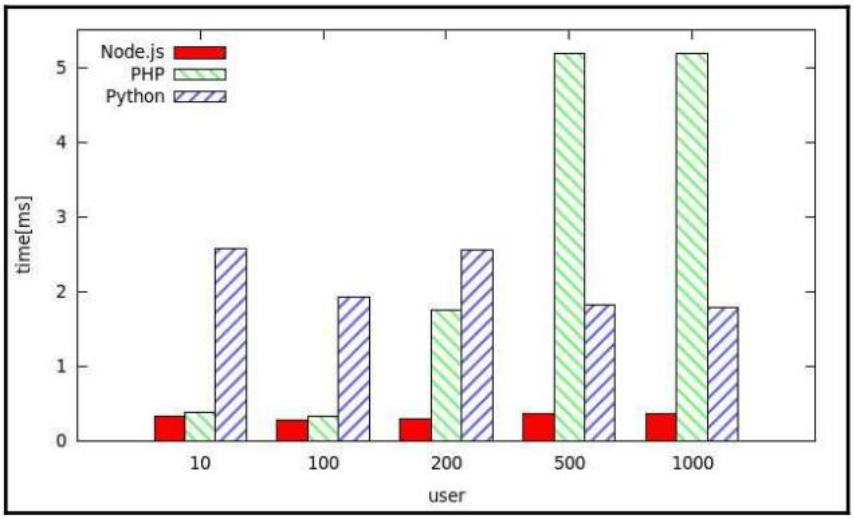


Figura 5: Resultados do tempo médio por pedido.

2.2.3 Microsoft Azure

O Microsoft Azure [23] foi utilizado para este projeto a fim que criar um serviço em *cloud* do projeto. O Azure foi escolhido por ser um dos líderes de mercado em serviços *cloud* e por apresentar uma grande variedade de variáveis de monitorização de serviços, algo que se revelou muito útil para os testes de desempenho do sistema.

2.3 Conclusão

Este capítulo é iniciado pela apresentação de um estudo sobre o processo de desenvolvimento *software*. Deste, retirou-se que os respetivos processos apesar de terem surgido há cerca de 40 anos atrás, hoje em dia ainda estão em constante desenvolvimento. Isto deve-se à complexidade da atividade de desenvolvimento que

envolve um conjunto de incertezas. Referiu-se que um dos problemas no desenvolvimento de software baseia-se nos três P's e que é necessário ter em conta o tipo de relação que é criada entre as pessoas e os processos. Constatou-se que algumas equipas preferem utilizar os métodos ágeis, em vez de processos de *software* tradicionais, focando-se, assim, em tarefas de desenvolvimento que aplicam práticas de acompanhamento. Esta secção de estudo termina com a lista de características que boas ferramentas de gestão de projetos devem possuir.

Logo após este estudo, é apresentado as tecnologias que foram utilizadas para o desenvolvimento deste sistema. O OpenUI5 é uma biblioteca em JS que fornece um conjunto de controles de interface utilizando o MVC e que possui algumas particularidades que requer algum estudo. O Node.js foi utilizado para criar a API REST deste sistema e que possui um grande desempenho, em comparação com algumas das ferramentas existentes no mercado. Os dados serão armazenados recorrendo a uma base de dados MySQL e, por fim, será utilizado o Microsoft Azure para criar um serviço em *cloud* do sistema.

3 FERRAMENTAS EXISTENTES

Nesta secção será apresentado algumas das ferramentas de apoio ao desenvolvimento de *software*, atualmente no mercado, através da apresentação das suas principais caraterísticas. No final do capítulo está presente uma comparação entre estas.

3.1 JIRA

A ferramenta JIRA[24] foi desenvolvida pela companhia australiana Atlassian. Esta permite a gestão de projetos [25] e tem como principais funcionalidades a monitorização de problemas que possam manifestar-se ao longo do desenvolvimento de um projeto de *software*.

Inicialmente, a utilização da ferramenta JIRA era, somente, para a captura e monitorização de problemas [26]. Só mais tarde, quando foi implementado o desenvolvimento de *software* ágil na organização, é que foram adicionados o uso de histórias de utilizador e a monitorização do progresso das tarefas para cada uma delas.

Esta ferramenta contém várias vistas, sendo uma delas um quadro de tarefas. Este quadro pode ser descrito através de uma lista de afazeres dividida em três colunas, a coluna "Por fazer", que indica quais as tarefas que estão por executar, a coluna "Em progresso" indica quais as tarefas que estão em execução e a coluna "Feito" é onde se encontra as tarefas que já estão terminadas. A partir destas colunas os utilizadores podem criar tarefas, ordená-las de acordo com a sua preferência e ver a quem foram atribuídas. Esta vista pode ser visualizada na Figura 6.

Além desta vista de tarefas existe, também, a vista de planeamento, que é indicada para o Scrum master. Esta permite obter uma vista geral sobre a lista de histórias de utilizador e suas respetivas tarefas por *sprint*. Podendo, ainda, atribuir tarefas aos membros de equipa.

Um dos pontos fracos desta ferramenta é a necessidade de haver maior suporte técnico em algumas das suas operações de gestão de *sprint* [27]. Esta não permite,

também, efetuar uma transferência de informação sobre a priorização de *sprint* ao longo das equipas.

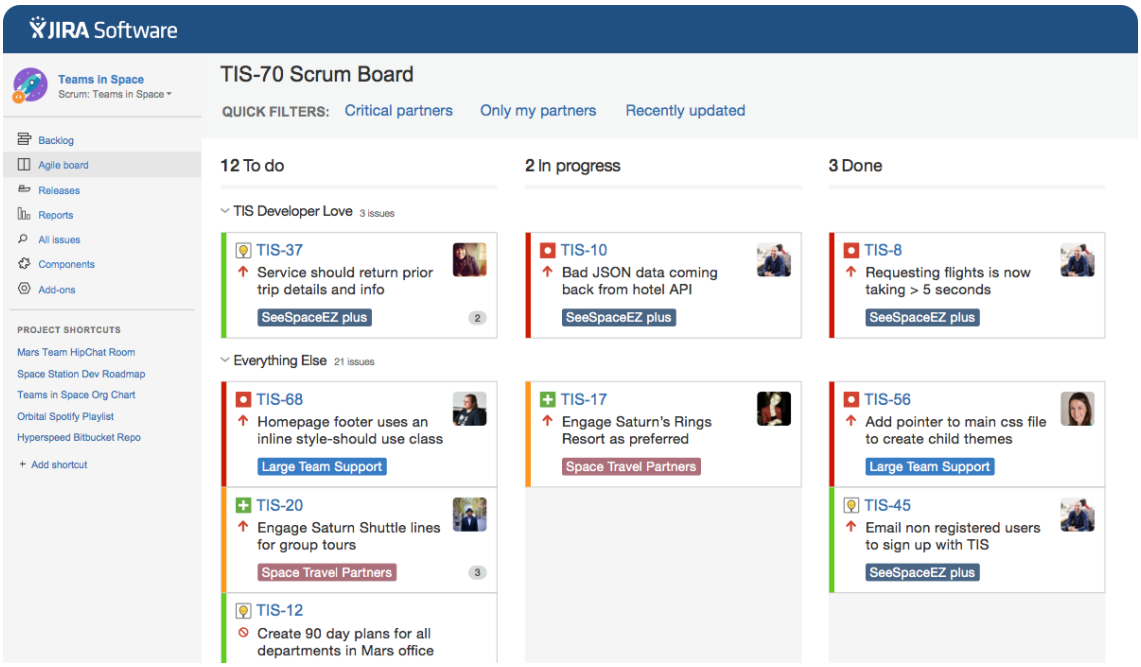


Figura 6: Vista do quadro de tarefas da ferramenta JIRA.

3.2 VersionOne

A ferramenta VersionOne baseia-se no desenvolvimento de *software* ágil, de forma a ajudar as empresas a abordar os seus clientes, monitorizar e reportar vários portefólios de *software* e projetos [28]. As suas funcionalidades estão interligadas com os métodos do Scrum e Kanban, recorrendo, desta forma, a quadros com objetivos, problemas e defeitos. Na Figura 7 pode ser visualizado a monitorização de Sprint desta ferramenta.

As suas caraterísticas torna-a numa ferramenta compreensiva e versátil, indicada para equipas e projetos de várias dimensões [29], sendo útil para grandes organizações. Desde o planeamento do portefólio até a monitorização e oferta de valor aos clientes, esta possibilita a entrega de produtos ao cliente e reduz o tempo do seu lançamento no mercado. O progresso e desempenho do projeto são informações que estão sempre visíveis, tal como, todas as informações necessárias para as tomadas de decisão acerca de alterações e problemas que possam surgir.

Esta ferramenta possui uma gestão de tarefas muito intuitiva e uma boa integração com outras aplicações, sendo estes um dos seus pontos mais fortes [30]. Referente a utilização móvel, pode-se apontar este como um ponto negativo desta ferramenta.

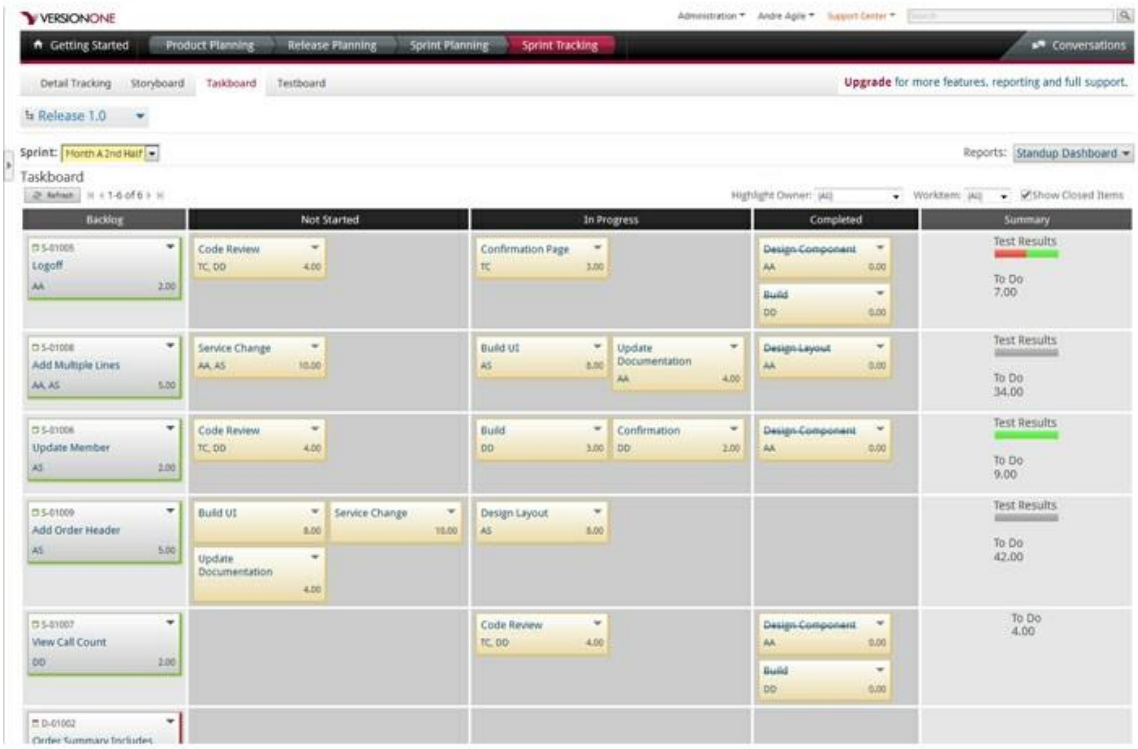


Figura 7: Vista de monitorização de *sprint* da ferramenta VersionOne.

3.3 Rally

A ferramenta Rally foi adquirida pela CA Technologies em 2015 [31]. Esta é considerada como uma plataforma ágil, com a capacidade de ajudar as organizações a definir estratégias, monitorizar projetos e gerir entregas. Fornece suporte para todas as fases de um projeto, desde o planeamento até ao seu lançamento. Possui uma hierarquia de projetos e de portfólios onde se pode visualizar a lista de tarefas, histórias de utilizador, trabalho realizado, progresso atual, etc. Possui uma gestão de equipas e suporte que permite identificar problemas, gerir várias equipas e utilizar o Scrum e o Kanban. Utiliza o lançamento de relatórios, métricas e gráficos, a fim de medir a produtividade, a capacidade de resposta e a qualidade do produto. Referente a troca de informação, esta integra um chat e uma caixa de entrada com emails e

notificações. Na Figura 8 está representada a vista dos quadros Kanban desta ferramenta.

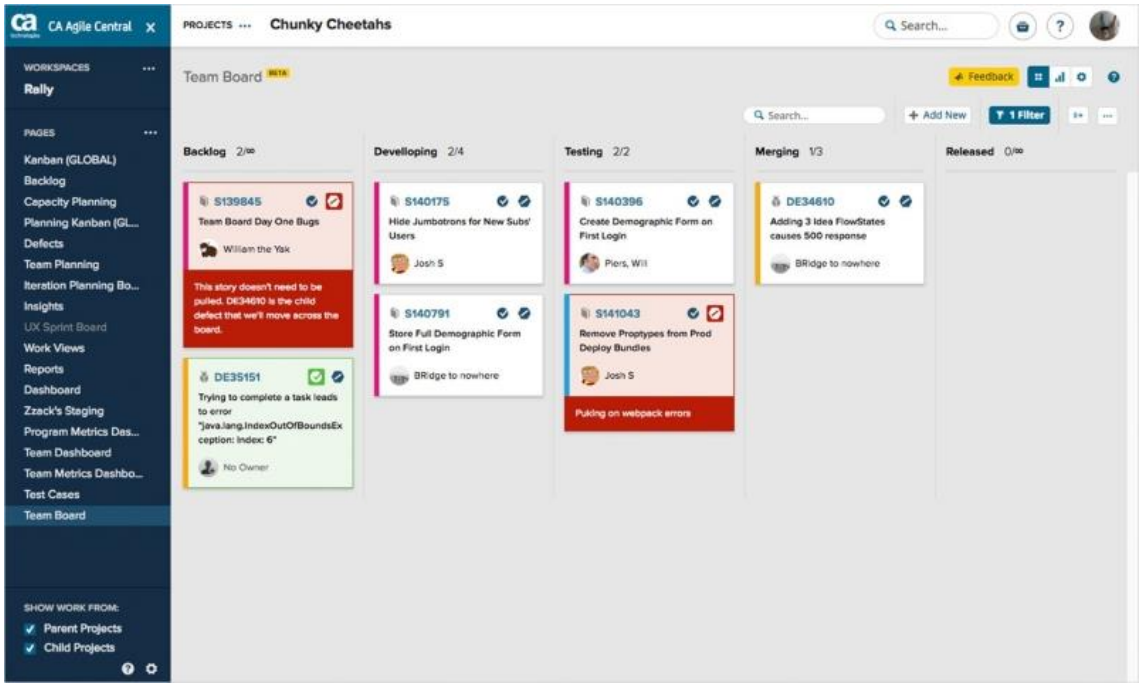


Figura 8: Vista dos quadros Kanban da equipa da ferramenta Rally.

3.4 Wrike

A ferramenta Wrike detém o prémio de melhor ferramenta de gestão de projetos de 2018. Foi desenhada com o propósito de melhorar a eficiência e a velocidade na criação de projetos, tanto em grupos locais como em distribuídos [32]. Os pontos fortes desta aplicação residem na flexibilidade em permitir que grupos multifuncionais produzam um produto, de forma eficaz, a partir de um único local. Permite a priorização de pedidos e atribuí-los de uma forma ágil, mantendo todos os intervenientes focados nas suas tarefas.

Esta ferramenta apresenta um *layout* com três painéis principais, que permitem observar os dados mais importantes num único ecrã, mostrar uma hierarquia de projetos e apresentar as listas de tarefas e os detalhes de cada tarefa (Figura 9). Faz uso de uma estrutura de pastas e etiquetas, permitindo visualizar e ordenar projetos e visualizar do progresso das atividades, por todos os membros da equipa. A funcionalidade das tarefas permite que um projeto seja dividido em tarefas, que por

sua vez podem ser divididas em subtarefas. Esta tolera a criação de relatórios, onde é possível visualizar os dados do projeto em oito categorias, obter gráficos de desempenho e métricas de estado das tarefas. Além disto a Wrike possui relatórios de auditoria para examinar ameaças e proteger os dados e operações.

A ferramenta Wrike oferece cinco planos de preços: Plan, Professional, Business, Marketers e Enterprise.

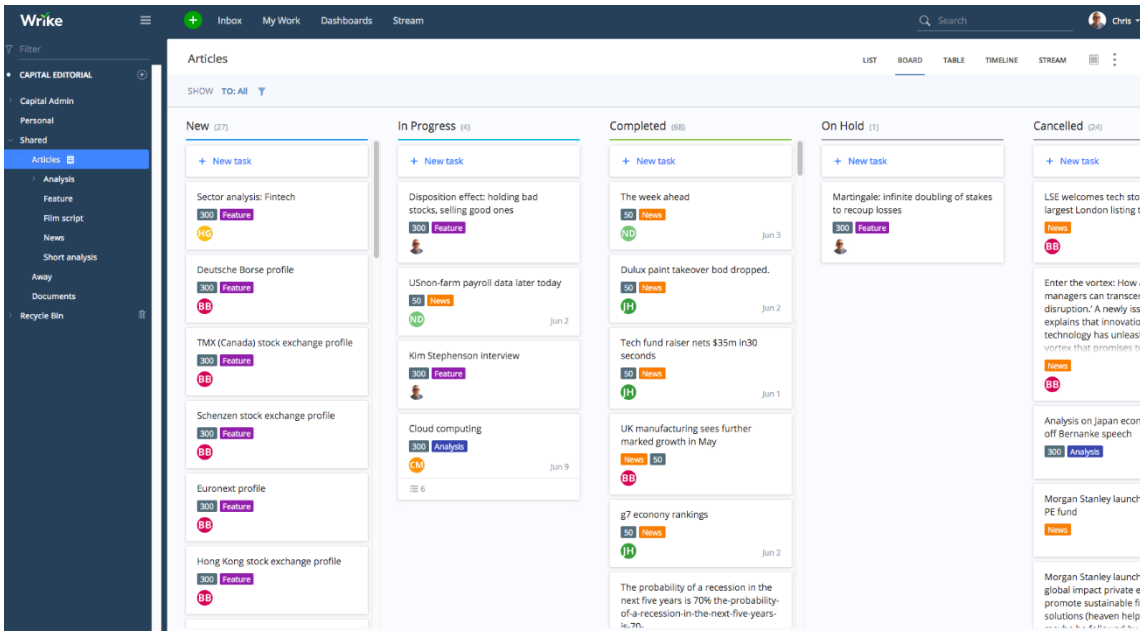


Figura 9: Vista com as listas de tarefas na ferramenta Wrike.

3.5 Planner

O Planner [33] é um produto do Office 365 baseado em quadros Kanban (Figura 10). Este foi desenvolvido a pensar em equipas que queiram gerir o seu trabalho, juntamente com a necessidade de trocar informações entre si. Apesar de esta ser uma ferramenta muito fácil de utilizar é comum ser utilizada apenas por utilizadores que possuem uma conta do Office 365, pois esta necessita de uma subscrição deste serviço.

No Planner cada cartão possui os detalhes da tarefa, tal como o título, estado, descrição, atribuição, data de entrega, etc. Está disponível a análise a partir de gráficos, que mostram uma visão geral sobre os estados das tarefas e quais as tarefas atribuídas a cada um dos membros.

A falta de uma barra de procura de tarefas é umas das desvantagens que se pode enumerar desta ferramenta. Esta é uma ferramenta que deve ser utilizada para projetos de pequena dimensão, pois para grandes projetos torna-se difícil a procura de tarefas.

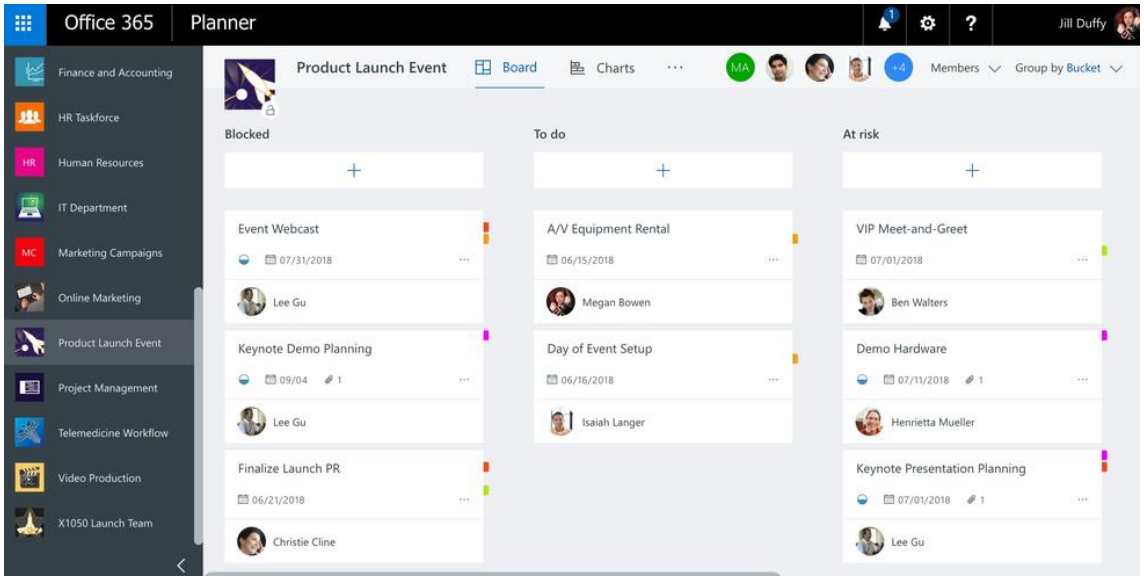


Figura 10: Vista dos quadros da ferramenta Planner.

3.6 Trello

O Trello é um dos líderes em aplicações de gestão de projetos [34]. Este é uma aplicação *web* baseada no método Kanban [35], utilizando o conceito de quadros para representar projetos e cartões que são constituídos por listas para representar tarefas (Figura 11). Permite a colaboração entre equipas e atribuir cartões a membros, organizando e monitorizando todas as tarefas, ficheiros e informações. Esta aplicação permite a integração com outras aplicações, por exemplo, após a atribuição do tempo de conclusão de uma tarefa é possível a integração com Gmail ou o Outlook. Esta é, ainda, considerada como uma das ferramentas mais atrativas do mercado.

Relativamente aos pontos negativos desta aplicação, um dos que se destaca é o facto de não haver uma forma direta para marcar uma tarefa como concluída. O que é habitual é criar uma etiqueta de cor verde, para representar a etapa de conclusão de uma tarefa, mas isto, no caso do trabalho em equipa, requer a ação de informar os restantes utilizadores desta característica. Um outro defeito a anotar é a

impossibilidade de remover a informação de que o prazo de entrega de uma tarefa excedeu a data limite.

Para utilizar esta aplicação os utilizadores dispõem de três possibilidades de preço. A versão básica, referida como "Free", que permite criar quadros, fazer a integração com outras aplicações, criar votações e adicionar ficheiros até 10MB. A versão Business Class permite observações gerais de equipas, integração com outras aplicações e carregamento de ficheiros até 250 MB. A versão Enterprise é indicada para grandes empresas que façam a gestão de vários projetos e equipas.

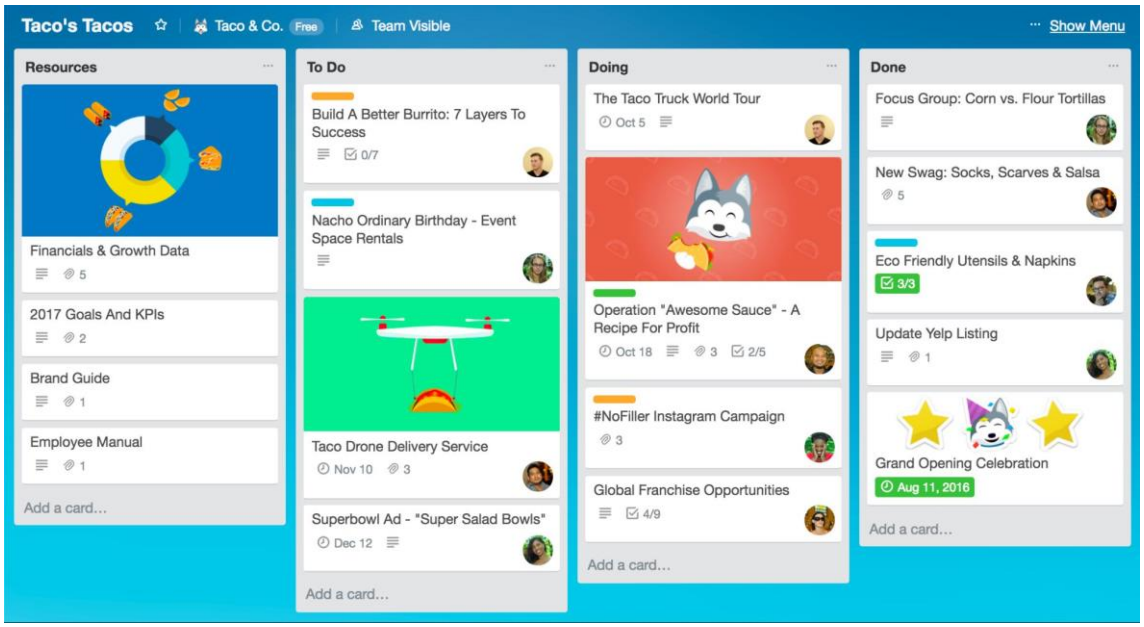


Figura 11: Listas de cartões da aplicação Trello.

3.7 Asana

A ferramenta Asana [36] tem como objetivo melhorar o trabalho em equipa, focando-se nos objetivos do projeto e nas tarefas diárias. Tal como as ferramentas anteriores esta utiliza, também, o Kanban (Figura 12). Esta contém as funcionalidades de criação de tarefas e a definição de entregas priorizadas, permite a atribuição de tarefas e a troca de informação entre membros de equipa. Possui um plano de projeto, onde é apresentado cada passo do desenvolvimento do projeto e qual o prazo de entrega previsto. Nesta ferramenta é, também, possível o carregamento de ficheiros.

Uma das vantagens desta ferramenta é a criação de relações entre tarefas e de possuir uma abordagem de colaboração que permite a todos os membros de uma

equipa partilharem toda informação num único lugar, apresentando um design semelhante a um *feed* de atividade utilizado em redes sociais. Esta ferramenta [37] garante a segurança nas conexões a partir da utilização de protocolos como o TLS 1.1 e guarda os dados sob o protocolo "secure SSAE 16" da Amazon. Um dos poucos defeitos que se pode nomear desta ferramenta é o facto de que, nas primeiras interações com a ferramenta, esta poderá transmitir a sensação de possuir uma má estrutura.

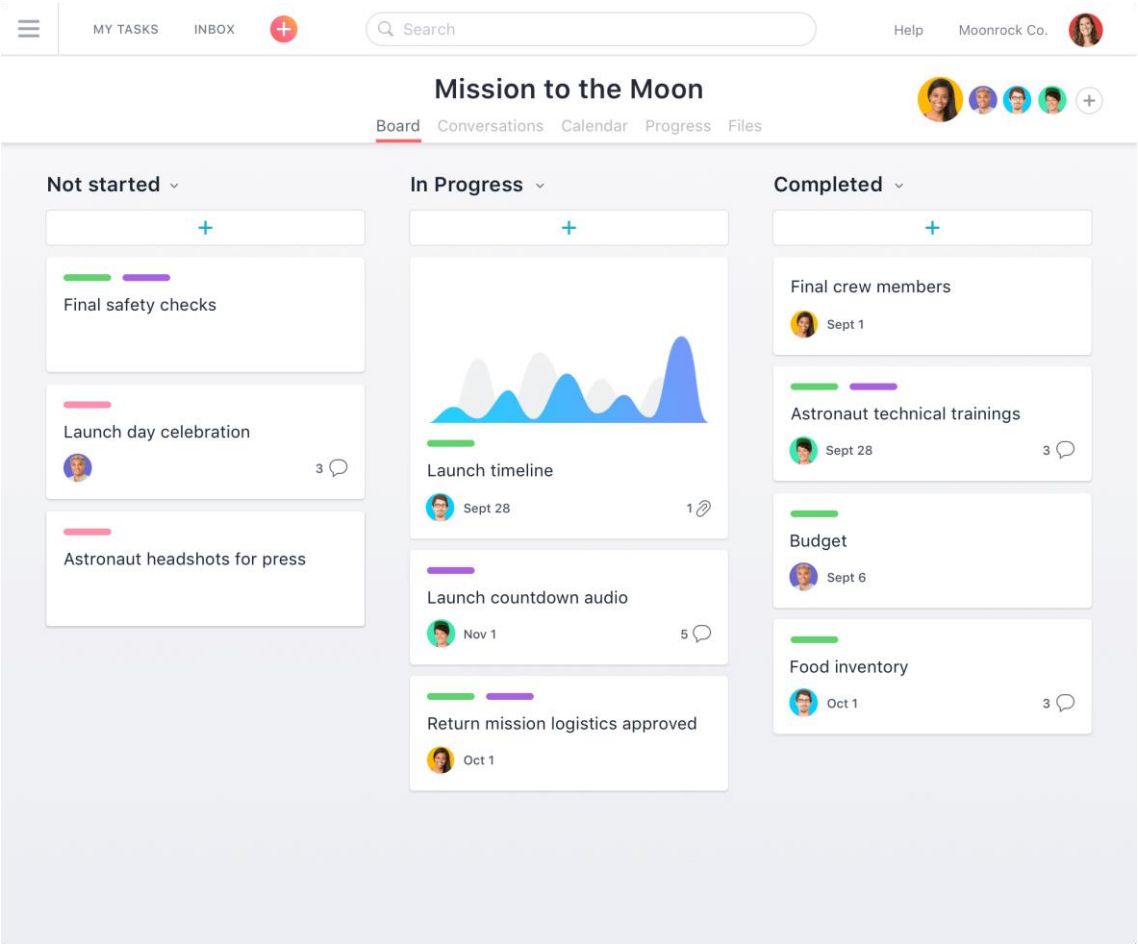


Figura 12: Vista dos quadros de tarefas da ferramenta Asana.

3.8 Conclusão

Este capítulo permitiu obter algum conhecimento sobre as ferramentas de gestão de projetos que existem atualmente no mercado. Estas podem-se basear em métodos ágeis, tal como as ferramentas JIRA, VersionOne, Rally e Wrike ou em soluções mais básicas de colaboração como o Planner, Trello e o Asana. Notou-se

que existe uma grande variedade deste tipo de ferramentas, mas que apresentam soluções muito comuns entre si, havendo maior diferenciação nas aplicações integradas e na complexidade de gestão de equipas. Além disto, retirou-se alguns conceitos e aspetos importantes que serão uteis para o desenvolvimento deste projeto, tal como a utilização de quadros Kanban e recorrer à utilização de gráficos. Foi ainda efetuado um outro estudo sobre o Kanban e o Scrum apresentado no Anexo A.

As ferramentas acima mencionadas podem ser divididas em ferramentas de desenvolvimento de *software* ágil e em ferramentas mais simples de colaboração. Nas tabelas Tabela 1 e Tabela 2 está presente uma comparação entre as ferramentas de cada tipo respetivamente.

Tabela 1: Tabela de comparação das ferramentas de desenvolvimento de software ágil.

	Jira	VersionOne	Rally	Wrike
Licença	Proprietário/ teste grátis	Proprietário/teste grátis	Proprietário/teste grátis/ grátis	Proprietário/ teste grátis/ grátis
Plataforma	Instalação/Web	Web	Web	Instalação/Web
Utilizadores finais	Pequenas, médias e grandes empresas	Médias e grandes empresas	Pequenas, médias e grandes empresas	Pequenas, médias e grandes empresas
Monitorização de tarefas	Sim	Sim	Sim	Sim
Relatórios	Sim	Sim	Sim	Sim
Gráfico *burn down*	Sim	Sim	Sim	Sim
Gestão de testes	Suporte parcial	Suporte total	Suporte total	Suporte parcial
Kanban	Suporte total	Suporte total	Suporte total	Suporte total
Scrum	Suporte total	Suporte total	Suporte total	Suporte total
XP	Suporte parcial	Suporte total	Suporte parcial	Suporte parcial
Integração com outras aplicações	Sim	Sim	Sim	Sim

Tabela 2: Tabela de comparação das ferramentas mais simples de colaboração.

	Planner	Trello	Asana
Licença	Proprietário/ teste grátis	Proprietário /teste grátis/ grátis	Proprietário/ teste grátis/ grátis
Plataforma	Web-Based	Instalação/Web-Based	Instalação/Web-Based
Utilizadores finais	Pequenas e médias empresas	Pequenas, médias e grandes empresas	Pequenas, médias e grandes empresas
Kanban	Sim	Sim	Sim
Monitorização de tarefas	Sim	Sim	Sim
Relatórios	Não	Não	Sim
Gráfico *burn down*	Não	Não	Não
Integração com outras aplicações	Sim	Sim	Sim

4 ANÁLISE

Neste capítulo será apresentado toda a análise do projeto, nomeadamente, será detalhado a fase de levantamento dos requisitos e apresentado todo o desenho de *software* efetuado para este projeto.

4.1 Requisitos funcionais

Os requisitos funcionais foram apresentados pela empresa Odkas e através de estudos de sistemas de aplicações de gestão de projetos. Estes foram utilizados para definir as funcionalidades do sistema e estão descritos na Tabela 3.

Tabela 3: Tabela de requisitos funcionais.

Nº	Requisito funcional
RF01	Adicionar e visualizar utilizadores, clientes, equipas, projetos e requisitos.
RF02	Escolher um líder e membros aquando a criação das equipas.
RF03	Validar os projetos e requisitos.
RF04	Criar listas de tarefas.
RF05	Criar e atribuir tarefas.
RF06	Alertar os membros de novas tarefas.
RF07	Adicionar uma descrição às tarefas.
RF08	Definir prazo para as tarefas.
RF09	Associar requisitos a tarefas.
RF10	Modificar o estado das tarefas.
RF11	Associar os requisitos ao projeto.
RF12	Modificar o estado dos projetos e requisitos.

4.2 Requisitos não funcionais

Os requisitos não funcionais foram definidos de acordo com cenários de desempenho, segurança, usabilidade e disponibilidade. Estes serão utilizados para definir a forma como os utilizadores terão acesso as funcionalidades da aplicação. Os requisitos não funcionais estão expostos na Tabela 4, Tabela 9, Tabela 12 e Tabela 14.

4.2.1 Cenários de desempenho

Os cenários de desempenho apresentados na Tabela 4 e Tabela 5 permitiram adquirir os requisitos não funcionais descritos na Tabela 6.

Tabela 4: Cenário de desempenho na atribuição de equipas.

Parte do cenário	Valores possíveis
Fonte	Líder de equipa.
Estímulo	Atribuição de tarefas.
Ambiente	Operação normal.
Artefactos	Gestor de tarefas e de notificação.
Resposta	Identifica o(s) utilizadores(s) e envia a notificação.
Medida de resposta	Envio e receção da notificação em menos de 5 segundos.

Tabela 5: Cenário de desempenho na procura de projetos.

Parte do cenário	Valores possíveis
Fonte	Utilizador válido do sistema.
Estímulo	Escrever o nome do projeto no campo de procura.
Ambiente	Operação normal.
Artefactos	Gestor de projetos.
Resposta	Identifica e mostra o projeto procurado.
Medida de resposta	A pesquisa do projeto deve demorar no máximo 3 segundos.

Tabela 6: Tabela dos requisitos de desempenho.

Nº	Requisito
RNF01	O resultado de uma pesquisa de um projeto não deve demorar mais do que 3 segundos
RNF02	Um alerta deve ser recebido no máximo até 5 segundos após o seu envio.

4.2.2 Cenários de segurança

Os cenários de segurança são apresentados na Tabela 7 e Tabela 8 e os requisitos descritos na Tabela 9.

Tabela 7: Cenário de segurança na tentativa de acesso aos dados do sistema por um utilizador desconhecido.

Parte do cenário	Valores possíveis
Fonte	Utilizador desconhecido.
Estímulo	Tentativa não autorizada para obter dados.
Ambiente	Operação normal.
Artefactos	Dados do sistema.
Resposta	Os dados e serviços são protegidos do acesso não autorizado.
Medida de resposta	É mostrado uma mensagem de acesso não autorizado.

Tabela 8: Cenário de segurança na tentativa de alterar aos dados do sistema por um utilizador conhecido.

Parte do cenário	Valores possíveis
Fonte	Utilizador conhecido.
Estímulo	Tentativa não autorizada para alterar dados.
Ambiente	Operação normal.
Artefactos	Dados do sistema.
Resposta	Os dados e serviços são protegidos do acesso não autorizado.
Medida de resposta	É mostrado uma mensagem de acesso não autorizado.

Tabela 9: Tabela de requisitos de segurança.

Nº	Requisito
RNF03	O sistema deverá possuir 3 perfis funcionais, nomeadamente, o administrador, líder de equipa e membro de equipa
RNF04	Não permitir que os dados/ações sejam acedidos/realizadas por entidades não autorizadas
RNF05	A palavra-passe do utilizador deve de ser encriptada.

4.2.3 Cenários de usabilidade

Os cenários de usabilidade estão especificados na Tabela 10 e Tabela 11 e os requisitos na Tabela 12.

Tabela 10: Cenário de usabilidade para a adição de tarefas.

Parte do cenário	Valores possíveis
Fonte	Utilizador final.
Estímulo	O utilizador final pretende adicionar uma nova lista de tarefas a um projeto.
Ambiente	Em tempo de execução.
Artefactos	Gestor de tarefas
Resposta	Apresentar vista para ver/adicionar as tarefas de um projeto com todas as suas funcionalidades
Medida de resposta	Obter sucesso na criação da lista de tarefas, mostrar ocorrência de erros (no caso de haver erros) e não haver perda de dados.

Tabela 11: Cenário de usabilidade para a receção de uma notificação.

Parte do cenário	Valores possíveis
Fonte	Utilizador final.
Estímulo	O utilizador final recebe uma notificação.
Ambiente	Em tempo de execução.
Artefactos	Gestor de notificação
Resposta	Apresenta a notificação do tipo *pop up*.
Medida de resposta	Mostra a notificação *pop up* durante 3 segundos e adiciona a notificação ao modelo de dados das notificações não lidas.

Tabela 12: Requisitos de usabilidade.

Nº	Requisito
RNF06	Utilizar notificações do tipo *pop up*.
RNF07	Deve ser possível adicionar novas tarefas utilizando no máximo 3 interações (cliques) com o rato.
RNF08	O sistema deverá ter validações de formulário.
RNF09	Deve ser possível aceder à plataforma em diferentes sistemas operativos.

4.2.4 Cenários de disponibilidade

Os cenários de disponibilidade estão descritos na Tabela 13 e o requisito correspondente na Tabela 14.

Tabela 13: Cenário de disponibilidade para quando o servidor é encerrado e existe sessões iniciadas.

Parte do cenário	Valores possíveis
Fonte	Servidor
Estímulo	Deixa de funcionar.
Ambiente	Operação normal.
Artefactos	Gestor de autenticação.
Resposta	Guardar dados da sessão na base de dados.
Medida de resposta	Carregar dados das sessões iniciadas assim que o servidor voltar a estar em funcionamento.

Tabela 14: Tabela de requisitos de disponibilidade.

Nº	Requisito
RNF10	Guardar dados de sessão

4.3 Casos de utilização

Os diagramas de caso de utilização são utilizados para descrever as ações que cada tipo de utilizador irá desempenhar num sistema, em colaboração com outros utilizadores. Para este sistema identificou-se as principais ações que cada tipo de utilizador irá desempenhar e a partir destas criou-se o diagrama representado na Figura 13. Os utilizadores que participam nas ações do sistema são o administrador, o líder de equipa e o membro de equipa.

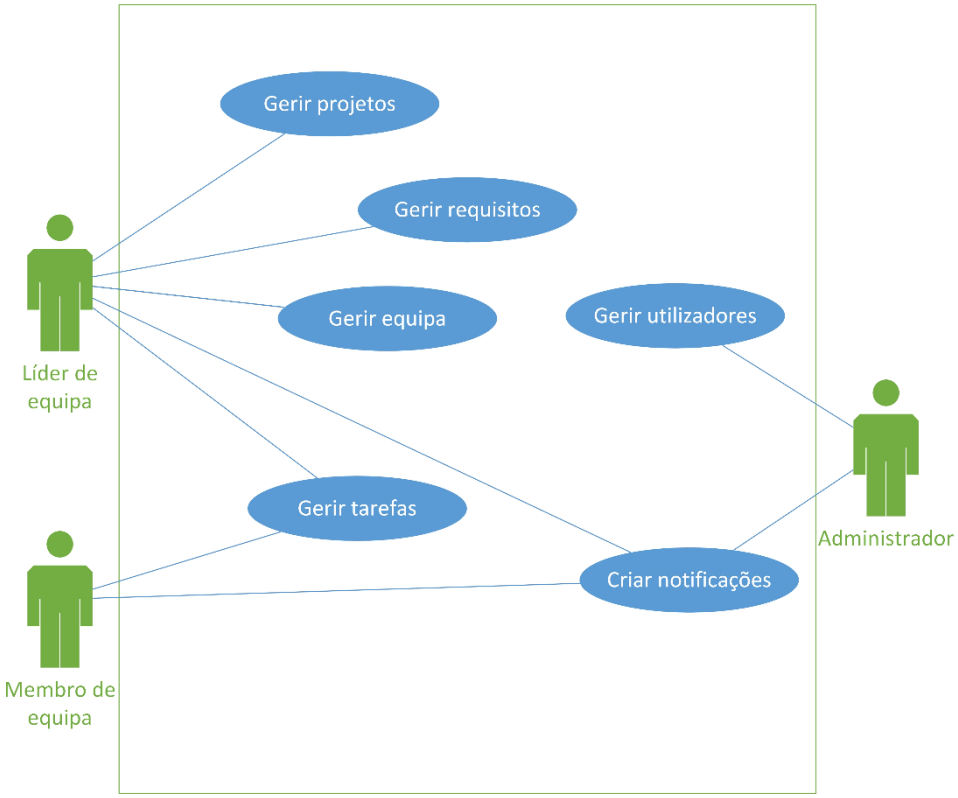


Figura 13: Diagrama de casos de utilização.

4.4 Diagramas de robustez

Segundo [38], os diagramas de robustez são semelhantes aos diagramas de colaboração UML, identificando quais os objetos utilizados num sistema e como estes interagem entre si. De uma forma geral, estes diagramas apresentam-se como um intermediário entre a fase de levantamento de requisitos e a faze de desenho de software, identificando quais os objetos e como estes interagem entre si durante a execução do sistema.

Na Figura 14 está representado o diagrama de robustez referente a gestão de equipas. Neste verifica-se que na gestão de equipas irá participar dois tipos de utilizadores, o líder de equipa e o membro de equipa. Para esta gestão são necessárias duas vistas distintas para cada utilizador, pois, o líder de equipa irá criar equipas e adicionar elementos a equipa, enquanto que o membro de equipa apenas irá visualizar a equipa e todos os elementos relacionados com esta, nomeadamente, os membros e projetos associados. Este gestor será, ainda, responsável pela criação de notificações, a fim, de notificar os membros aquando a sua adição na equipa e também na

associação de projetos a equipa. Para controlar o acesso às operações de cada tipo utilizador é utilizado permissões.

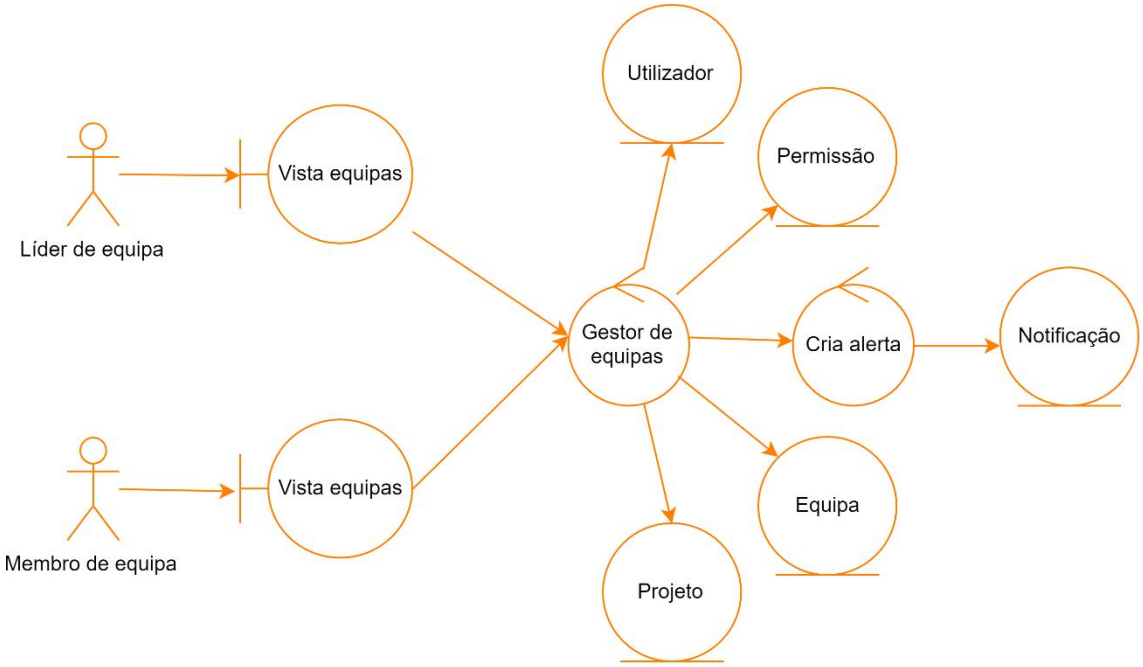


Figura 14: Diagrama de robustez para a gestão de equipas.

Referente a gestão de tarefas tanto o líder de equipa como o membro de equipa podem fazer toda a gestão de tarefas (criar, eliminar, editar, atribuir membros, definir intervalo de datas e associar requisitos). Tendo isto em consideração, apenas é necessária uma vista para ambos os tipos de utilizador. Esta vista terá um controlador que será responsável por criar, eliminar e editar tarefas. Tarefas estas que poderão ser associadas a membros da equipa e a requisitos. As notificações são criadas aquando a atribuição de membros à tarefa e, além disto, é necessário bloquear o acesso a utilizadores fora do sistema a esta gestão, recorrendo por isso a permissões. O diagrama de robustez da gestão de tarefas está representado na Figura 15.

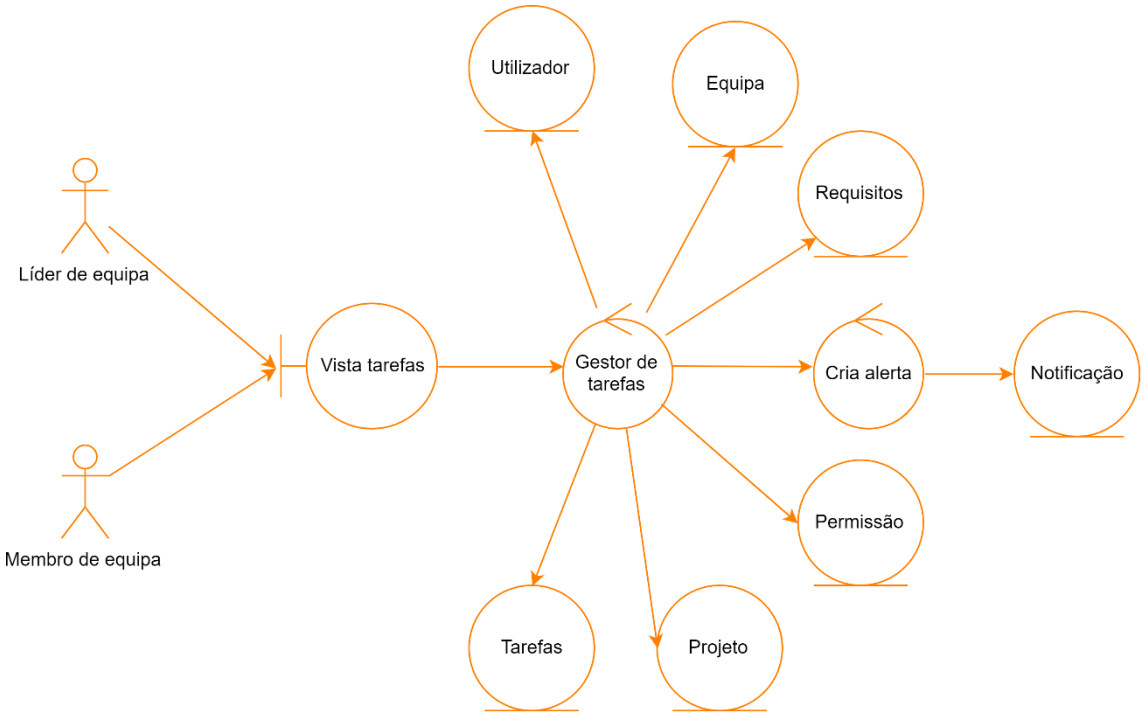


Figura 15: Diagrama de robustez para a gestão de tarefas.

A gestão de requisitos está representada no diagrama de robustez da Figura 16. O líder de equipa tem como possíveis ações a adição de requisitos a um projeto e alterar os estados dos requisitos. O membro de equipa está restrito apenas a visualização dos requisitos e das tarefas associadas a estes.

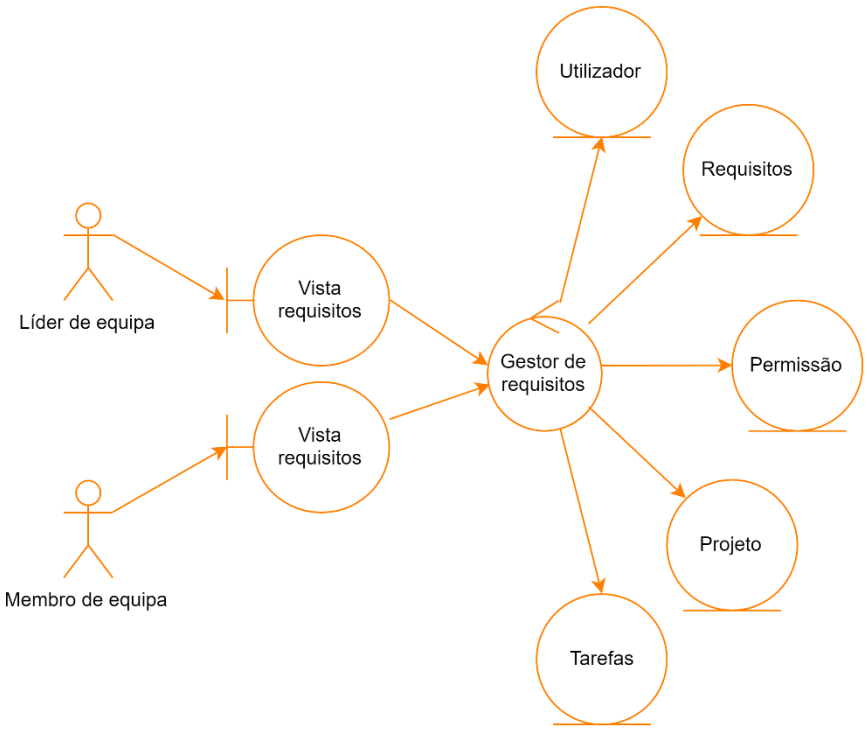


Figura 16: Diagrama de robustez da gestão de requisitos.

Para a gestão de projetos tem-se novamente duas vistas. Uma vista para o líder de equipa em que pode adicionar projetos e editá-los, associar uma equipa ao projeto e modificar o estado do projeto e dos requisitos. Referente a vista do membro de equipa, apenas está disponível a leitura das entidades antes referidas e, também, do cliente. De modo a criar esta separação de funcionalidades entre utilizadores, utilizou-se as permissões. Nesta gestão, uma notificação é desencadeada quando uma equipa é adicionada ao projeto. Na Figura 17 está esquematizado o diagrama de robustez da gestão de projetos.

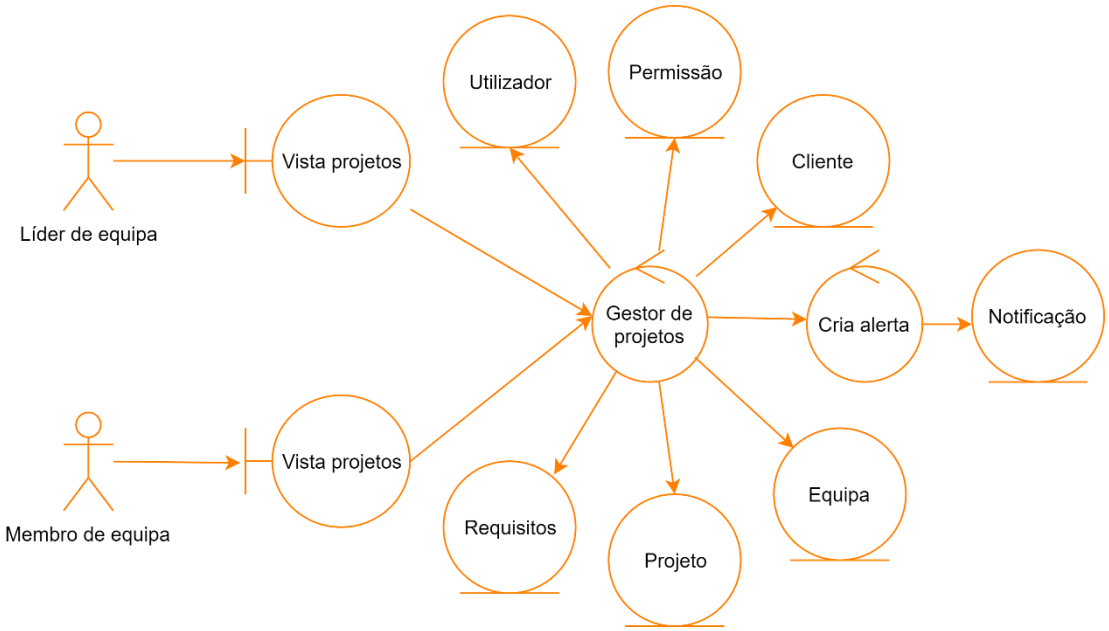


Figura 17: Diagrama de robustez da gestão de projetos.

Referente a gestão de utilizadores, esta é gerida apenas por um utilizador (administrador) sendo, portanto, constituída apenas por uma vista. As funcionalidades desta será adicionar e eliminar utilizadores do sistema. Ao adicionar o utilizador, dependendo do tipo definido, será criado permissões para este. A partir destes fundamentos elaborou-se o diagrama de robustez representado na Figura 18.

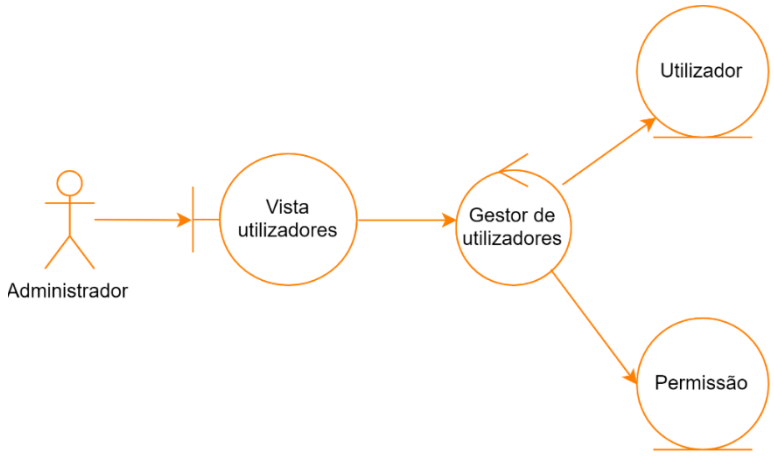


Figura 18: Diagrama de robustez para a gestão de utilizadores.

A vista de iniciar sessão será utilizada por todos os tipos de utilizadores do sistema e será controlada por um gestor de autenticação, que irá verificar se o utilizador tem permissões de acesso. Caso seja um utilizador válido a sua sessão será guardada numa entidade denominada de sessão. Na Figura 19 pode-se visualizar o diagrama de robustez correspondente a gestão de autenticação.

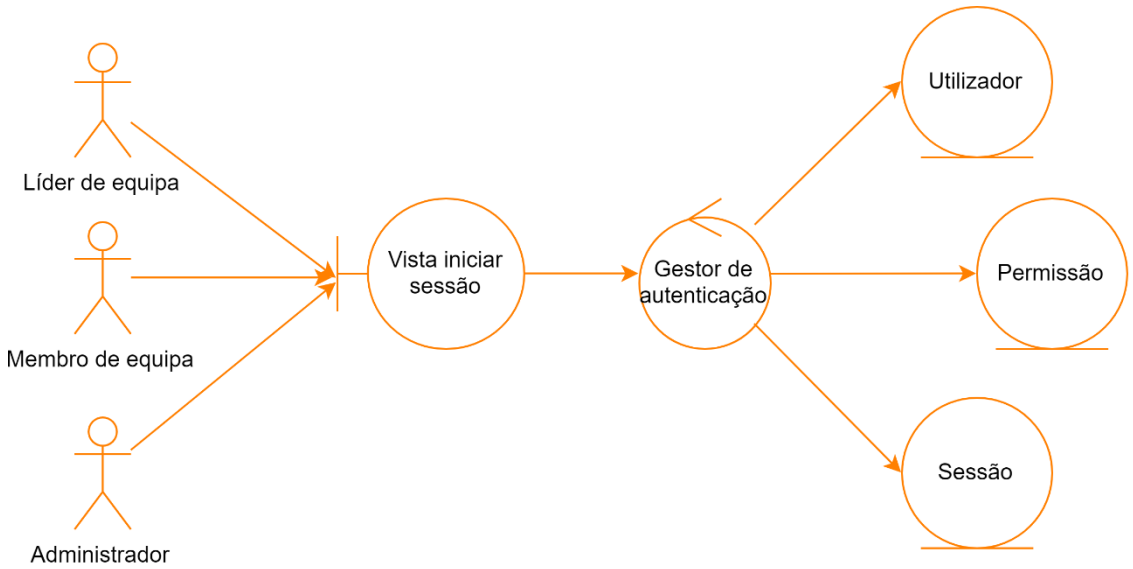


Figura 19: Diagrama de robustez para a gestão de autenticação.

4.5 Modelo de dados

O modelo de dados representa as entidades e as relações entre elas. Como o modelo de dados completo possui uma dimensão ilegível em relação à escala da página do documento, decidiu-se dividi-lo por secções. Estas secções foram denominadas de

projeto, notificação e permissões e estão esquematizadas da Figura 20 à Figura 22. No Anexo B pode-se visualizar o modelo relacional baseado nestas secções do modelo de dados.

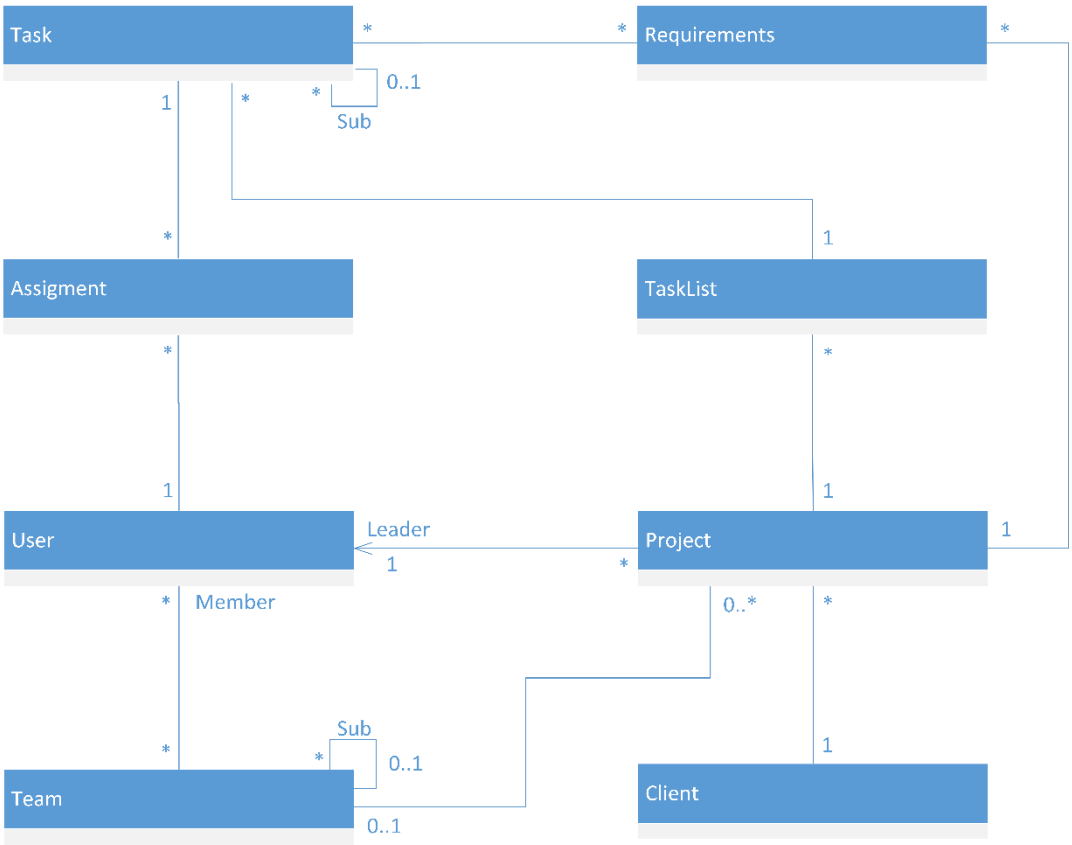


Figura 20: Secção do projeto no modelo de dados.

Para este sistema a construção do modelo de dados iniciou-se com a construção das entidades principais do sistema e suas relações (projeto, requisitos, tarefas, equipas, utilizadores, notificações e clientes). O modelo de dados da secção projeto está representado na figura 20.

Na Figura 21 está representado um modelo de dados para a secção das permissões. Este é constituído pelas entidades base "Item", "Entity" e "Permission" e pode ser considerado como um modelo versátil, que pode ser implementado noutro sistema. Este surge da ideia de que uma entidade, no caso deste sistema pode ser um grupo de utilizadores ou só um utilizador, possuir certas permissões sobre um determinado item, por exemplo, um projeto.

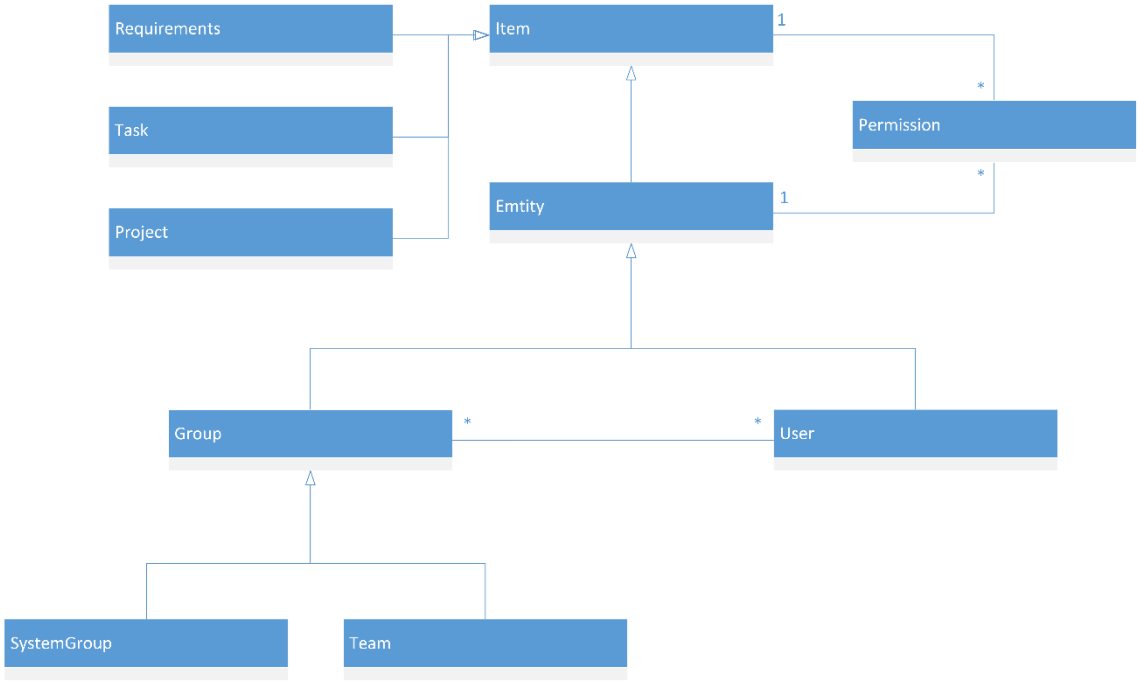


Figura 21: Secção das permissões no modelo de dados.

Na Figura 22 está representada a secção do modelo de dados referente as notificações, que inclui as entidades "Notification", "Subscription" e "User". Um utilizador possui uma relação de 1:n para o envio de notificações e a relação n:n para a receção de notificações.

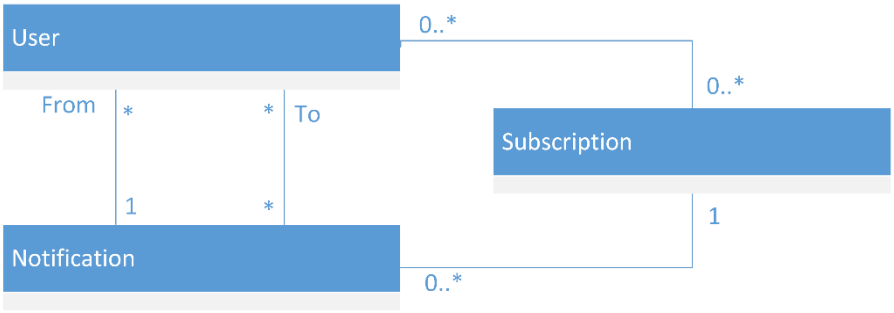


Figura 22: Secção das notificações no modelo de dados.

4.6 Diagramas de estado

Os diagramas de estado [39] representam o comportamento de uma parte do sistema recorrendo a transições finitas de estado.

Para este sistema foi elaborado diagramas de estado para os componentes projetos, tarefas, requisitos e notificações. Os estados destes componentes estão

esquematizados da Figura 23 à Figura 26 e surgem de uma análise ao comportamento que estes poderão ter ao longo do seu processo de desenvolvimento. Neste sistema um projeto ao ser submetido toma o estado de submetido e fica a aguardar a submissão dos requisitos. Os requisitos depois de adicionados terão que ser validados e no caso serem inválidos será necessário efetuar uma nova submissão destes, no caso de serem válidos passam a fase de implementação. Após a fase de implementação vem a fase de testes, onde poderão ser aprovados chegando, assim, a fase final de entrega ou no caso de falha, voltar ao estado de implementação. Caso o utilizador deseje eliminar um projeto este irá tomar o estado de eliminado. Na Figura 23 é apresentado o diagrama de estados dos projetos.

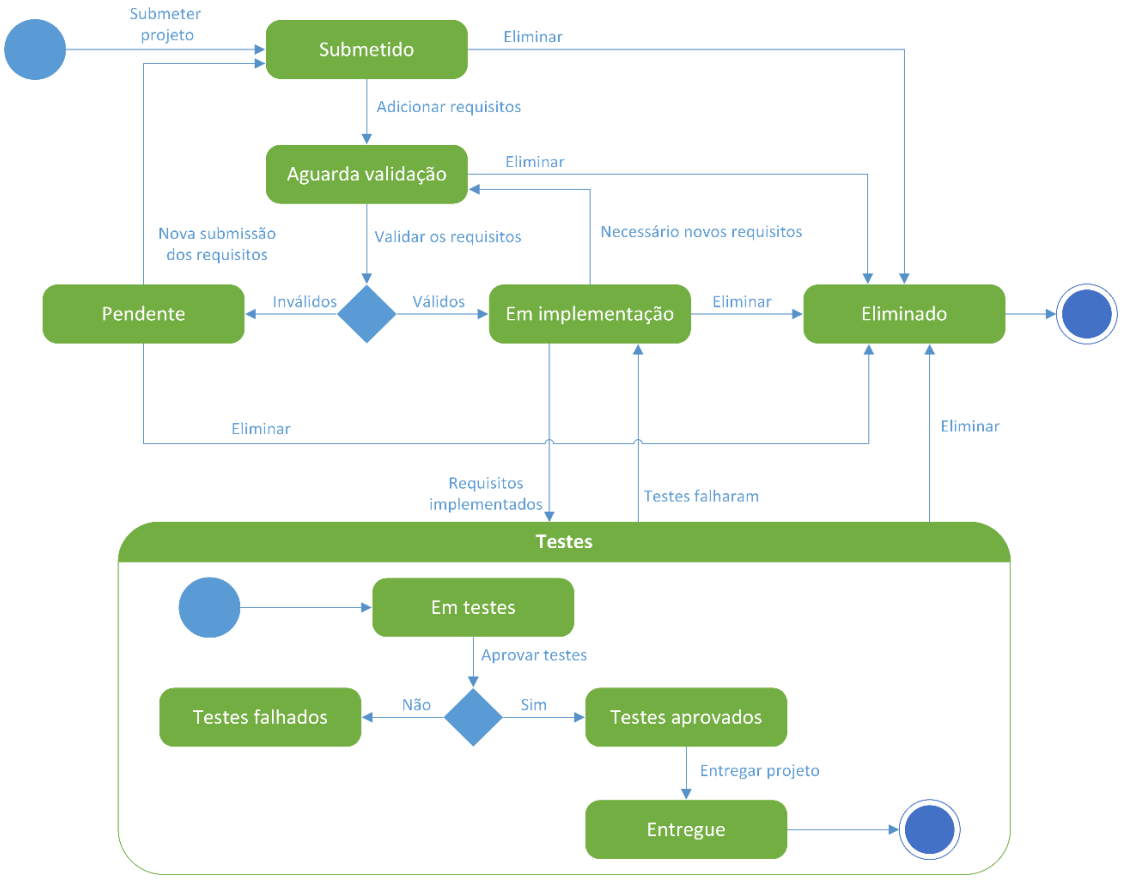


Figura 23: Diagrama de estados dos projetos.

Referente aos requisitos, após a sua criação necessitam de ser validados. A próxima tarefa após a validação é atribuir tarefas passando, desta forma, a fase de implementação e por fim passam à fase de conclusão. Este comportamento pode ser visualizado na Figura 24.

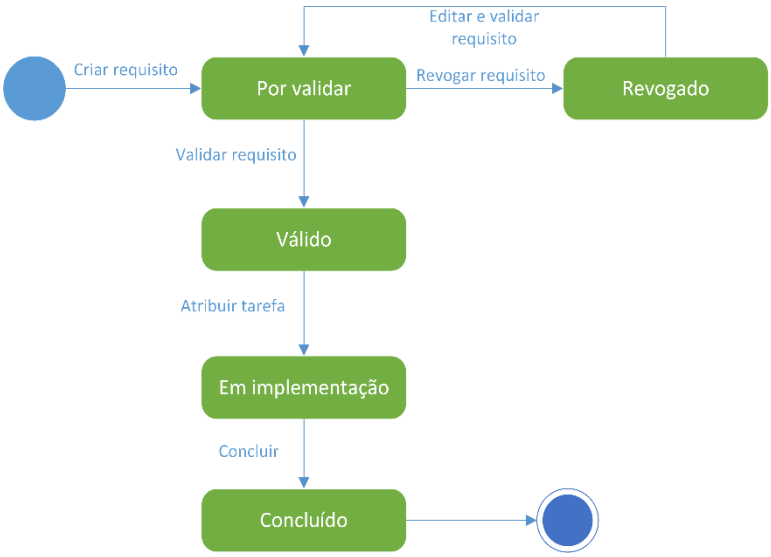


Figura 24: Diagrama de estados dos requisitos.

No caso das tarefas estas são criadas e depois ao serem atribuídas aos membros passam a estar em execução e só estão concluídas quando termina a sua execução. O diagrama de estados para as tarefas pode ser visualizado na Figura 25.

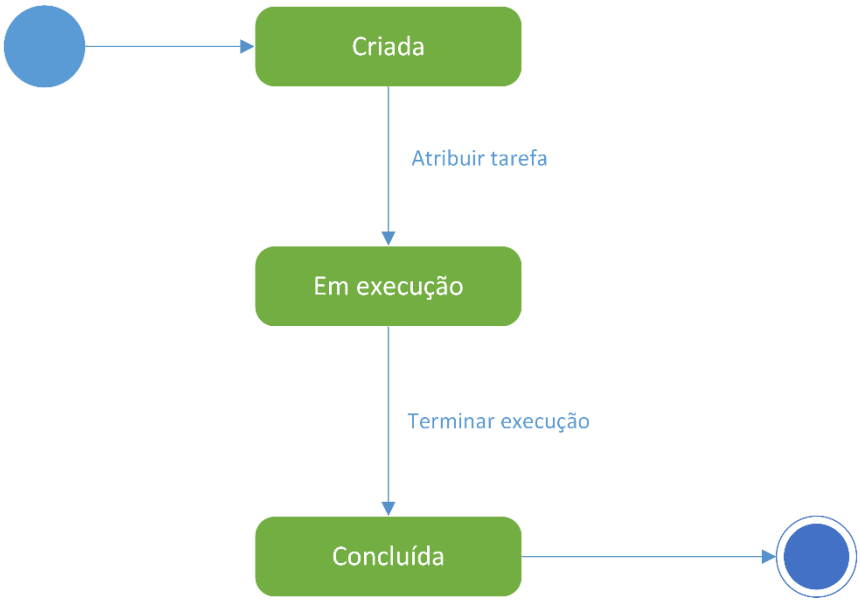


Figura 25: Diagrama de estados dos requisitos.

Referente às notificações, quando ocorre a receção de uma nova notificação o seu estado é "Por ler" e ao ler acaba o seu comportamento. Na figura 26 está representado o diagrama de estado das notificações.

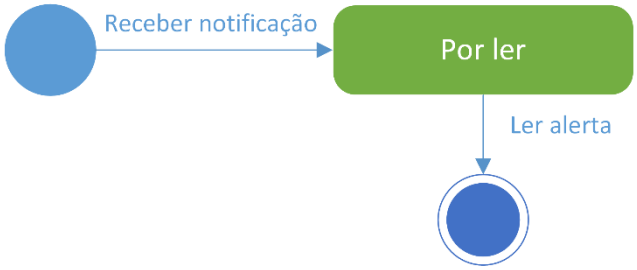


Figura 26: Diagrama de estados das notificações.

4.7 Casos de uso essenciais

Os casos de uso essenciais [40] são utilizados para obter as intenções que os utilizadores pretendem executar em relação a um sistema, estes não dependem nem da tecnologia nem da implementação. Os casos de uso essenciais mais pertinentes a este sistema estão apresentados da Tabela 15 à Tabela 20.

4.7.1 Efetuar a autenticação

Na Tabela 15 estão descritos os casos de uso essenciais para efetuar a autenticação.

Tabela 15: Casos de uso essenciais para a autenticação.

Intenção do utilizador	Resposta do sistema
	Solicitar credenciais.
Fornecer credenciais.	
	Validar. Verificar permissões de utilizador. Mostrar página inicial.

4.7.2 Adicionar um item

Visto que a adição de projetos, requisitos, equipas, utilizadores e clientes serem idênticos a nível dos casos essenciais, decidiu-se agrupá-los numa única entidade denominada de item. Na Tabela 16 está apresentado os casos de uso essenciais para a adição de um item.

Tabela 16: Casos de uso essenciais para a adição de um item.

Intenção do utilizador	Resposta do sistema
	Carregar dados necessários. Mostrar formulário.
Preencher formulário.	
	Validar.

	Verificar permissões de utilizador. Mostrar mensagem de sucesso. Mostrar página de detalhe do item adicionado.

4.7.3 Associar equipa a um projeto

Na Tabela 17 está descrito os casos de uso essenciais para associar uma equipa a um projeto.

Tabela 17: Casos de uso essenciais para associar uma equipa a um projeto.

Intenção do utilizador	Resposta do sistema
	Mostrar lista de projetos.
Escolher um projeto.	
	Mostrar detalhes do projeto.
Clicar em adicionar equipa.	
	Mostrar lista de equipas.
Escolher equipa.	
	Verificar permissões. Associar equipa. Adicionar permissões aos membros de equipa. Enviar notificações. Mostrar projeto com equipa associada.

4.7.4 Criar listas de tarefas e preencher detalhes da tarefa

Na Tabela 18 está presente os casos essenciais para a criação de listas de tarefas, que incluem a criação de tarefas e o preenchimento dos detalhes de cada tarefa.

Tabela 18: Casos de uso essenciais para criar listas de tarefas.

Intenção do utilizador	Resposta do sistema
	Mostrar projetos.
Selecionar projeto.	
	Verificar permissões. Mostrar quadro de lista de tarefas.
Clicar em adicionar lista.	
	Mostrar campo para inserir o nome da lista.
Inserir nome da lista.	
	Mostrar lista de tarefas introduzida.
Clicar em adicionar tarefa.	
	Mostrar campo para inserir a tarefa.
Inserir tarefa.	
	Mostrar tarefa introduzida.

4.7.5 Associar membros a uma tarefa

Na Tabela 19 é detalhado os cassos essenciais para a associação de membros a uma tarefa.

Tabela 19: Casos de uso essenciais para associar membros a uma tarefa.

Intenção do utilizador	Resposta do sistema
	Mostrar quadro de tarefas.
Clicar em tarefa.	
	Mostrar detalhes da tarefa.
Clicar em adicionar membros.	
	Mostrar lista de membros.
Escolher membros.	
	Associar membros a tarefa. Enviar notificações.

4.7.6 Modificar o estado de um projeto

Na Tabela 20 pode-se visualizar os casos essenciais para a tarefa de modificar o estado de um projeto.

Tabela 20: Casos de uso essenciais para modificar o estado de um projeto.

Intenção do utilizador	Resposta do sistema
	Mostrar lista de projetos.
Escolher projeto.	
	Verificar permissões. Mostrar detalhes do projeto.
Visualizar projeto. Clicar no estado pretendido.	
	Modificar estado do projeto.

4.7.7 Receber notificações

Na Tabela 22 está representado os casos de uso essenciais para a receção de notificações.

Tabela 21: Casos de uso essenciais para receber notificações.

Intenção do utilizador	Resposta do sistema
	Identificar subscritores Enviar mensagem
Receber notificação do tipo *pop up* Clicar em notificações	
	Apresentar lista de notificações
Visualizar notificação.	

4.8 Protótipos

Os protótipos de baixa fidelidade permitem obter uma primeira idealização sobre a forma como o projeto será estruturado a nível visual. Estes protótipos focam-se na estrutura e organização da interface que aplicação terá e apresentam um design com pouco detalhe. Além disto, é utilizado também para mostrar que tipo de ações e funcionalidades cada página terá. Este tipo de protótipo é útil para ser exposto ao cliente numa fase inicial de um projeto, que é caraterizada pela discussão dos aspetos estruturais do design, suscetível a ocorrência de alterações. Portanto, nesta fase, é vantajoso possuir um protótipo que seja fácil e rápido de modificar.

Para este sistema foram criados os protótipos de baixa fidelidade. Estes protótipos foram desenvolvidos tendo como base os requisitos do sistema e os tipos de utilizadores. Na Figura 27 está representado um exemplo do protótipo da página inicial, que é comum para os três tipos de utilizador. Os restantes protótipos do sistema estão disponíveis no Anexo D.

Figura 27: Protótipo da vista geral.

4.9 Arquitetura geral

Para a criação da arquitetura, este sistema foi denominado de Factory Odkas Management System (FOMS). Este divide-se numa implementação para o lado do cliente e outra para lado do servidor. O lado do cliente (cliente FOMS) é composto pelo pacote "FactoryOdkas" que contém a implementação do padrão MVC do sistema, englobando os pacotes OpenUI5, Socket.io e ChartJS. O lado do servidor (servidor FOMS) é constituído pela conexão entre a base de dados MySQL e a API REST. O servidor FMOS e o cliente FMOS comunicam entre si a partir de pedidos Ajax efetuados pelo cliente ao servidor, utilizando um protocolo HTTP. A utilização de uma API permite que qualquer outro sistema, desde que possua permissão, possa conectar-se e obter os dados pretendidos ou efetuar outro tipo de operação. Na Figura 28 está esquematizado a arquitetura do sistema FOMS.

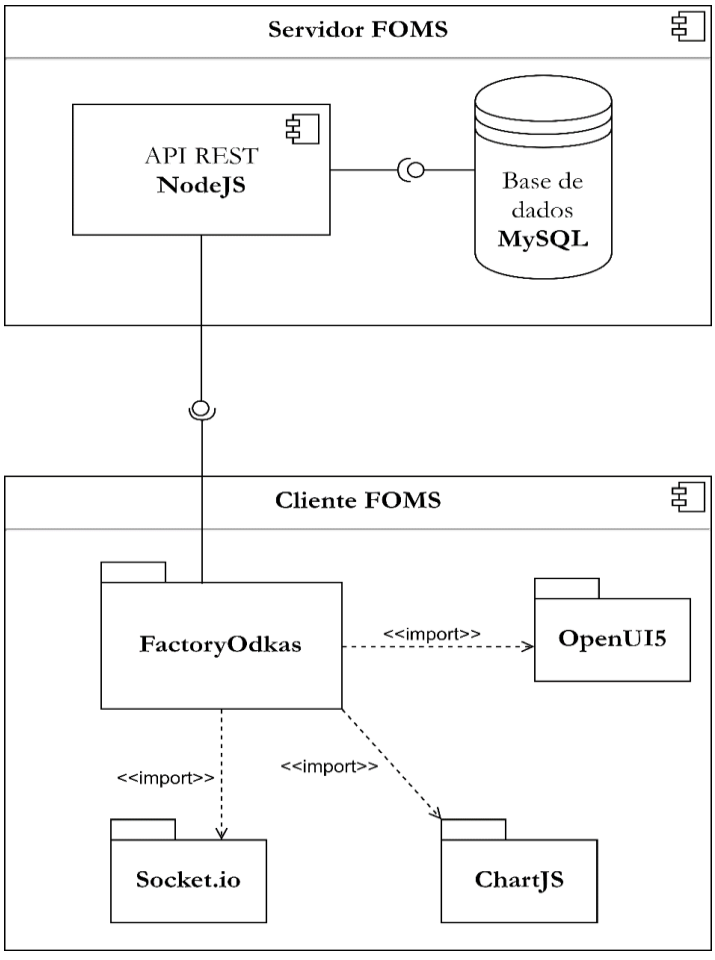


Figura 28: Arquitetura do sistema.

4.10 Arquitetura das notificações

Referente as notificações, o seu funcionamento foi baseado no padrão *publish-subscribe*. Este padrão consiste em produtores que emitem eventos, que são compostos por informação que será recebida pelos consumidores que estão subscritos a esses eventos. Neste sistema os componentes que irão desempenhar o papel de produtores serão os gestores de equipas, tarefas e projetos, o papel de consumidor estará a cargo do gestor de notificações. Entre os consumidores e produtores estará presente o componente *socket service,* que será responsável pela identificação dos consumidores, aquando o envio de mensagens, por subscrever e cancelar eventos. Os consumidores apenas recebem as notificações caso estejam conectados. Caso não estejam conectados o conteúdo da notificação é guardado na base de dados, para posteriormente ser apresentado. Esta arquitetura está apresentada na Figura 29 e foi baseada numa arquitetura do padrão *publish-subscribe* geral que está representada no Anexo E.

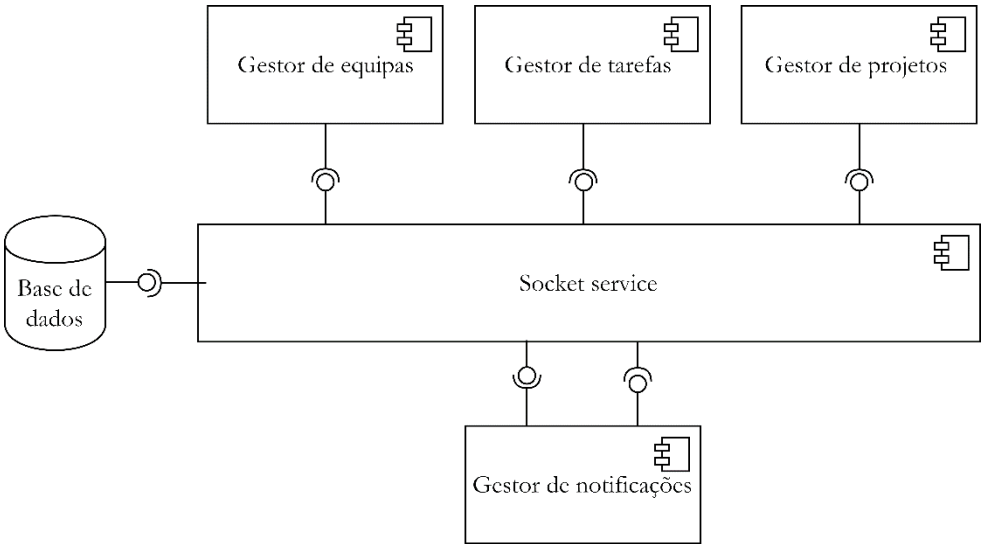


Figura 29: Arquitetura das notificações.

4.11 Conclusão

Neste capítulo foi apresentado o desenho de *software* elaborado para este sistema. Numa fase inicial definiu-se os requisitos funcionais, os cenários de desempenho, segurança, usabilidade e disponibilidade e os respetivos requisitos não funcionais, criando assim, a base de funcionalidades pretendidas. Após isto, foram identificados os utilizadores e os seus casos de utilização. Tendo os casos de utilização, fora necessário definir as entidades e descobrir como estas se iriam relacionar no contexto MVC, em relação as vistas, controladores e utilizadores a partir dos diagramas de robustez. Posteriormente, houve a necessidade de desenhar a forma como todas as entidades do sistema se iriam relacionar criando, por isso, os modelos de dados e o diagrama relacional. Constatou-se que algumas entidades iriam necessitar de estados e elaborou-se os diagramas de estado para os projetos, requisitos, tarefas e notificações. Com o objetivo de aprofundar o estudo das funcionalidades e ações do sistema, foram desenvolvidos os casos de uso essenciais. Tendo as funcionalidades, as entidades e utilizadores definidos, procedeu-se à criação de protótipos de baixa fidelidade. Por fim, passou-se ao desenho das arquiteturas do sistema, afim, de criar uma estrutura que interligasse todos os componentes necessários para que o sistema pudesse funcionar.

5 DESENVOLVIMENTO

Esta secção contém a explicação do processo de transição do desenho de *software* para a implementação. Esta fase foi dividida na implementação do cliente e servidor FOMS.

5.1 Criação do cliente FOMS

Como já foi referido anteriormente, a biblioteca escolhida para desenvolver o cliente FOMS foi o OpenUI5. Em [41] está disponível quatro opções de iniciação de uma aplicação em OpenUI5/SAPUI5. Como as opções relativas ao SAPUI5 é necessário possuir uma licença paga, restava as opções utilizando o Node.js ou utilizar o ambiente de desenvolvimento do Openui5. Como se notou que a opção referente a utilização do Node.js apresentava uma estrutura de ficheiros e pastas um pouco mal estruturada e confusa, optou-se por utilizar o ambiente de desenvolvimento do OpenUI5. Para esta opção seguiu-se os passos descritos em [42] e obteve-se a estrutura de ficheiros representada na Figura 30.

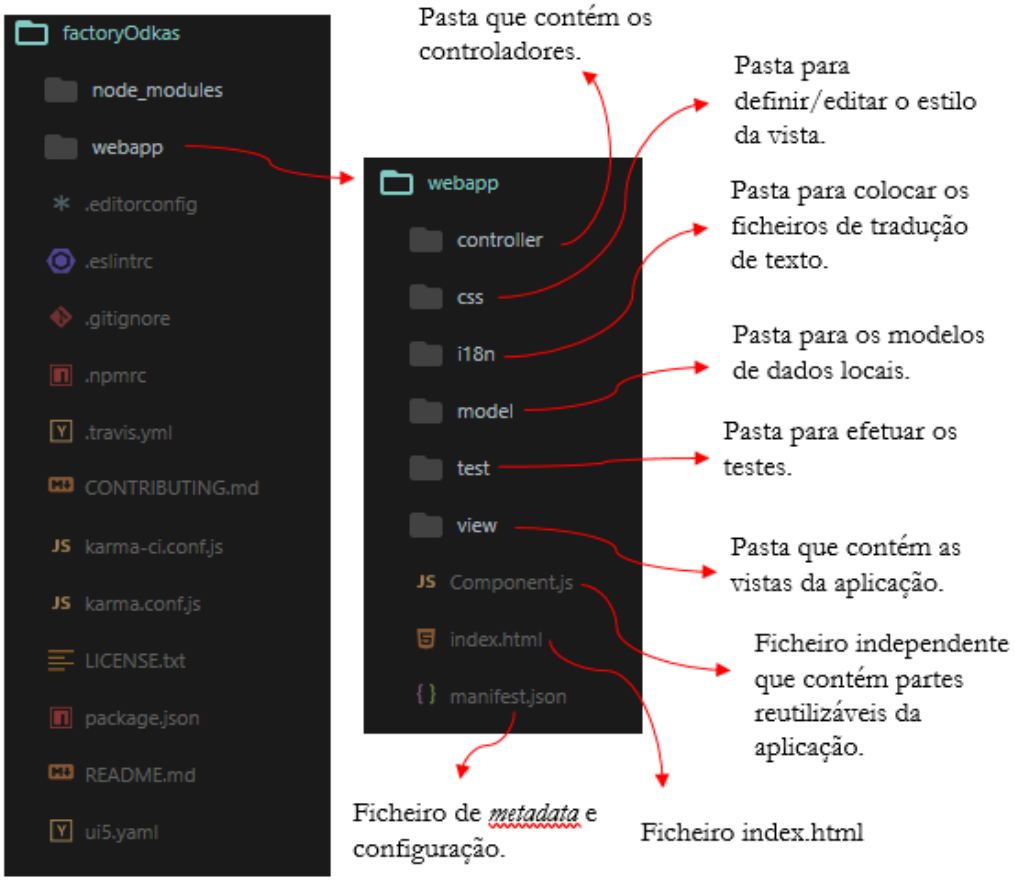


Figura 30: Estrutura de ficheiros para o desenvolvimento do *front end.*

Após a criação da estrutura era necessário obter um conhecimento, mais técnico sobre o funcionamento do OpenUI5. Para isto, completou-se o estudo efetuado na secção 2.2.1 sobre as vistas, controladores e modelos. Referente às vistas, o nome destas é especificado pelo nome do ficheiro e pela estrutura de pastas. Por exemplo, para a estrutura da pasta `resources/sap/hcm/Addess.view.xml`, o nome da vista será `sap.hcm.Addess`. Este nome poderá ser utilizado para apresentar uma instância desta vista. Uma vista em XML apresenta uma estrutura como abaixo representada:

```
<mvc:View
        controllerName="sap.hcm.Address"
        xmlns="sap.m"
        xmlns:mvc="sap.ui.core.mvc">
   <Panel>
        <Image src="images/sap.png "/>
        <Button text="Press Me!"/>
   </Panel>
</mvc:View>
```

Passando em seguida aos controladores, deparou-se com uma funcionalidade que se revelou útil no desenvolvimento deste sistema, que é a possibilidade de utilizar um controlador genérico. Este permite adicionar métodos e propriedades para serem utilizados como uma classe base para outros controladores [43]. Para definir um controlador, basta utilizar o seguinte construtor:

```
new sap.ui.core.mvc.controller("controllerName", {
        //lógica do controlador
})
```

O OpenUI5 contém métodos de ciclo de vida predefinidos, que permitem a adição de manipuladores de eventos ou outras funções de emissão de eventos. Estes métodos podem ser:

- `onInit()`: é chamado quando a vista é instanciada e tem como finalidade a modificação da vista antes de ser criada;
- `onExit()`: tem como finalidade a libertação de recursos e finalizar atividades quando a vista é destruída;

- `onAfterRendering()`: é chamado quanto a vista já efetuou o processo de carregamento. Podendo ser útil para efetuar manipulações após a vista ser apresentada;
- `onBeforeRendering()`: é sempre chamado antes do carregamento da vista e antes de o HTML ser colocado na estrutura de árvore DOM.

Em relação ao tipo de modelo de dados a utilizar para este sistema, foi escolhido o modelo JSON. Uma vez que este modelo permite o carregamento de dados do servidor e depois a manipulação destes no lado do cliente. No caso do envio de dados, do cliente para o servidor, como era pretendido enviar os dados com auxílio de pedidos AJAX, em formato JSON, este modelo era o mais indicado. Para utilizar este modelo é necessário criar uma instância, da seguinte forma:

```
var oModel = new sap.ui.model.json.JSONModel();
```

Após a criação da instância, é necessário definir os dados do modelo. Para isto existem as seguintes possibilidades:

- Utilizar o método **`setData`**:

```
oModel.setData({
      firstName: "Peter",
      lastName: "Pan"
});
```

- Utilizar o método **`loadData`**, que carrega os dados a partir de um URL**:**

```
oModel.loadData("data.json");
```

Após a criação da estrutura de ficheiros e o estudo técnico do MVC do OpenUI5 passou-se a implementação. Esta fase foi dividida pela implementação dos diferentes gestores deste projeto. A relação entre as vistas e os controladores adotada foi de 1:1, portanto, todas as vistas possuem um controlador com o mesmo nome, embora exista controladores genéricos com funcionalidades comuns, a fim de reutilizar código. Em relação ao design das vistas, o OpenUI5 possui uma grande variedade de amostras [44], em que se baseou para a criação das vistas. Os *screenshots* das vistas a seguir mencionadas podem ser visualizadas no Anexo F.

5.1.1 Gestor de projetos

O gestor de projetos engloba as vistas/controladores e fragmentos apresentados na Tabela 22 e Tabela 23 respetivamente.

Tabela 22: Vistas do gestor de projetos.

Vista/Controlador	Descrição
Projects	Apresenta uma lista com os projetos.
AddProject	Contém o formulário para adicionar um novo projeto.
ProjectDetails	Apresenta os detalhes de um projeto, os seus requisitos e qual a equipa a que está associado.
RecycleBin	Apresenta os projetos com o estado de eliminado, com a possibilidade de os eliminar permanentemente.
Clients	Visualizar e adicionar clientes.

Os fragmentos descritos na Tabela 23 pertencem a vista "ProjectDetails" e ocorre uma mudança do fragmento "DisplayProjectDetails" para o fragmento "ChangeProjectDetails", correspondendo à situação em que o líder de equipa decide alterar os detalhes do projeto. O contrário acontece quando o líder de equipa guarda as alterações ou cancela-as. O fragmento "RequirementsTable" é agrupado pelo fragmento "ChangeProjectDetails" e é partilhado com o gestor de requisitos. Enquanto que no fragmento "ChangeProjectDetails" o "RequirementsTable" tem como a principal função a edição dos requisitos, no gestor de requisitos é utilizado para adicioná-los. Este fragmento pode ser visualizado na Figura 33.

Tabela 23: Fragmentos da vista ProjectDetails.

Fragmento	Descrição
DisplayProjectDetails	Apresenta todos os detalhes do projeto
ChangeProjectDetails	Apresenta o formulário de edição do projeto e dos requisitos.
RequirementsTable	Tabela para adicionar/editar os requisitos.

A vista "Projects" além da lista de projetos apresenta, também, uma estrutura de filtros baseada em ícones de estado e um campo de procura pelo nome do projeto. Os ícones são utilizados para filtrar os projetos pelo estado e em cada ícone é indicado a soma do número de projetos que existe em cada estado. Esta estrutura está representada na Figura 31.

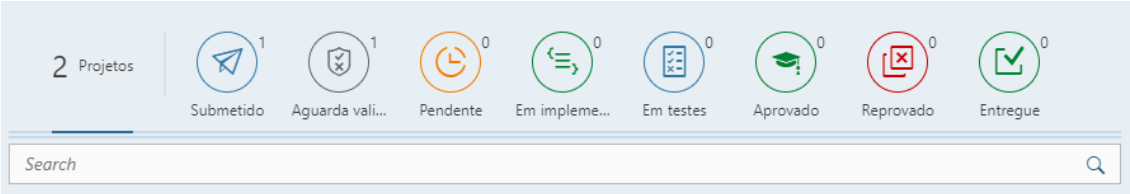


Figura 31: Estrutura de filtros para os projetos.

A vista "ProjectDetails" contém os botões do rodapé da página que apenas estão disponíveis para o administrador e líder de equipa. Estes botões estão representados na Figura 32, no lado direito, e são compostos pelas opções de editar, eliminar, cancelar a edição e guardar a edição do projeto.

Na Figura 32 é possível visualizar, também, os botões no lado esquerdo que se referem a funcionalidade para mudar o estado do projeto. Estes botões possuem uma visibilidade dinâmica, ou seja, os botões visíveis dependem do estado atual do projeto, representando quais os estados seguintes possíveis. Para voltar ao estado anterior do projeto foi implementado um botão "undo".

Figura 32: Rodapés da vista "ProjectDetais".

Para a funcionalidade de eliminação de um projeto recorreu-se ao conceito de reciclagem, em que um projeto toma o estado de eliminado e é enviado para o modelo da vista "RecycleBin", que depois poderá ser eliminado permanentemente ou ser restaurado. Esta funcionalidade apenas está disponível ao líder e ao administrador.

5.1.2 Gestor de requisitos

O gestor de requisitos neste sistema pode ser considerado como um componente que pertence ao gestor de projetos, pois, é a partir deste que se chega ao gestor de requisitos. Este é composto pelas vistas/controladores da Tabela 24.

Tabela 24: Vistas/controladores da gestão de requisitos.

Vista/Controlador	Descrição
ViewRequirement	Apresenta os detalhes de um requisito e tarefas associadas a este.
AddRequirements	Contém o formulário para adicionar um novo requisito.

A vista "AddRequirements" possui o fragmento "RequirementsTable" utilizado também pelo gestor de projetos. Um exemplo deste fragmento pode ser visualizado na Figura 33.

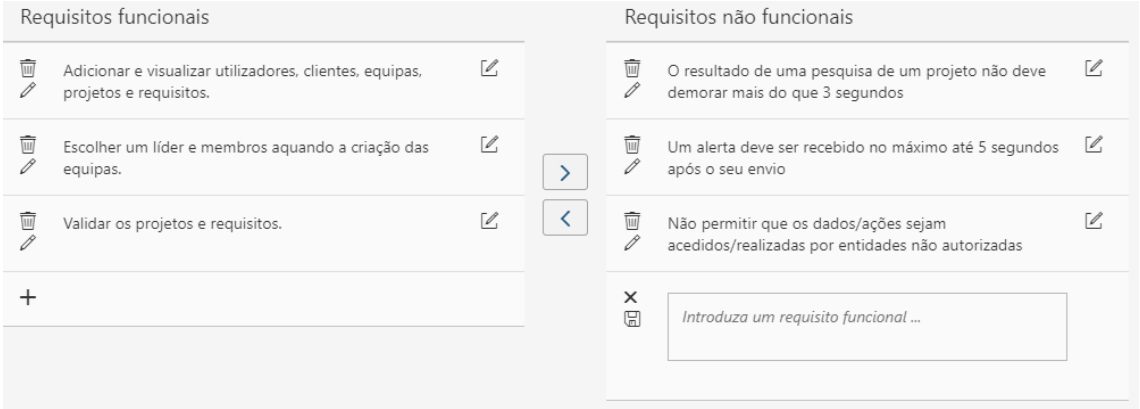


Figura 33: Fragmento para adicionar/editar requisitos.

Este fragmento é composto por duas colunas, uma coluna para os requisitos funcionais e outra para os requisitos não funcionais. No fim de cada coluna existe uma linha com o símbolo + para adicionar um novo requisito. Ao clicar neste símbolo aparece um campo para adicionar o requisito com os ícones de guardar 🖫 ou cancelar ✕ (Representado no final da coluna da direita na Figura 33). Após adicionado o requisito é apresentado as opções de editar ✎, eliminar 🗑 e adicionar uma justificação 🗹 (campo opcional). Entre as tabelas existem dois botões direcionais com a função de mudar o requisito de tabela.

Teve-se ainda o cuidado de bloquear a alteração de coluna da linha correspondente a adição de uma novo requisito. Toda a parte responsável pelo controlo deste fragmento está presente no ficheiro "MyUtils.js", que é depois instanciado pelo gestor de projetos e requisitos.

Referente a mudança de estado dos requisitos é possível validar ou revogar todos a partir da vista "ProjectDetails" (vista que pertence ao gestor de projetos. Ver Figura 34) ou fazer a validação individual de cada requisito na vista "ViewRequirement" representado na Figura 35. Esta solução deve-se a facto de os projetos dependerem do estado de validação dos requisitos. Os restantes estados dos requisitos podem ser alterados apenas na vista "ViewRequirement" (Figura 35). Caso

um projeto ainda não tenha requisitos é apresentado um botão para adicionar requisitos ao projeto. Isto apenas para o líder, para o membro apenas aparece a informação de que ainda não foram adicionados requisitos.

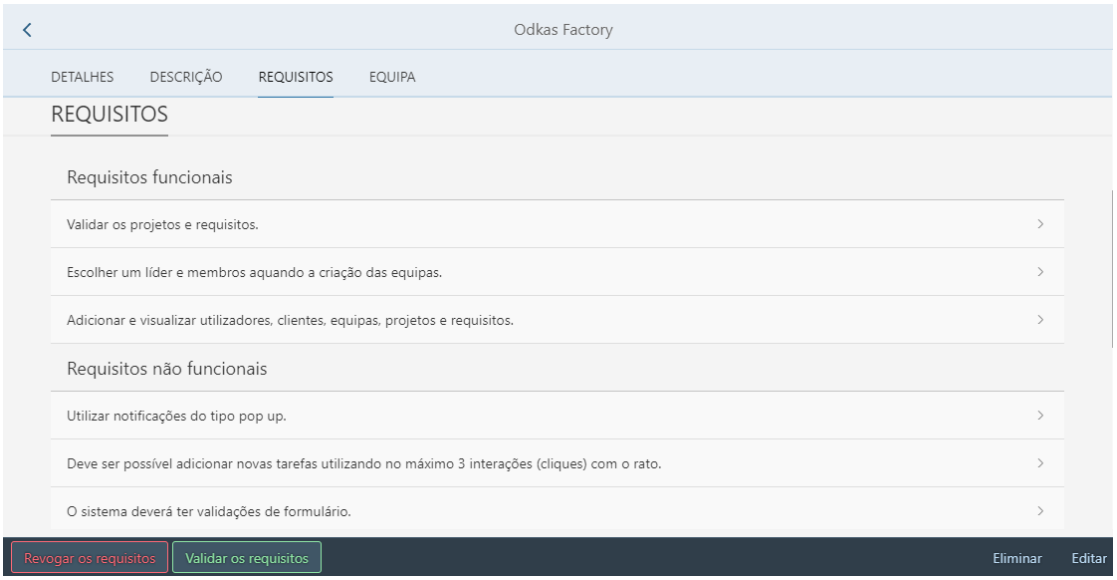


Figura 34: Vista "ProjectDetails" com as opções de validação de todos os requisitos.

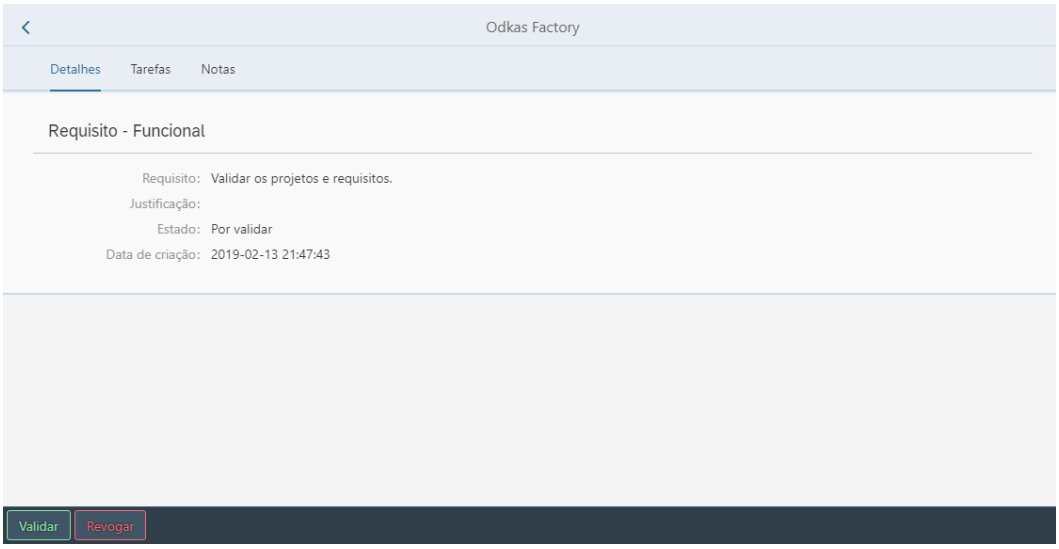


Figura 35: Vista "ViewRequirement" com as opções de rodapés para validar um requisito.

Para esta gestão foi, ainda, adicionado uma funcionalidade extra, que permite a adição de notas/comentários a cada um dos requisitos. Estas notas apenas podem ser adicionadas pelos membros da equipa a que o projeto está associado e poderá ser útil para criar uma discução ou adicionar informação relevante a um requisito. Esta função está representada na Figura 84.

Com a implementação do gestor de projetos e do gestor de requisitos dá-se como verificado os requisitos RF03, RF11, RF12, parte do requisito RF01 e parte do requisito RNF09.

5.1.2.1 Mapa de navegação do gestor de projetos e requisitos

Na Figura 36 está representado o mapa de navegação entre as vistas e os fragmentos que compõem a gestão de projetos e requisitos. A linha contínua representa a mudança de vista e a linha a tracejado representa a fragmentação de uma vista. O mapa de navegação completo do sistema está disponível no Anexo C.

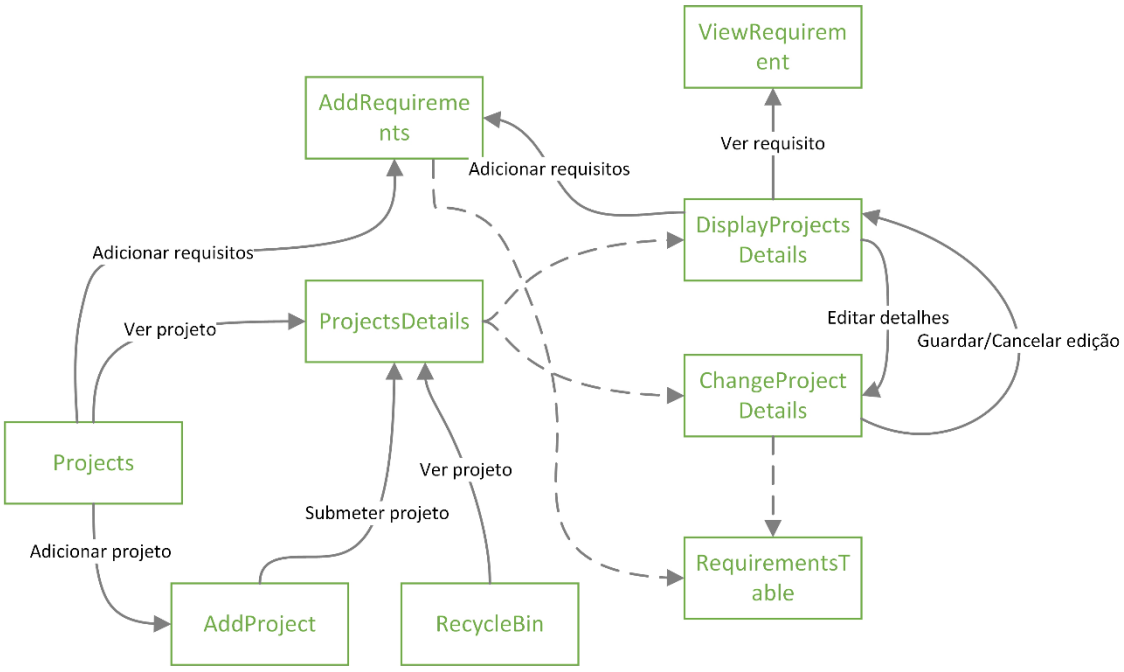


Figura 36: Mapa de navegação dos gestores de projetos e requisitos.

5.1.3 Gestor de tarefas

Pode-se considerar o gestor de tarefas como o gestor central deste sistema, pois, é onde o utilizador irá gerir o tempo de execução de um projeto como, também, a atribuição de tarefas. As vistas/controladores deste gestor estão descritas na Tabela 25.

Tabela 25: Vistas/controladores da gestão de tarefas.

Vista/Controlador	Descrição
Tasks	Apresenta os projetos numa estrutura de quadros para serem selecionados.
ViewTasks	Contém o quadro Kanban e calendário de planeamento relacionado com um projeto.

Cada líder de equipa e membro de equipa terá na vista "Tasks" (Figura 37) a possibilidade de escolher um dos seus projetos para poder visualizar o quadro de tarefas, baseado no método Kanban, e o calendário de planeamento na vista "ViewTasks".

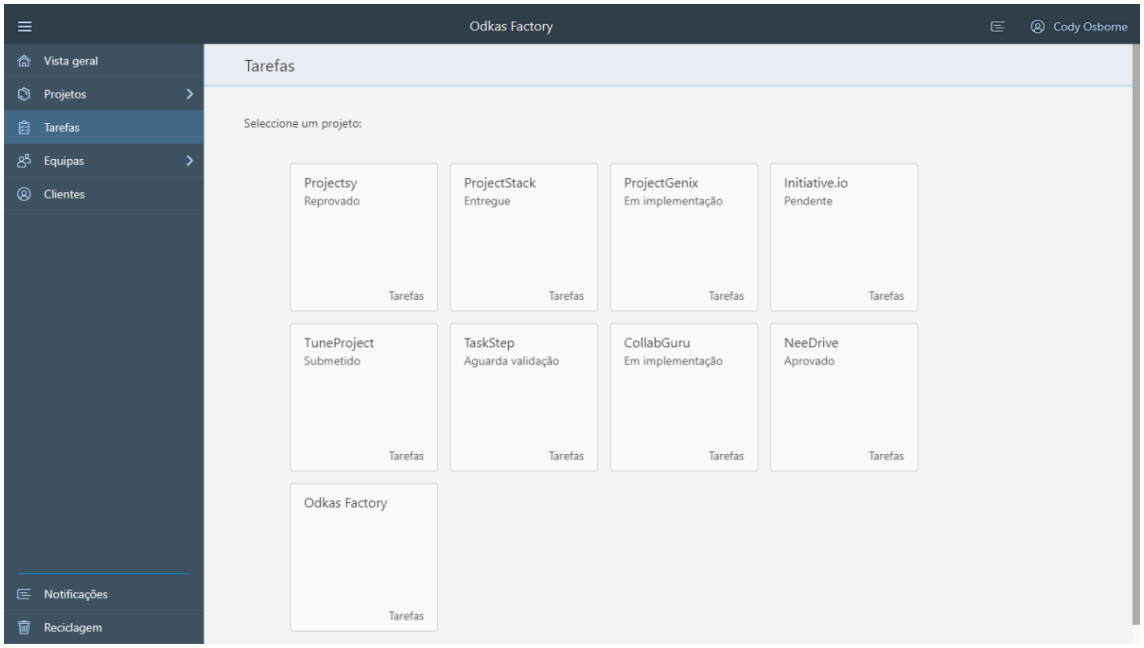


Figura 37: Vista Tasks.

Como a vista "ViewTasks" é composta por dois separadores, um para o quadro de tarefas e outro para o calendário de planeamento, por uma questão de organização de código criou-se um fragmento para cada um deles. Estes fragmentos estão descritos na Tabela 26.

Tabela 26: Fragmentos da vista "ViewTasks".

Vista/Controlador	Descrição
AddTasks	Quadro Kanban.
TasksCalendar	Calendário de planeamento de tarefas.

O desenvolvimento do fragmento "AddTasks" foi desenvolvido a partir de uma criação em XML dos componentes estáticos deste fragmento e no controlador da vista "ViewTasks" está a criação dos elementos dinâmicos. Seguiu-se esta abordagem pelo facto de ser mais fácil manipular objetos dinâmicos em JS do que em XML.

Os elementos estáticos definem-se pelo componente responsável pela navegação horizontal das listas de tarefas (ScrollContainer), pelos componentes para inserir os elementos dinâmicos (HBox com o id="taskHBox") e o botão para adicionar uma nova lista. Na Figura 38 pode-se visualizar o código XML do fragmento "AddTasks" onde estão representados os elementos estáticos.

```
<core:FragmentDefinition
    xmlns:core="sap.ui.core"
    xmlns:u="sap.uxap"
    xmlns:f="sap.ui.layout.form"
    xmlns="sap.m"
    height="100%">

    <ScrollContainer
        style="{padding: 50rem}"
        horizontal="true"
        vertical="false"
        focusable="false">

        <HBox renderType="Bare" >
            <HBox id="tasksHBox" class="hBoxTaskTable">

            </HBox>
            <Button
                id="addTaskButton"
                class="addTaskListButton"
                text="Adicionar nova lista"
                icon="sap-icon://add-activity-2"
                press="addNewListPress"/>
        </HBox>
    </ScrollContainer>
</core:FragmentDefinition>
```

Figura 38: Código XML do fragmento "AddTasks".

Os elementos dinâmicos encorporam as seguintes entidades do modelo relacional, presente no Anexo B:

- **"taskList":** para representar as listas de terefas do projeto selecionado;
- **"task":** para apresentar as tarefas para cada lista de tarefas;
- **"assigment" e "user":** para identificar quais os membros atribuidis a cada tarefa;

- **“task_has_requirements” e “requirements”:** para mostrar quais os requisitos que pertencem a cada tarefa;
- **“team”, “group”, “membership” e “user”:** para apresentar a lista de membros da equipa que podem ser atribuidos as tarefas.

Após a identificação das entidades que envolvem a gestão de tarefas, procedeu-se à criação da estrutura para o modelo de dados JSON, que irá constituir cada lista de tarefas. Esta estrutura está representada na Figura 39. Nesta estrutura foram adicionados variáveis de controlo adicionais, tal como a “listVisible”, que torna a lista invisível na inserção do nome da lista e quando é eliminada e torna-a visivel no momento da submissão do nome da lista (ver Figura 40). A variável “editList” mostra ou esconde o campo de edição do nome da lista (ver Figura 41). A variável “editTask” mostra ou esconde o campo de edição do nome da tarefa, ao clicar com o botão direito do rato no nome da tarefa (ver Figura 42).

```
{
    "listId": "id da lista",
    "listName": "Nome da lista",
    "projectId": "id do projeto",
    "listVisible": "valor booleano que define a visibilidade da lista de tarefas",
    "editList": "valor booleano que endica se a lista está a ser editada ou não",
    "tasks":[
        {
            "taskId": "id da tarefa",
            "name": "nome da tarefa",
            "description": "descrição da tarefa",
            "state": "estado da tarefa",
            "deliveryDate": "data de entrega da tarefa",
            "beginningDate": "data de inicio da execução da tarefa",
            "taskListId": "id da lista de tarefas a que esta tarefa pertence",
            "editTask": "valor booleano que indica se o nome da tarefa está a ser editada",
            "newTask": "valor booleano que indica se é uma nova tarefa",
            "members": [
                {
                    "memberId": "id do membro",
                    "name": "nome do membro"
                }
            ],
            "requirements": [
                {
                    "requirementId": "id do requisito",
                    "requirement": "requisito",
                    "type": "tipo do requisito"
                }
            ],
        }
    ]
}
```

Figura 39: Estrutura em JSON de uma lista de tarefas.

Figura 40: Tornar a lista de tarefas visível ao premir a tecla "Enter".

Figura 41: Tornar o campo de edição do nome da lista visível.

Na Figura 42 é possível, ainda, visualizar a caixa de diálogo que aparece ao clicar numa tarefa presente na lista de tarefas. Esta apresenta os detalhes da tarefa, como o nome, descrição, estado e data de início e a de entrega, os membros e os requisitos associados. Para o intervalo de datas teve-se em atenção limitar a opção entre a data de submissão do projeto até à data de entrega do projeto. No rodapé está disponível a opção de eliminar a tarefa e de fechar a caixa de diálogo.

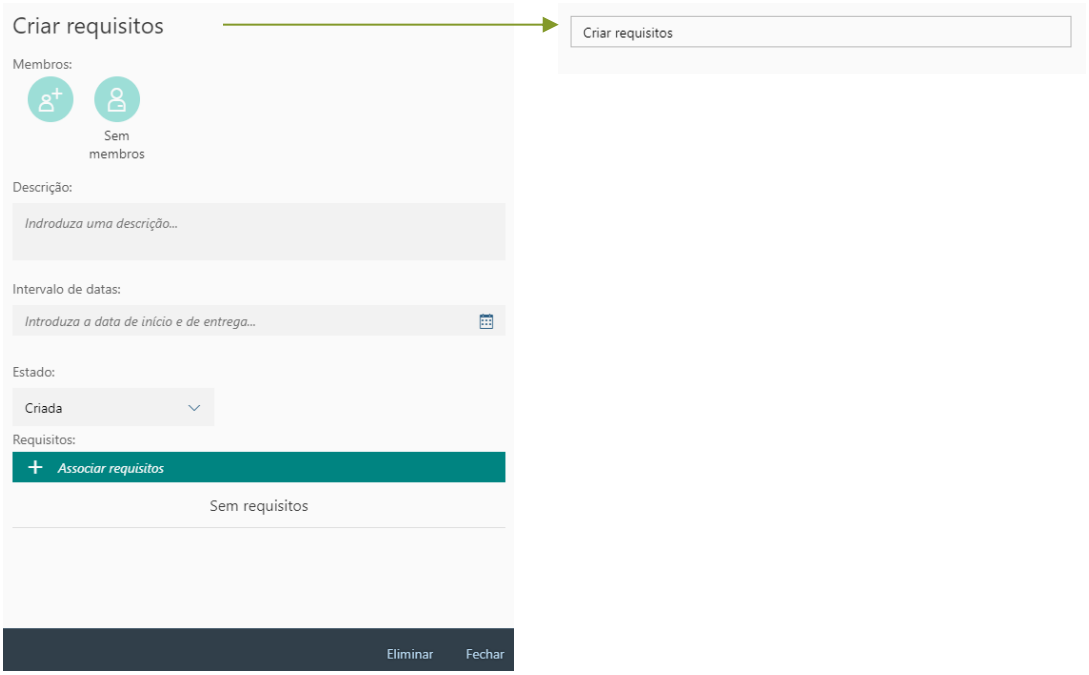


Figura 42: Tornar visível o campo de edição do nome da tarefa.

O quadro de tarefas apresentado na Figura 43 foi baseado nos quadros Kanban utilizado nas ferramentas Trello e Planner apresentadas na secção 3.

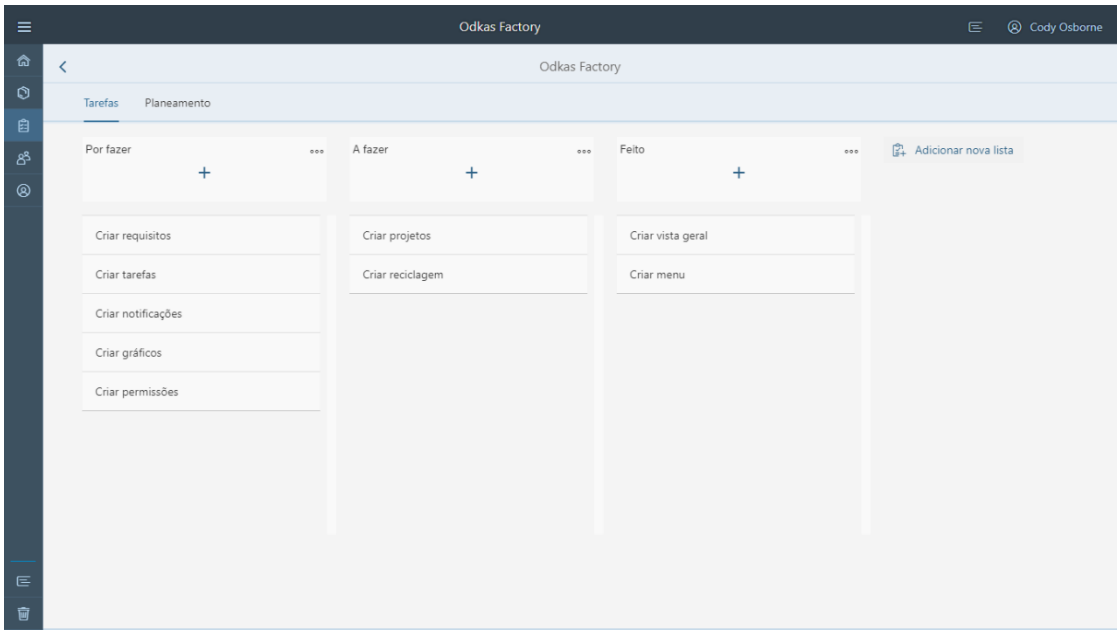


Figura 43: Quadro de tarefas da ferramenta Odkas Factory.

O fragmento "TasksCalendar" pode ser visualizado na Figura 44. Este é constituído por colunas temporais e linhas de tarefas associadas a cada membro de equipa. A nível do intervalo temporal é possível mudar a escala para horas, dias, meses, uma semana e para um mês. O tempo atual é representado por uma linha na vertical (representada a roxo na Figura 44). A partir deste calendário o utilizador consegue organizar o seu tempo de execução de tarefas, verificando os dias em que terá sobrecarga de trabalho e em que altura terá mais tempo livre. O *template* deste calendário foi retirado dos calendários do OpenUI5, disponíveis como amostra em [45]. Inicialmente tinha-se planeado implementar um gráfico de Gantt, mas como este componente só está disponível para o SAPUI5, utilizou-se este calendário como alternativa.

Após a implementação do gestor de tarefas verifica-se os requisitos RF04, RF05, RF07, RF08, RF09, RF10 e parte do RNF08. Referente ao RNF07 o número máximo de interações com o rato é 2 , um clique para criar a lista de tarefas e outro para criar a tarefa. No caso de uma lista de tarefas criada apenas é necessário uma interação com o rato para adicionar uma nova tarefa a lista, portanto este requisito é, também, verificado.

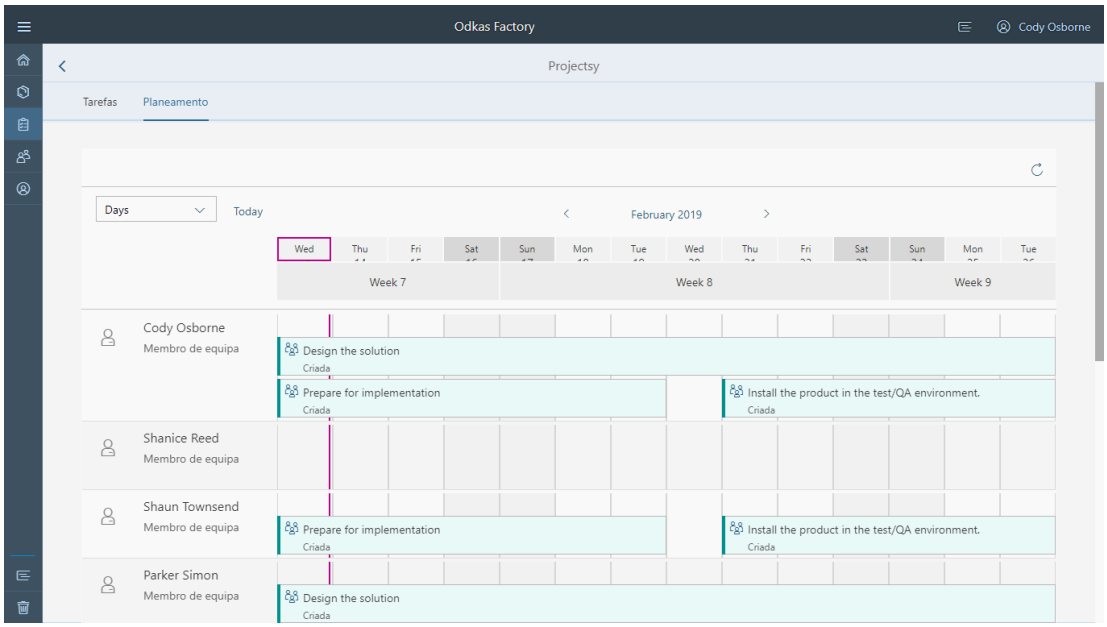


Figura 44: Fragmento "TasksCalendar" da vista "ViewTasks".

5.1.3.1 Mapa de navegação da gestão de tarefas

O mapa de navegação referente a gestão de tarefas está representada na Figura 45.

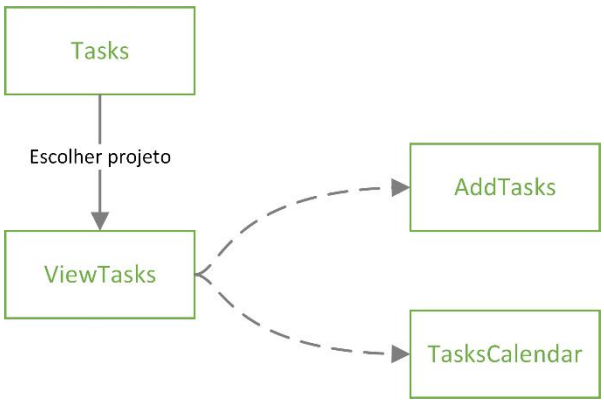


Figura 45: Mapa de navegação do gestor de tarefas.

5.1.4 Gestor de equipas

O gestor de equipas agrupa as vistas/controladores descritos na Tabela 27.

Tabela 27: Vistas/controladores do gestor de equipas.

Vista/controlador	Descrição
AddTeam	Responsável por adicionar uma equipa.
Teams	Apresenta as equipas adicionadas.
ViewTeam	Visualizar os detalhes da equipa e opções de edição.

A criação de uma equipa é efetuada pela introdução do seu nome e a seleção de um líder e dos membros de equipa. O nome da equipa é um parâmetro único e para selecionar o líder de equipa o utilizador dispõe de um botão que abre um diálogo composto por uma lista de seleção. Caso haja a necessidade de mudar a escolha, basta voltar a clicar no mesmo botão e escolher outro líder. O mesmo comportamento acontece para a seleção dos membros, mas com a opção de uma lista de multi-seleção, que está representada no diálogo do lado direito da Figura 46. Depois de os membros estarem selecionados existe a possibilidade de os remover clicando no ✕ representado na Figura 47.

Como um líder de equipa pode ser também um membro de equipa, estes aparecem tanto na lista de líderes como na lista de membros. Caso um líder seja selecionado como líder e como membro da mesma equipa, a seleção deste como papel de membro será ignorada e ficará como líder. Na operação de seleção de membros teve-se em atenção a remoção das opções que foram anteriormente selecionadas. Por exemplo após selecionar os membros Parker Simon, Shaun Townsend e Shanice Reed se o utilizador decidir em voltar a lista de membros estes membros já não irão estar disponíveis, uma vez que já foram adicionados.

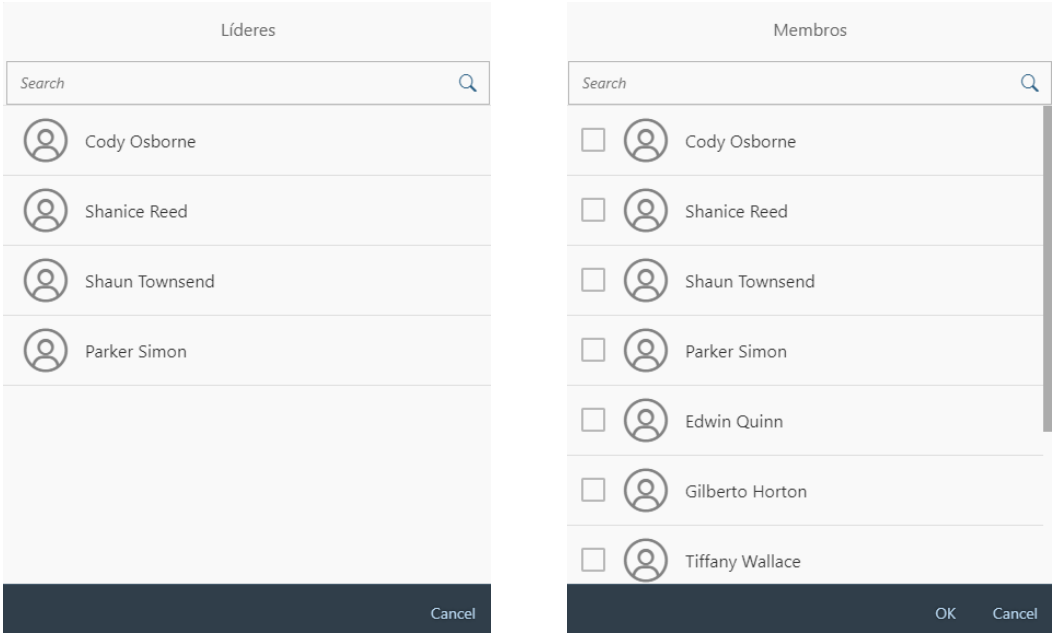


Figura 46: Diálogos para selecionar o líder de equipa e membros de equipa.

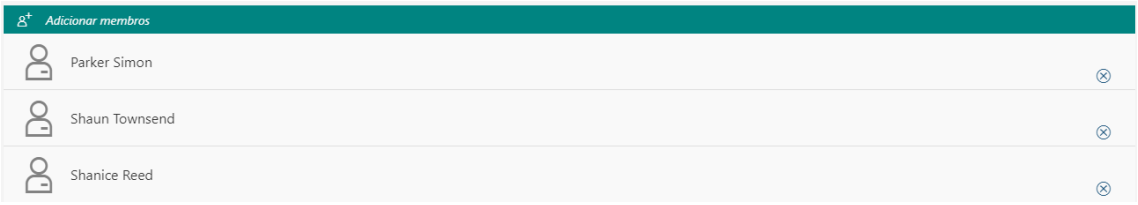


Figura 47: Lista dos membros selecionados.

> O gestor de equipas verifica os requisitos RF02, parte do RF01 e parte do RNF08.

5.1.4.1 Mapa de navegação para a gestão de equipas

O mapa de navegação para o gestor de equipas está representado na Figura 48. Nesta navegação é possível navegar até à gestão de projetos (vista "ProjectDetails") quando se pretende visualizar um dos projetos que pertencem a uma equipa e o contrário acontece quando se pretende adicionar uma nova equipa a um projeto.

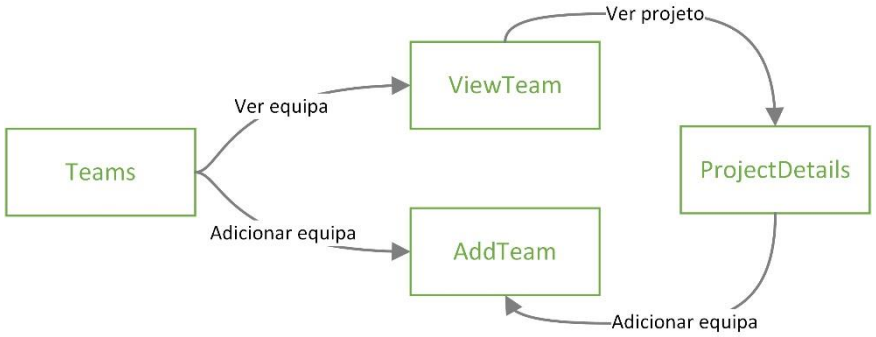


Figura 48: Mapa de navegação para a gestão de equipas.

5.1.5 Gestor de utilizadores

O gestor de utilizadores apenas está acessível ao administrador e é responsável por adicionar e remover utilizadores do sistema. Além disto, mostra para cada utilizador uma visualização dos seus detalhes. Na Tabela 28 está representado as vistas/controladores deste gestor.

Tabela 28: Tabela das vistas/controladores da gestão de utilizadores.

Vista/controlador	Descrição
AddUser	Responsável por adicionar novos utilizadores.
Users	Lista de utilizadores.
UserProfile	Visualizar um utilizador.

A vista "Users" e a vista "AddUser" podem ser visualizadas na Figura 92 e na Figura 93, respetivamente, no Anexo F. Para as funcionalidades de edição do tipo de utilizador e a eliminar um utilizador do sistema foi utilizado um conjunto de ícones localizados em cada linha da lista de utilizadores. O ícone representa a edição do tipo de utilizador, o ícone simboliza a eliminação do utilizador, composta por um diálogo de confirmação do mesmo, o ícone é responsável por guardar as alterações efetuadas e o ícone é utilizado para cancelar a edição.

5.1.5.1 Mapa de navegação do gestor de utilizadores

Na figura 49 está representado o mapa de navegação da gestão de utilizadores.

Figura 49: Mapa de navegação para a gestão de utilizadores.

O gestor de utilizadores conclui a implementação do requisito RF01 e verifica o requisito RNF03.

5.1.6 Gestor de autenticação

O gestor de autenticação é composto apenas pela vista/controlador "Login". Este é responsável pela autenticação do utilizador, que envolve a validação das credenciais do utilizador ao entrar na aplicação e guardar a sessão, até que o utilizador

decida terminá-la ou não utilize a sua conta iniciada durante um dia. Este controlo é efetuado a partir de uma *cookie* de sessão, com o nome "connect.sid".

O utilizador que não tiver sessão iniciada e tentar aceder a alguma página, além da página para iniciar sessão, é lhe apresentado a mensagem representada na Figura 94 no Anexo F. Isto acontece devido a um pedido de verificação da autenticação feito ao servidor sempre que uma vista é inicializada. Este pedido é efetuado no ficheiro Component.js, que é comum a todas as vistas e, sempre que existe um carregamento de uma vista, no OpenUI5, este ficheiro é o primeiro a ser executado. A restante explicação deste gestor no lado do servidor continua no ponto 5.2.2.

> O gestor de autenticação verifica os requisitos RNF03 e finaliza o RNF08.

5.1.7 Gestor de permissões

O gestor de permissões, no lado do cliente, é responsável por mostrar ao utilizador apenas o que lhe é permitido e está definido no ficheiro "PermissionMangager.js".

A solução para este problema inicia-se pela definição de uma política de mostrar apenas, em cada vista, o que é permitido aceder para cada tipo de utilizador. Para definir as permissões que cada tipo de utilizador terá no sistema elaborou-se a Tabela 29, com a seguinte legenda:

- ● : Permissão total. Por exemplo, um líder de equipa pode ler todos os clientes;
- ◖ : Permissão parcial. Por exemplo, os membros de equipa e líderes de equipa só têm acesso a leitura dos projetos das equipas a que pertencem;
- ▬ : Sem permissão. Por exemplo, os membros de equipa não têm permissão para atualizar projetos.

Tabela 29: Tabela de permissões.

Ação	Administrador	Líder de equipa	Membro de equipa
CRUD utilizadores	●	—	—
CRUD clientes	●	●	—
Criar projetos/requisitos	●	●	—
Ler projetos/requisitos	●	◖	◖
Atualizar projetos/requisitos	●	◖	—
Eliminar projetos/requisitos	●	◖	—
Criar equipas	●	●	—
Ler equipas	●	◖	◖
Atualizar equipas	●	◖	—
Eliminar equipas	●	◖	—
Associar projetos a equipas	●	◖	—
CRUD tarefas	●	◖	◖
Associar membros/requisitos a tarefas	●	◖	◖
CRUD notificações	◖	◖	◖
Acesso a vista geral	●	●	●

Após elaborar a tabela de permissões, definiu-se uma atribuição dinâmica de rotas permitidas a cada tipo de utilizador, ou seja, sempre que um utilizador inicie sessão ou carregue uma vista, é verificado qual o tipo de utilizador e consoante o seu tipo são carregadas as rotas permitidas a este. Para isto, utilizou-se um modelo de rotas local para cada tipo de utilizador, que estão descritos na Tabela 30.

Tabela 30: Tabela dos modelos locais para as rotas.

Modelo	Descrição
adminRoutes	Rotas permitidas para o administrador.
leaderRoutes	Rotas permitidas para o líder de equipa.
memberRoutes	Rotas permitidas para o membro de equipa.

Na Figura 50 está representado a função que define as rotas. Esta recebe as rotas que foram definidas anteriormente e depois adiciona-as ao Router. O Router é o componente responsável pela gestão de rotas no OpenUI5.

```
createRoutes: function( routesData, self ) {
    var oRouter = self.getRouter();
    oRouter.addRoute( routesData );
},
```

Figura 50: Código para definir rotas.

> O gestor de permissões verifica os requisitos RNF03 e RNF04.

Para criar a barra de navegação lateral, recorreu-se a mesma lógia utilizada nas rotas e criou-se o ficheiro "sideContent.js". Este retorna quais os itens a colocar na barra de navegação para cada tipo de utilizador e na Figura 51 pode-se visualizar os três tipos de barra de navegação deste sistema.

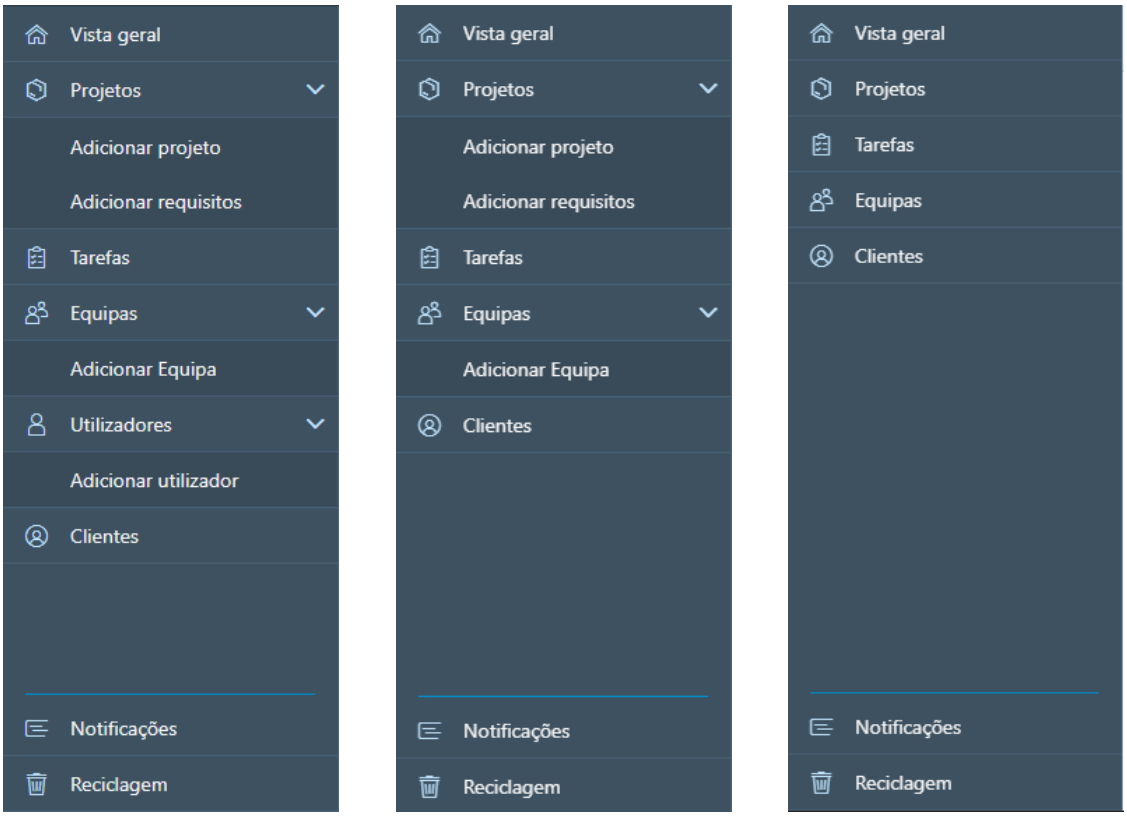


Figura 51: Barra lateral de navegação para o administrador, líder de equipa e membro de equipa respetivamente.

5.1.8 Gestor de notificações

O gestor de notificações foi desenvolvido utilizando a biblioteca socket.io e envolve os ficheiros referidos na Tabela 31. Na Figura 52 está representado a importação desta biblioteca feita no ficheiro index.html.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.2.0/socket.io.js"></script>
```

Figura 52: Importação da biblioteca socket.io.

Tabela 31: Ficheiros que compõem a gestão de notificações.

Ficheiro	Descrição
Notifications.view.xml	Vista que apresenta todas as notificações do utilizador.
Notifications.controller.js	Controlador para a vista "Notifications".
WebSocket.js	Responsável pela conexão do socket com o servidor e enviar e receber notificações.

Para as notificações *pop up* utilizou-se a biblioteca "sap.m.MessageToast" do OpenUI5. Na Figura 53 está um exemplo de quando um utilizador recebe uma notificação de que foi associado a uma tarefa (somente a parte superior da vista).

Figura 53: Receção da notificação de uma nova tarefa.

As notificações podem ser acedidas a partir de três formas neste sistema. Através da barra superior, ao clicar no ícone semelhante a 🗩, onde contém a lista de notificações não lidas. Esta opção é útil quando o utilizador recebe uma notificação e está a executar outras ações na aplicação e necessita de verificar a notificação rapidamente. A segunda opção está na barra de navegação lateral, que mostra todas as notificações lidas e não lidas. A terceira opção está na vista geral que permite ler as novas notificações, assim que o utilizador entra na aplicação. Na Figura 54 está representado, a vermelho, o local de acesso a estas opções.

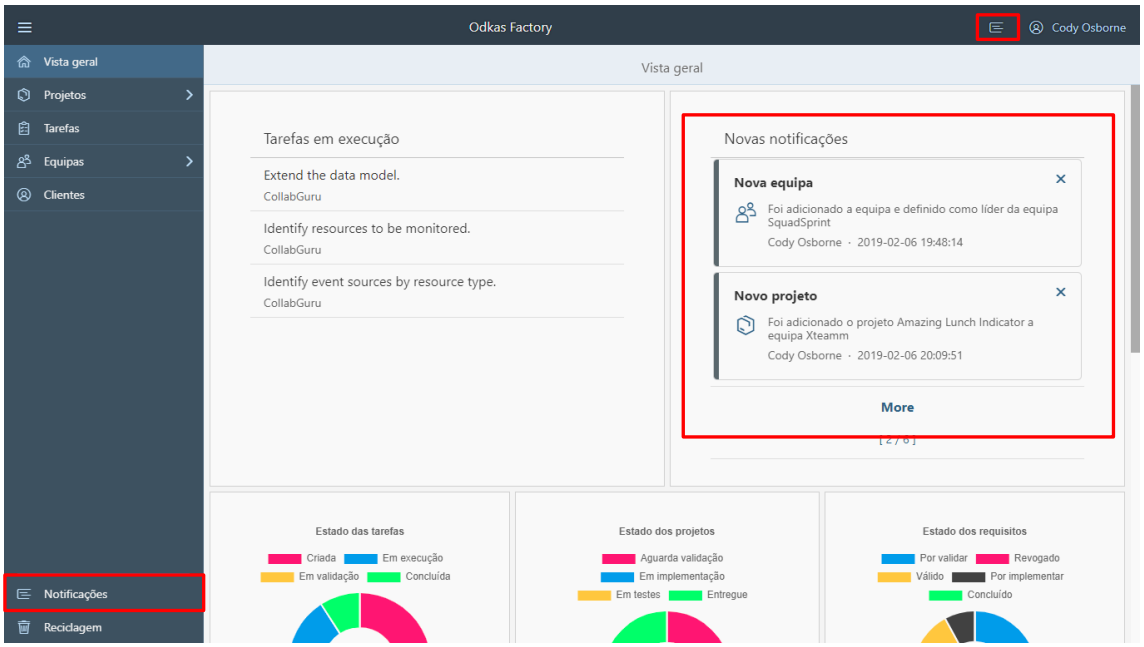


Figura 54: Opções para visualizar as notificações.

Para um utilizador poder receber notificações necessita de subscrever os correspondentes eventos. Na Tabela 32 está representado as subscrições utilizadas neste sistema.

Tabela 32: Subscrições para poder receber notificações.

Evento	Descrição
"notificationsForProject" + projectId	Subscrição para receber notificações sobre um projeto.
"notificationsForTask" + taskId	Subscrição para receber notificações sobre uma tarefa.
"teamLeaderForTeam" + teamId	Subscrição para o líder de uma equipa.
"teamMemberForTeam" + teamId	Subscrição para o membro de uma equipa.
"setUserSocket"	Evento para associar o *socket* ao utilizador no lado do servidor.
"checkUserSocketOK"	Enviar uma mensagem ao cliente infomando de que houve uma conexão do socket com sucesso.
"newNotification"	Subscrição para criar a notificação *pop up*.
"newSubscription"	Evento para subscrever a um novo evento.

O gestor de notificações verifica os requisitos RF06 e RNF02

5.1.9 Validações

As validações foram implementadas em todos os formulários deste sistema. Estas fazem com que o utilizador não envie dados incorretos ou nulos para serem armazenados. Por exemplo, para iniciar sessão os campos "email" e "Palavra-passe" não podem ser nulos e necessitam de ser válidos. Na Figura 55 pode ser observado um aviso, do tipo *pop up,* quando o utilizador clica num campo que possui erro. Os campos com erro são identificados por um contorno a vermelho.

Figura 55: Validação do formulário para iniciar sessão.

5.1.10 Vista geral

Na vista geral além de apresentar as tarefas em execução e as novas notificações, como apresentado na Figura 54, foi também elaborado gráficos circulares para os estados das tarefas, dos projetos e dos requisitos. Além destes, é ainda apresentado um gráfico de linhas composto por uma linha a vermelha, que indica o número de tarefas que termina em cada mês, e por uma linha verde, que indica o número de tarefas que iniciam em cada mês. Por exemplo, a partir da Figura 56 é possível ver que no mês de fevereiro iniciam duas tarefas e terminam também duas tarefas.

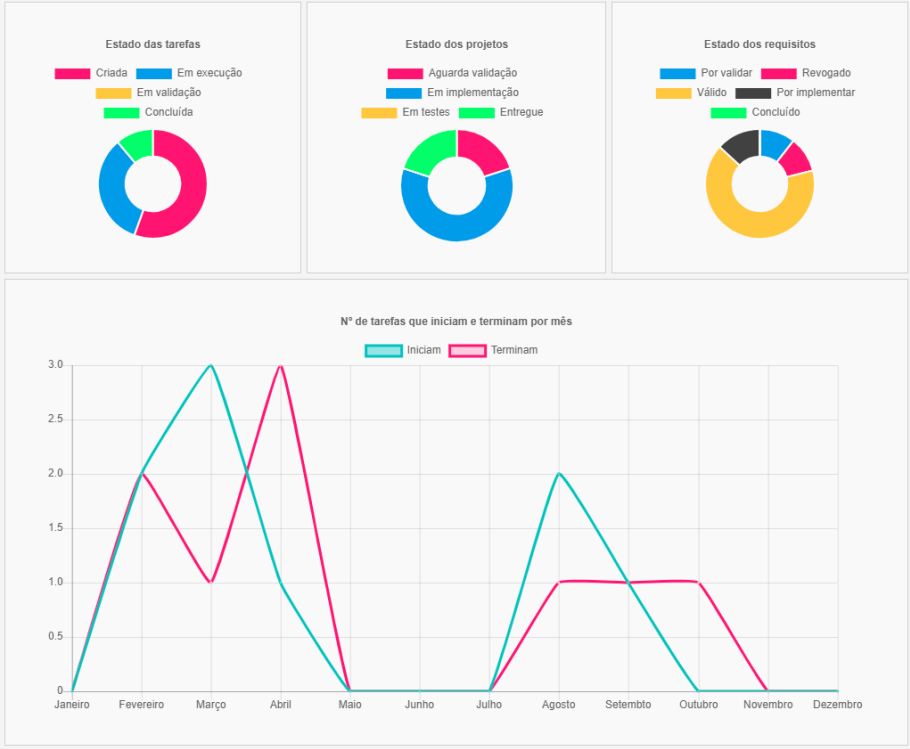


Figura 56: Gráficos na vista geral.

5.2 Criação do servidor FOMS

A criação do servidor FOMS foi baseada numa API REST utilizando a *framework* Express e o NodeJS. Para isto seguiu-se os passos descritos em [46].

O servidor baseia-se em rotas e modelos denominados pelos nomes das entidades do sistema. Como algumas das entidades estão relacionadas entre si, algumas das rotas podem englobar mais do que um modelo. Por exemplo, as rotas para as tarefas ("taskRoutes") utilizam os modelos "assigment" e "requirements". As rotas e modelos criados podem ser visualizados na Figura 57. Estes tiveram como função efetuar operações CRUD sobre as entidades do sistema definidas na base de dados.

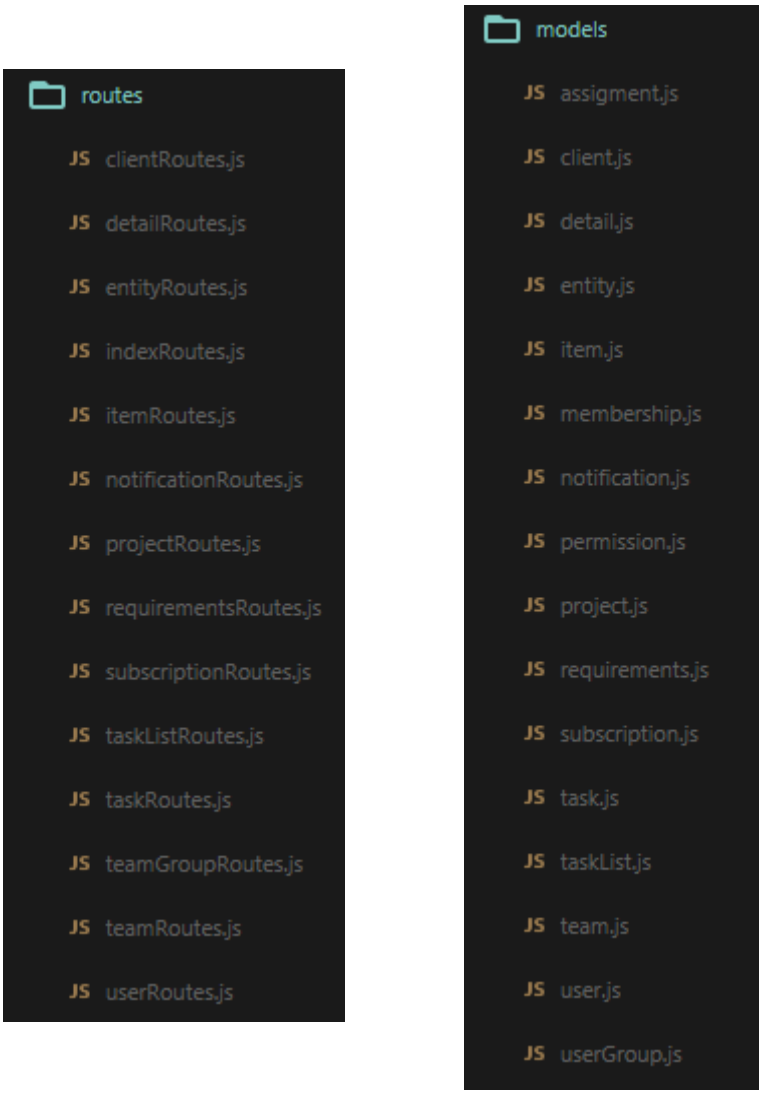


Figura 57: Ficheiros das rotas e modelos da API REST.

5.2.1 Autenticação

A autenticação no lado do servidor é efetuada utilizando as bibliotecas identificadas na Figura 58.

```
const session = require( 'express-session' );
const passport = require( 'passport' );
const MySQLStore = require( 'express-mysql-session' )( session );
```

Figura 58: Bibliotecas de autenticação do lado do servidor.

As funcionalidades das bibliotecas acima mencionadas são:

- "**session**" [47]: responsável pela criação de uma cookie de sessão identificada por um id;

- "**passport**" [48]: utilizado para identificar pedidos efetuados ao servidor a partir do id presente na *cookie*. Na Figura 60 pode-se analisar um caso em que é obtido o id do utilizador utilizando o *passport*.
- "**MySQLStore**" [49]: cria uma tabela de sessão na base de dados, a fim de guardar a sessão. Esta solução permite que a sessão persista, mesmo que o servidor deixe de funcionar.

Referente à palavra passe do utilizador, esta é encriptada na base de dados no momento da criação do utilizador, recorrendo a biblioteca "bcrypt" [50]. No momento de iniciar sessão a palavra passe introduzida pelo utilizador é comparada com a respetiva palavra passe guardada na base de dados através de uma função de comparação, fornecida por esta biblioteca.

> A autenticação no lado do servidor verifica o RNF05 e RNF10

5.2.2 Permissões

As permissões no lado do servidor foram definidas no ficheiro "permissionManager.js". Este é constituído por *middlewares* com a funcionalidade de verificar se uma certa entidade (um utilizador ou grupo de utilizadores, por exemplo uma equipa) tem permissão sobre um determinado item ou ação (por exemplo leitura de um projeto). Na Figura 59 pode-se observar um exemplo de uma rota que executa um pedido para a leitura de um projeto. Esta tem um *middleware* denominado de "canReadItemMiddleware" que está definido na Figura 60. Este tem como função verificar se o utilizador, que tem sessão iniciada, tem permissão de leitura sobre esse projeto. Na Figura 61 pode-se verificar o carregamento de informação sobre a permissão de leitura, que o utilizador tem sobre o projeto.

```
app.get( '/project/:projectId', PermissionManager.canReadItemMiddleware(), ( req, res ) => {
    Project.getProjectById( req.params.projectId, ( err, data ) => {
        var result = {
            data: data,
            canRead: true
        }
        res.json( result );
    });
});
```

Figura 59: Rota para ler um projeto.

```
  canReadItemMiddleware: function() {
      return ( req, res, next ) => {
          if( req.session.passport ){
              var userId = req.session.passport.user
              var itemId = req.params.projectId;
              Permission.canReadItem( userId, itemId, ( err, data ) => {
                  if( data ) return next();
                  var result = {
                      data: "",
                      canRead: false
                  }
                  res.json( result );
              })
          } else {
              res.status( 403 ).send( "Sem autorização" );
          }
      }
  },
```

Figura 60: *Middleware* para verificar se tem permissão para ler um projeto.

```
permissionModel.canReadItem = ( userId, itemId, callback ) => {
    if ( connection ) {
        connection.query(
            'SELECT canRead FROM permission WHERE entity_item_id = ? AND item_id = ?', [ userId, itemId ],
            ( err, result ) => {
                if( err ) throw err;
                var canRead = false;
                if ( result.length > 0 ) {
                    canRead = result[0].canRead;
                }
                callback( null, canRead );
            })
    }
}
```

Figura 61: Modelo para buscar a permissão de leitura que o utilizador tem sobre o item.

> As permissões no lado do servidor complementam o RNF03 e RNF04

5.3 Criação do serviço em *cloud*

Após a implementação do cliente e servidor FOMS, procedeu-se a criação do serviço em *cloud* do sistema. Este foi suportado por dois recursos criados e configurados no Microsoft Azure, um para o cliente FOMS e outro para o servidor FOMS. Dentro de cada recurso criou-se um serviço "Web App", mas com diferentes configurações, utilizando para ambos o plano B1. Este plano oferece 1,45GB de memória e uma computação equivalente a A-Series [51] e custa cerca de 47,02€/mês. Foi escolhido este plano devido a ser um plano indicado para sistemas em desenvolvimento/teste e por o Azure oferecer um crédito de 180€ que pode ser utilizado durante um mês. Após este período o sistema será colocado no plano F1, que é grátis e apenas permite 60 minutos de computação diária.

Referente à configuração, para o cliente FOMS foi necessário criar um "build" da implementação (utilizando o comando "ui5 build -a"), depois criou-se um repositório no GitHub e associou-se este repositório ao serviço do Azure utilizando o Deployment Center do Azure. Para o servidor FOMS após a associação ao seu repositório do GitHub, criou-se uma instância da base de dados na opção "MySQL in App" e efetuou-se as configurações necessárias. Algumas destas foram a indicação da versão do Node.js compatível com a utilizada na API, mudar a biblioteca "bcrypt" para uma versão que pudesse ser executada no Azure e efetuar a configuração de CORS entre o cliente e servidor FOMS.

> Devido a este sistema ser uma aplicação *web* verifica-se o RNF09.

5.4 Conclusão

Neste capítulo foi abordado a etapa de desenvolvimento do sistema, desde a sua criação até a utilização do serviço em *cloud*. Durante a implementação o desenho de *software* desenvolvido serviu como um guia de apoio durante toda a implementação. Teve-se o cuidado em reutilizar o código, criando por isso ficheiros como o "BaseController" e "MyUtils", que são comuns a outros controladores, como

também foi respeitado o padrão MVC. Fatores como a segurança, foram também considerados e implementou-se a gestão de permissões, tanto no cliente como no servidor FOMS. Além das permissões foi utilizado um controlo de sessões que expira após algum tempo de inatividade.

No que toca a funcionalidades, todos os requisitos foram verificados. Tentou-se sempre respeitar os requisitos definidos e encontrar soluções que alcançassem as características de uma ferramenta de gestão de *software,* como por exemplo a utilização de quadros Kanban, planeamento de tarefas, notificações e gráficos. Acerca dos formulários existiu o cuidado para que o utilizador não guarde dados errados, ou envie dados que possam provocar falhas no servidor.

6 AVALIAÇÃO

A avaliação do sistema foi dividida em testes de usabilidade e desempenho.

6.1 Testes de usabilidade

Devido a este sistema ter como utilizadores alvo desenvolvedores de *software,* recorreu-se a estudantes desta área e também aos membros da equipa Odkas Factory. O teste de usabilidade utilizado está disponível no Anexo G. Para este teste foi pedido ao utilizador que pensasse em voz alta durante a execução das tarefas e foi ainda filmado, com o seu consentimento, as operações efetuadas no teclado, rato e ecrã. Este teste foi apresentado a 14 utilizadores e no fim de cada teste foi pedido ao utilizador sugestões, que pudessem melhorar o sistema. As sugestões identificadas como as mais pertinentes foram:

- Ao adicionar requisitos deveria ser possível utilizar toda a linha para adicionar um novo requisito e não apenas o ícone +;
- O botão "Fechar" para a justificação do requisito não é explícito se a justificação fica guardada;
- Permitir associar equipas a projetos também através da vista de detalhes de uma equipa;
- Calendário do planeamento de tarefas não é muito legível;
- Colocar a lista de tarefas em execução também na vista das tarefas e não só na vista geral;
- Utilizar um indicador de nova notificação no ícone de notificação (por exemplo um pequeno ponto a vermelho);
- Apresentar uma legenda para os ícones da tabela de adicionar requisitos;
- O título da tabela das novas notificações, na vista geral, está mal realçado em relação as próprias notificações;
- O botão para mudar de página na vista das equipas está muito transparente;

- Modificar o gráfico das tarefas para algo mais simples e rápido de obter informações;
- Para a gestão das tarefas poderia utilizar uma ComboBox para selecionar o projeto e depois seria apresentado as suas tarefas como também o planeamento. No planeamento ter a possibilidade de validar as tarefas;
- Permitir associar apenas um utilizador a um projeto (atualmente isto é possível ao criar uma equipa com apenas um membro);
- Utilizar notificações sempre que ocorre uma mudança no estado de uma tarefa;
- Definir um limite mínimo para a data de entrega de um projeto;
- Ter em atenção a limpeza dos campos de um formulário após a sua adição.

Após a execução do teste de usabilidade foi pedido a cada utilizador que realizasse um pequeno questionário, que está no Anexo H. Este questionário é baseado no SUS (System Usability Scale) apresentado em [52]. Após os resultados procedeu-se a análise dos dados, que estão disponíveis, também, no Anexo H. Para isto apenas foram consideradas as perguntas referentes a usabilidade (retirando as primeiras duas perguntas). Após a recolha de dados procedeu-se aos cálculos, definidos pelo SUS, e obteve-se a classificação de 79,25. Após aplicar a normalização dos dados, apresentada em [53], obteve-se uma classificação de A- que está contida num intervalo percentual de 85% a 89%.

6.2 Testes de desempenho

Para os testes de desempenho recorreu-se aos gráficos de monitorização do Azure. Teve-se em conta os gráficos relacionados com o número de pedidos efetuados e a média do tempo de resposta, ambos em relação a um período de 15 minutos. Para estes testes foram selecionados os dados no intervalo de tempo correspondente a alguns dos testes de usabilidade.

Para o cliente FOMS os dados recolhidos referentes a soma do número de pedidos a cada 15 minutos e o tempo médio dos pedidos a cada 15 minutos estão

representados na Figura 62 e Figura 63 respetivamente. Destes pode-se retirar que a média de tempo de resposta nunca foi superior a 250 milissegundos, aproximadamente. Ao analisar ambos os gráficos, pode-se concluir que o tempo de resposta não depende, unicamente do número de pedidos. Isto porque se olharmos para ambos os gráficos na hora 16:14:15 obtêm-se um máximo relativo no gráfico do tempo médio de resposta, enquanto que na soma do número de pedidos é obtido um máximo relativo só nos próximos 15 minutos e nesta altura é obtido um mínimo relativo no tempo médio de resposta. Isto poderá acontecer devido a velocidade da rede no momento do pedido e também da quantidade de dados presente na operação. O mesmo já não acontece às 17:14:15, obtendo um máximo relativo em ambos os gráficos.

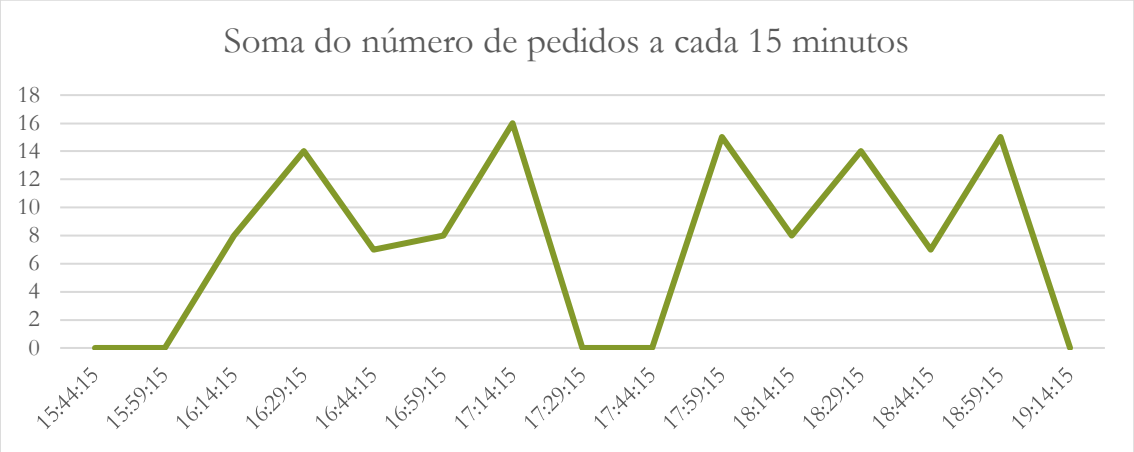


Figura 62: Gráfico da soma do número de pedidos a cada 15 minutos do cliente FOMS.

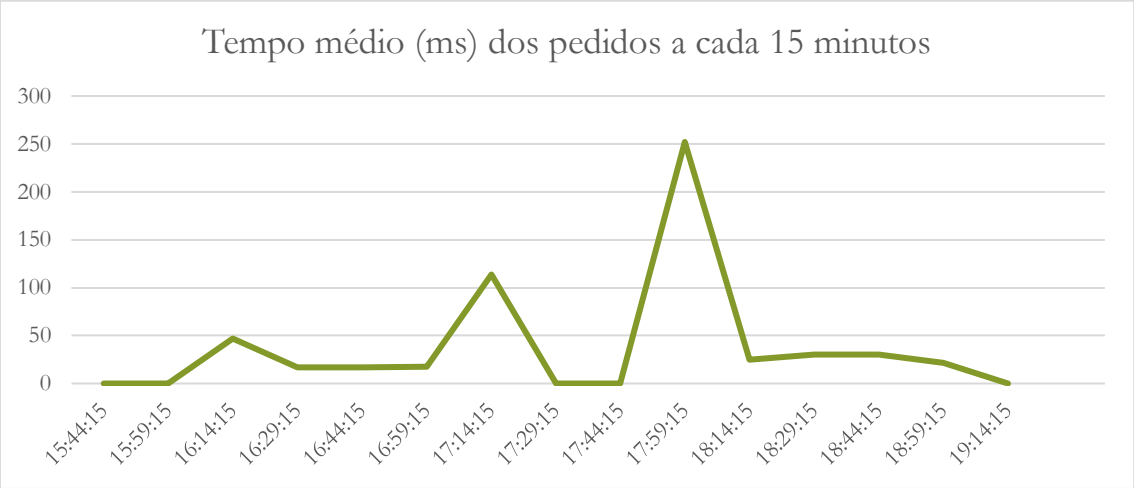


Figura 63: Gráfico da média do tempo dos pedidos a cada 15 minutos do cliente FOMS.

Para o servidor FOMS o gráfico da Figura 64 representa a soma do número de pedidos a cada 15 minutos e a Figura 65 representa o tempo médio dos pedidos a cada 15 minutos. O mesmo comportamento observado para o cliente FOMS acontece também no servidor FOMS, sendo que o tempo por pedido não depende unicamente do número de pedidos. Referente ao máximo absoluto do tempo médio por pedido é alcançado as 17:14:15 e tem o valor de 2 segundos, aproximadamente. Comparando este valor com o valor do gráfico da figura 59, observa-se a ocorrência de um máximo relativo neste, o que poderá ser uma consequência da velocidade lenta da rede. Isto porque se compararmos com o que ocorre às 18:14:15 em que se tem o máximo absoluto de números de pedidos e obtém-se um tempo de resposta inferior a 200 milissegundos.

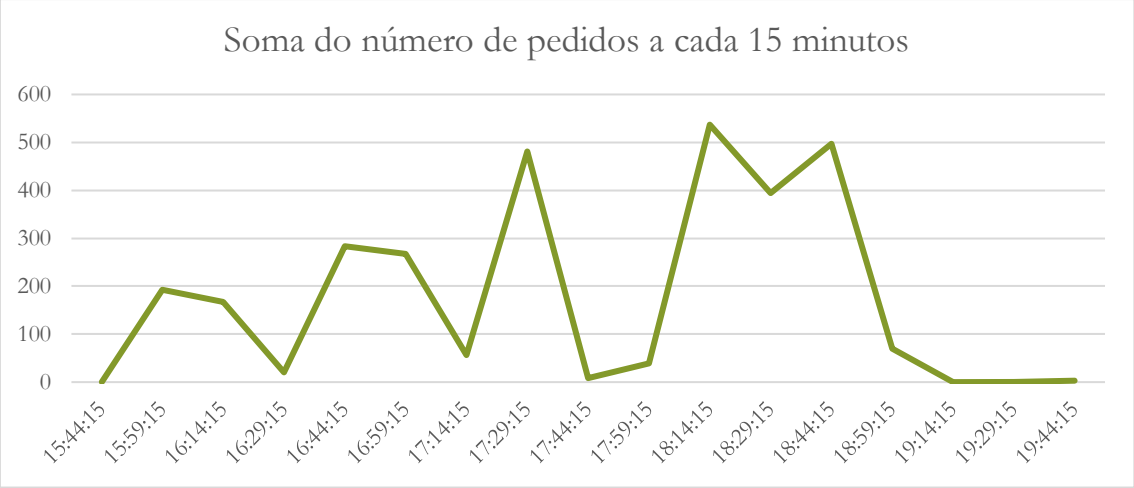


Figura 64: Gráfico da soma do número de pedidos a cada 15 minutos.

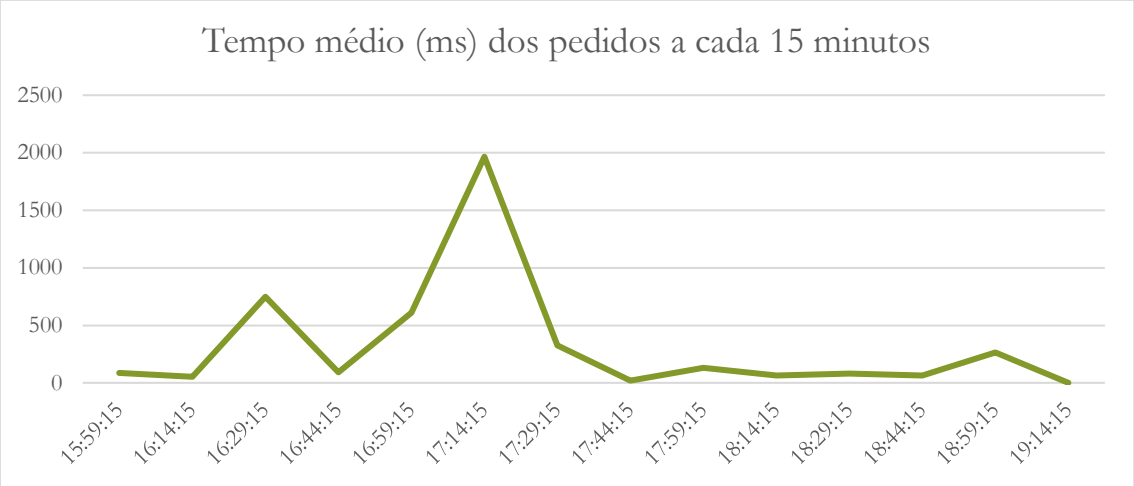


Figura 65: Gráfico do tempo médio de pedidos a cada 15 minutos.

Por uma questão de tempo e disponibilidade de número de utilizadores disponíveis para fazer testes recorreu-se a este tipo de abordagem. Embora não tenham surgido de uma utilização excessiva do sistema, podem ser utilizados como ponto de partida para testes mais exaustivos.

> Os testes de desempenho verificam o RNF01 e RNF02, uma vez que não houve operações superiores a 3 segundos ou a 5 segundos.

6.3 Conclusão

Nesta secção foram elaborados os testes de usabilidade utilizando um teste composto por tarefas e um questionário baseado no SUS. Teve-se em atenção o tipo de utilizador final para esta aplicação e recorreu-se a estudantes de engenharia informática e a membros da equipa Odkas Factory. Destes testes pode-se averiguar problemas surgidos e sugestões que devem ser utilizadas para melhorias futuras. O questionário efetuado permitiu obter uma classificação quantitativa a cerca da usabilidade. Pode-se concluir, a partir de esta classificação, que o sistema obtém uma avaliação positiva a cerca da usabilidade.

Referente aos testes de desempenho o Microsoft Azure revelou-se muito útil, ao apresentar gráficos de monitorização e permitir carregar os dados em formato do Excel para que possam ser tratados. Destes testes podemos concluir que para a utilização, com um utilizador, o tempo médio de pedidos atingiram um máximo de 250 milissegundos para o cliente FMOS e 2 segundos, aproximadamente, para o servidor FMOS. Em testes futuros aconselha-se a realização de testes mais exaustivos utilizando, por exemplo, ferramentas de simulação de pedidos recorrendo a *scripts* ou simular utilizações reais com vários utilizadores.

7 CONCLUSÃO

Nesta secção será apresentada uma conclusão geral ao projeto realizado, referindo as conclusões obtidas do trabalho realizado e apresentar algumas das sugestões que, futuramente, poderão ser implementadas.

7.1 Trabalho realizado

Todo o trabalho realizado foi de acordo com a ideia de criar uma plataforma que pudesse suportar uma gestão uniformizada e centralizada de projetos, para a equipa Odkas Factory. Para isto foi necessário identificar o problema surgido e quais os objetivos a alcançar.

Sabendo que a solução a alcançar iria se enquadrar no tipo de ferramentas baseadas em desenvolvimento de processos de *software*, foi elaborado um estudo sobre este assunto. O estudo foi abordado utilizando uma perspetiva de evolução, começando por uma introdução histórica até alcançar a ideia de que, ainda hoje, é um tema que está em constante desenvolvimento, não havendo um processo de *software* geral e infalível. Isto deve-se ao facto de ser um desenvolvimento complexo e suscetível a muitas variáveis. Ainda referente à parte de estudos, com a finalidade de comparar ferramentas deste tipo existentes no mercado, elaborou-se um estudo sobre estas e constatou-se a existência de ferramentas ágeis e outras que apresentam soluções mais básicas de colaboração, havendo muitas semelhanças entre todas elas. Retirou-se, também, características que se acabou por utilizar neste sistema, como a utilização de quadros Kanban. Após isto, fez-se ainda um outro estudo sobre as tecnologias que se iria utilizar no desenvolvimento deste projeto, como o OpenUI5 e o Node.js.

Em seguida procedeu-se a fase de análise e desenho, caracterizada pelo levantamento de requisitos e a criação de cenários definindo, deste modo, as principais funcionalidades do sistema. A elaboração de diagramas de casos de utilização, de robustez e modelo de dados permitiu identificar os utilizadores do sistema e os seus papeis como, também as entidades e as suas relações. Ao notar a necessidade de que alguns componentes possuíam estados, criou-se os diagramas de

estado. Posto isto procedeu-se a elaboração de protótipos e desenho da arquitetura do sistema.

Após a análise e o desenho estava, assim, definido a base do sistema e prosseguiu-se com a sua implementação. Esta foi dividida no desenvolvimento do cliente FMOS e do servidor FMOS. Para o cliente FMOS foi utilizado o OpenUI5 que obrigou a algum estudo e tomadas de decisões acerca deste. Dentro destas as mais cruciais foram a escolha do ambiente de desenvolvimento, que modelo de dados utilizar, que tipo de relação adotar entre as vistas e controladores e. Posteriormente, iniciou-se a implementação dos gestores que constituem este sistema. O servidor FMOS foi desenvolvido, de uma forma paralela ao cliente FMOS, criando as rotas e modelos necessários a este. Após a implementação foi efetuado o *build* do projeto e colocado no Azure.

Após a criação do serviço em *cloud* iniciou-se a fase de avaliação onde se efetuou os testes de usabilidade e desempenho. Deste pôde-se obter as primeiras reações na utilização do sistema pelo utilizador final identificando, assim, os problemas e melhoramentos a efetuar. Aplicou-se, ainda, o SUS a fim de obter uma avaliação quantitativa da usabilidade do sistema. Acerca do desempenho do sistema os máximos absolutos obtidos não foram de grande ordem, no entanto, é necessário ter em conta que estes resultados resultaram da utilização de um único utilizador durante os testes de usabilidade.

Conclui-se, desta forma, o processo de desenvolvimento deste projeto. A aplicação possui um funcionamento agnóstico em relação aos processos de desenvolvimento de *software*, centrando-se mais na gestão de tarefas. Referente a utilização de tecnologias, o OpenUI5 é uma *framework* que obriga a algum estudo até conseguir uma utilização confiante, devido a algumas particularidades, como por exemplo, à abordagem de utilização dos modelos e a forma como efetua as ligações entre os modelos e as vistas. Embora o OpenUI5 possua bibliotecas em JS, este requer algum estudo sobre a sua documentação, por possuir métodos próprios. Um ponto negativo é que não possui uma grande comunidade, o que dificulta em situações de ocorrência de erros encontrar uma solução. Na utilização das vistas é necessário ter em conta se as *tasgs* XML filho são compatíveis com as *tasgs* pai, isto traz alguma

complicação ao tentar juntar componentes de diferentes bibliotecas, o que poderá inibir, de certa forma, a criatividade. Quando se efetua o envio dos dados para o servidor é necessário inserir estes mesmos dados no modelo de dados da vista, para que o utilizador não tenha de recarregar a página para ver a alteração dos dados. Os pontos fortes situam-se numa boa e detalhada documentação, além disto, possui muitas amostras úteis de componentes que se tornou vantajosa para a criação das vistas. Referente ao Node.js, pode-se considerar este como uma boa opção para o desenvolvimento de uma API, devido a sua larga comunidade, grande diversidade de bibliotecas e bom desempenho. Acerca do Azure, este demonstrou ser uma boa opção para a utilização de serviços em *cloud*, por apresentar vários planos de pagamento, incluindo um plano grátis. Permite a utilização de uma conta gratuita durante um ano e ainda um crédito para ser utilizado em serviços durante um mês.

7.2 Trabalho futuro

Apesar de os requisitos terem sido todos verificados, existe algumas melhorias e funcionalidade que poderão ser aplicadas, tal como:

- Criação de subtarefas;
- Aplicar a funcionalidade de *drag and drop* aos cartões Kanban, para mudar as tarefas de lista;
- Criar subequipas, podendo definir equipas com especialidades;
- Adicionar um sistema de prioridades as notificações e tarefas;
- Utilizar uma classificação para cada utilizador sobre a realização de tarefas no prazo estipulado;
- Utilizar um gráfico *burn-down* que indica qual o trabalho por fazer em relação ao tempo disponível;
- Utilizar aptidões a nível de desenvolvimento de *software* para cada membro de equipa;
- Utilizar um gráfico de Gantt para o planeamento de tarefas.

Além destas funcionalidades, que poderão ser implementadas, as sugestões apresentadas pelos utilizadores após os testes de usabilidade (secção 6.1) deverão ser, também, consideradas.

REFERÊNCIAS

[1] F. P. B. Jr, *The Mythical Man-Month, Anniversary Edition: Essays On Software Engineering, Portable Documents*. Pearson Education, 1995.

[2] C. Deephouse, T. Mukhopadhyay, D. R. Goldenson, e M. I. Kellner, «Software Processes and Project Performance», *J. Manag. Inf. Syst.*, vol. 12, n. 3, pp. 187–205, Winter95/96 1995.

[3] B. Singh e S. Gautam, «The Impact of Software Development Process on Software Quality: A Review», em *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016, pp. 666–672.

[4] I. Sommerville e T. Rodden, «Human, Social and Organisational Influences on the Software Process», p. 21.

[5] «Odkas». [Online]. Disponível em: https://www.odkas.com/. [Acedido: 28-Nov-2018]

[6] «OpenUI5». [Online]. Disponível em: https://openui5.org/. [Acedido: 29-Nov-2018]

[7] J. Lasky, «Software development», *Salem Press Encyclopedia*. Salem Press, 2018.

[8] D. Chevers, «Software Process Improvement: Awareness, Use, and Benefits in Canadian Software Development Firms», *Mejora Proceso Softw. Concientización Uso Benef. En Desarro. Softw. Can.*, vol. 57, n. 2, pp. 170–177, Abr. 2017.

[9] F. H. Knight, *Risk, Uncertainty and Profit*. Courier Corporation, 2012.

[10] J. Bach, «Enough about process: what we need are heroes», *IEEE Softw.*, vol. 12, n. 2, pp. 96–98, Mar. 1995.

[11] M. Kuhrmann, G. Kalus, e M. Then, «The Process Enactment Tool Framework —Transformation of software process models to prepare enactment», *Sci. Comput. Program.*, vol. 79, pp. 172–188, Jan. 2014.

[12] W. S. Humphrey, «S o b a r e Process Improvement at Hughes Aircraft», *IEEE Softw.*, p. 13.

[13] S. P. Patil e J. R. Neve, «Productivity Improvement of Software Development Process Through Scrumban: A Practitioner's Approach», em *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, 2018, pp. 314–318.

[14] A. Mihalache, «Project Management Tools for Agile Teams», *Inform. Econ.*, vol. 21, n. 4, pp. 85–93, Out. 2017.

[15] J. Raigoza e R. Thakkar, «Browser Performance of JavaScript Framework, SAPUI5 amp;amp; jQuery», em *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2016, pp. 1420–1421.

[16] «OpenUI5 SDK - MVC». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/topic/91f233476f4d1014b6dd926db0e91070. [Acedido: 10-Dez-2018]

[17] «OpenUI5 SDK - Views». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/topic/91f27e3e6f4d1014b6dd926db0e91070. [Acedido: 17-Dez-2018]

[18] «OpenUI5 SDK - Models». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/topic/d2c8cf7ae19d447aad5b5ce40e3b14e3. [Acedido: 17-Dez-2018]

[19] «OpenUI5 SDK - Data Binding». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/topic/68b9644a253741e8a4b9e4279a35c247. [Acedido: 06-Jan-2019]
[20] N. js Foundation, «About», *Node.js*. [Online]. Disponível em: https://nodejs.org/en/about/. [Acedido: 11-Jan-2019]
[21] L. Liang, L. Zhu, W. Shang, D. Feng, e Z. Xiao, «Express supervision system based on NodeJS and MongoDB», em *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017, pp. 607–612.
[22] K. Lei, Y. Ma, e Z. Tan, «Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js», em *2014 IEEE 17th International Conference on Computational Science and Engineering*, 2014, pp. 661–668.
[23] «Microsoft Azure Cloud Computing Platform & Services». [Online]. Disponível em: https://azure.microsoft.com/en-us/. [Acedido: 10-Fev-2019]
[24] Atlassian, «Jira Software Data Center», *Atlassian*. [Online]. Disponível em: https://www.atlassian.com/enterprise/data-center/jira. [Acedido: 08-Jan-2019]
[25] «JIRA Tutorial: A Complete Guide for Beginners». [Online]. Disponível em: https://www.guru99.com/jira-tutorial-a-complete-guide-for-beginners.html. [Acedido: 08-Jan-2019]
[26] H. M. Sarkan, T. P. S. Ahmad, e A. A. Bakar, «Using JIRA and Redmine in requirement development for agile methodology», em *2011 Malaysian Conference in Software Engineering*, 2011, pp. 408–413.
[27] O. C. Buturugă, V. M. Gogoi, e I. A. Prodan, «Agile Project Management Tools», *Econ. Inform.*, vol. 16, n. 1, pp. 19–26, Jan. 2016.
[28] «VersionOne Software - 2019 Reviews, Pricing & Demo». [Online]. Disponível em: https://www.softwareadvice.com/project-management/versionone-profile/. [Acedido: 08-Jan-2019]
[29] «VersionOne Reviews: Overview, Pricing and Features», *Financesonline.com*. [Online]. Disponível em: https://reviews.financesonline.com/p/versionone/. [Acedido: 08-Jan-2019]
[30] M. Manole e M.-Ş. Avramescu, «A Comparative Analysis of Agile Project Management Tools», *Econ. Inform.*, vol. 17, n. 1, pp. 25–31, Jan. 2017.
[31] «CA Agile Central Software Review: Overview – Features – Pricing», *Project-Management.com*. 12-Abr-2018 [Online]. Disponível em: https://project-management.com/ca-agile-central-software-review/. [Acedido: 15-Jan-2019]
[32] «Wrike Reviews: Overview, Pricing and Features of wrike.com | FinancesOnline», *Financesonline.com*. [Online]. Disponível em: https://reviews.financesonline.com/p/wrike/. [Acedido: 15-Jan-2019]
[33] B. J. Duffy, 2017 11:45AM EST November 6, e 2017 November 6, «Microsoft Planner», *PCMAG*. [Online]. Disponível em: https://www.pcmag.com/review/356065/microsoft-planner. [Acedido: 23-Jan-2019]
[34] «Trello Reviews: Overview, Pricing and Features», *Financesonline.com*. [Online]. Disponível em: https://reviews.financesonline.com/p/trello/. [Acedido: 15-Jan-2019]
[35] H. A. Johnson, «Trello», *J. Med. Libr. Assoc.*, vol. 105, n. 2, pp. 209–211, Abr. 2017.

[36] «Asana Reviews: Overview, Pricing and Features», *Financesonline.com*. [Online]. Disponível em: https://reviews.financesonline.com/p/asana/. [Acedido: 23-Jan-2019]
[37] B. B. M. and J. Duffy, 2018 1:08PM EST October 2, e 2018 October 2, «Asana», *PCMAG*. [Online]. Disponível em: https://www.pcmag.com/article2/0,2817,2408011,00.asp. [Acedido: 23-Jan-2019]
[38] D. Rosenberg e K. Scott, *Applying Use Case Driven Object Modeling with UML: An Anotated E-commerce Example*. Addison-Wesley Professional, 2001.
[39] «UML State Machine Diagrams - Overview of Graphical Notation». [Online]. Disponível em: https://www.uml-diagrams.org/state-machine-diagrams.html. [Acedido: 20-Jan-2019]
[40] L. L. Constantine e L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Pearson Education, 1999.
[41] «OpenUI5 SDK - Development Environment». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/topic/7bb04e05f9484e1b95b38a2e48ecef4f. [Acedido: 26-Jan-2019]
[42] *OpenUI5 Sample App. Contribute to SAP/openui5-sample-app development by creating an account on GitHub*. SAP, 2019 [Online]. Disponível em: https://github.com/SAP/openui5-sample-app. [Acedido: 26-Jan-2019]
[43] «OpenUI5 SDK - Controller». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/api/sap.ui.core.mvc.Controller/constructor. [Acedido: 17-Dez-2018]
[44] «OpenUI5 SDK - Samples». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/controls. [Acedido: 27-Jan-2019]
[45] «OpenUI5 SDK - Planning Calendar». [Online]. Disponível em: https://openui5.hana.ondemand.com/#/entity/sap.m.PlanningCalendar. [Acedido: 28-Jan-2019]
[46] Onejohi, «Building a simple REST API with NodeJS and Express.», *Medium*. 28-Jun-2018 [Online]. Disponível em: https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9. [Acedido: 29-Jan-2019]
[47] «express-session-expire-timeout», *npm*. [Online]. Disponível em: https://www.npmjs.com/package/express-session-expire-timeout. [Acedido: 29-Jan-2019]
[48] «passport», *npm*. [Online]. Disponível em: https://www.npmjs.com/package/passport. [Acedido: 29-Jan-2019]
[49] «express-mysql-session», *npm*. [Online]. Disponível em: https://www.npmjs.com/package/express-mysql-session. [Acedido: 29-Jan-2019]
[50] «bcrypt», *npm*. [Online]. Disponível em: https://www.npmjs.com/package/bcrypt. [Acedido: 29-Jan-2019]
[51] «Virtual Machine series | Microsoft Azure». [Online]. Disponível em: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/series/. [Acedido: 10-Fev-2019]

[52] A. I. Martins, A. F. Rosa, A. Queirós, A. Silva, e N. P. Rocha, «European Portuguese Validation of the System Usability Scale (SUS)», *Procedia Comput. Sci.*, vol. 67, pp. 293–300, 2015.
[53] J. R. Lewis, «The System Usability Scale: Past, Present, and Future», *Int. J. Human–Computer Interact.*, vol. 34, n. 7, pp. 577–590, Jul. 2018.

ANEXOS

Anexo A – Kanban VS Scrum

Tabela 33: Tabela de comparação entre o Kanban e o Scrum.

Similaridades [1]	Diferenças[2]	
	Kanban	**Scrum**
• Metodologias ágeis; • Dividir o trabalho em pequenas partes; • Esquipas organizadas; • Entrega de trabalho significativo em datas pré-estabelecidas; • Efetuar alterações rapidamente ao trabalho; • Limitar trabalho em processo;	• Mede a liderança e o tempo.	• Mede a velocidade.
	• Melhora o conceito existente;	• Necessário começar com o conceito certo.
	• Visualiza o que está a acontecer	• Visualiza uma ideia
	• Prescreve iterações.	• Prescreve papéis.
	• Limita o trabalho me processo por fluxo de trabalho.	• Limita trabalho em processo por iteração.
	• Permite adicionar itens, desde que haja capacidade disponível.	• Não permite adicionar itens numa iteração a decorrer.
	• Priorização é opcional.	• Prescreve um produto *backlog* priorizado.
	• Comprometimento é opcional.	• A equipa compromete-se a executar uma determinada quantidade de trabalho para uma determinada iteração.

[1] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, e P. Ozcan, «A statistical analysis of the effects of Scrum and Kanban on software development projects», *Robot. Comput.-Integr. Manuf.*, vol. 43, pp. 59–67, Fev. 2017.

[2] «Kanban-vs-Scrum.pdf». [Online]. Disponível em: https://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf. [Acedido: 14-Fev-2018]

Anexo B – Modelo relacional da base de dados

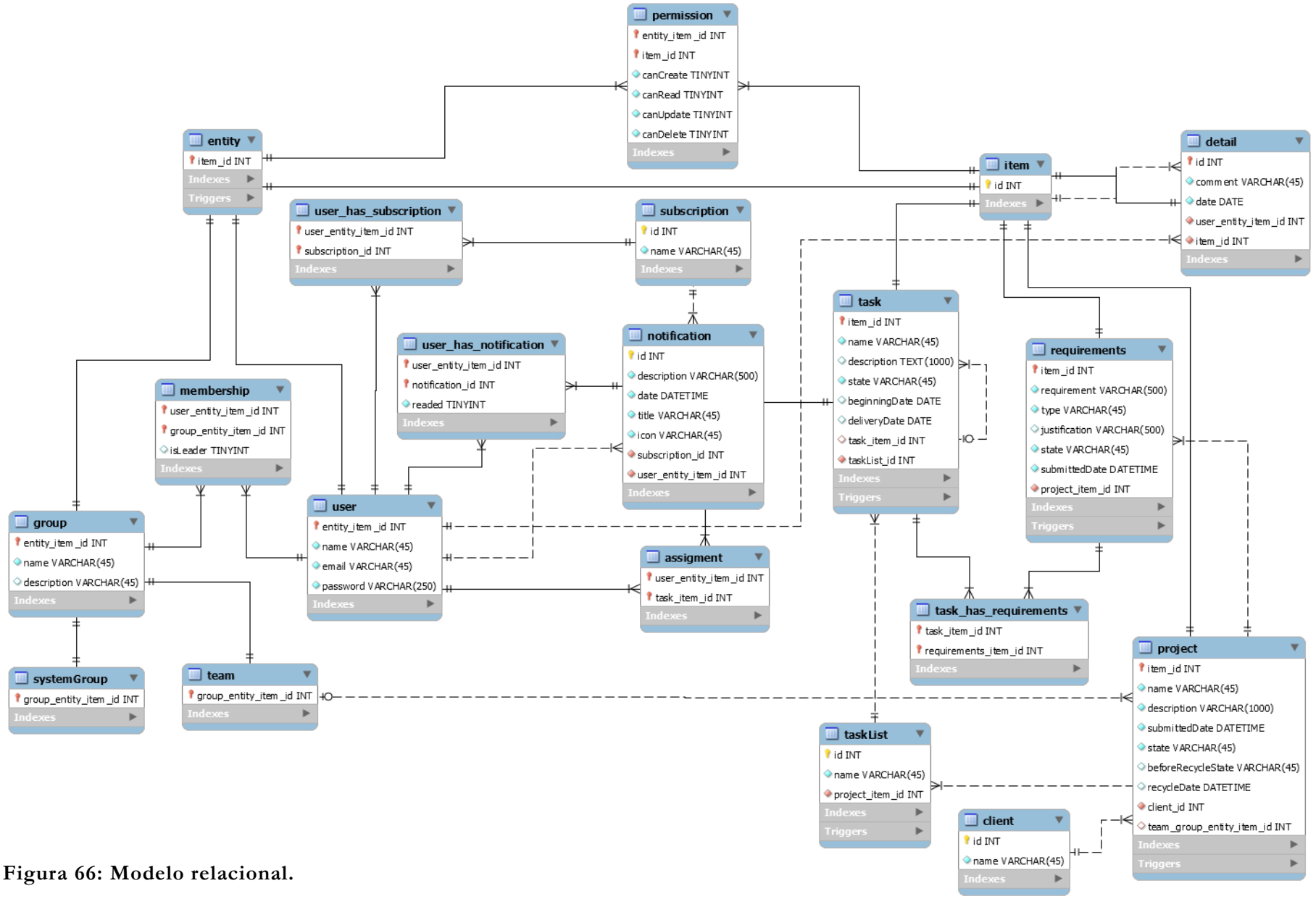


Figura 66: Modelo relacional.

Anexo C – Mapa de navegação

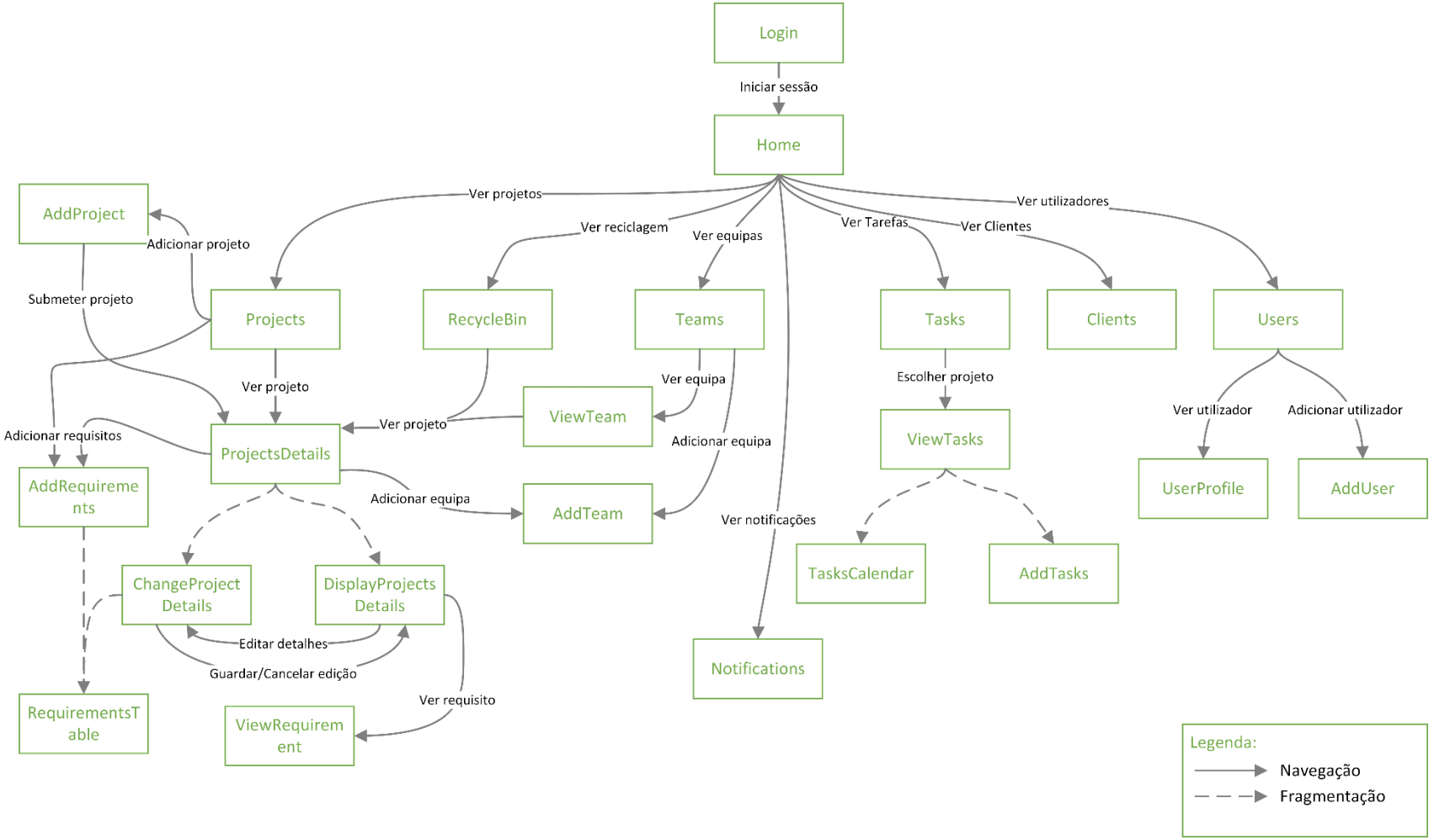


Figura 67: Mapa de navegação.

Anexo D – Protótipos de baixa fidelidade

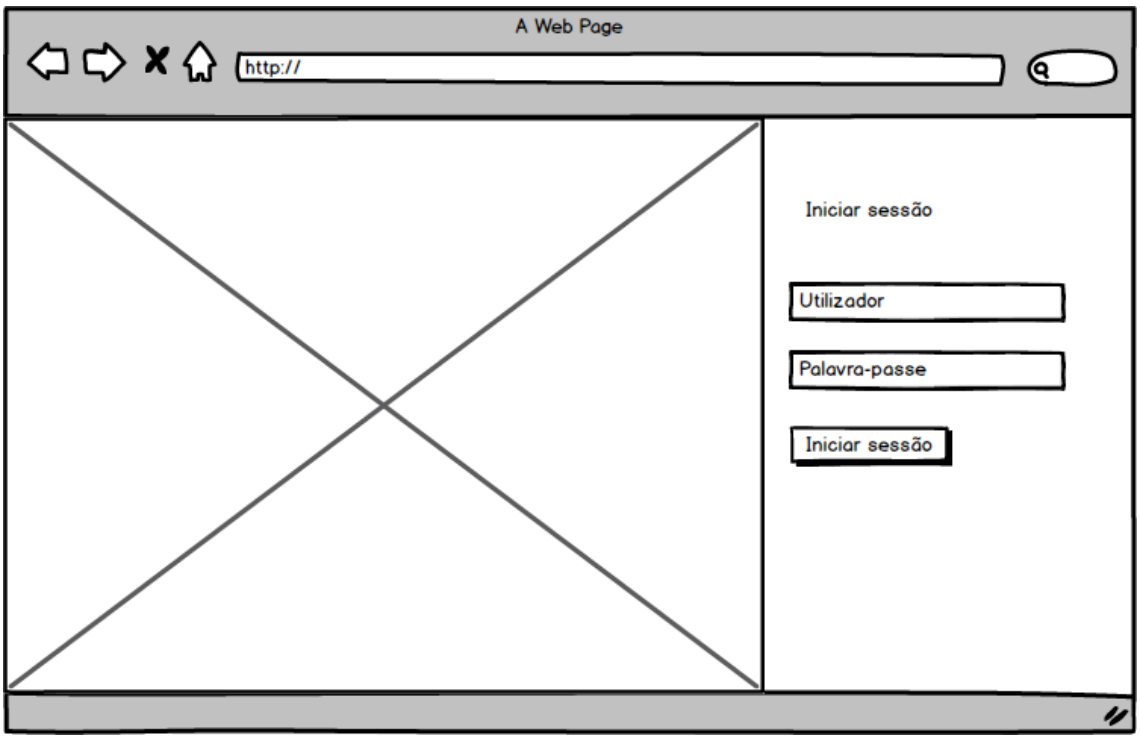


Figura 68: Protótipo da vista para iniciar sessão.

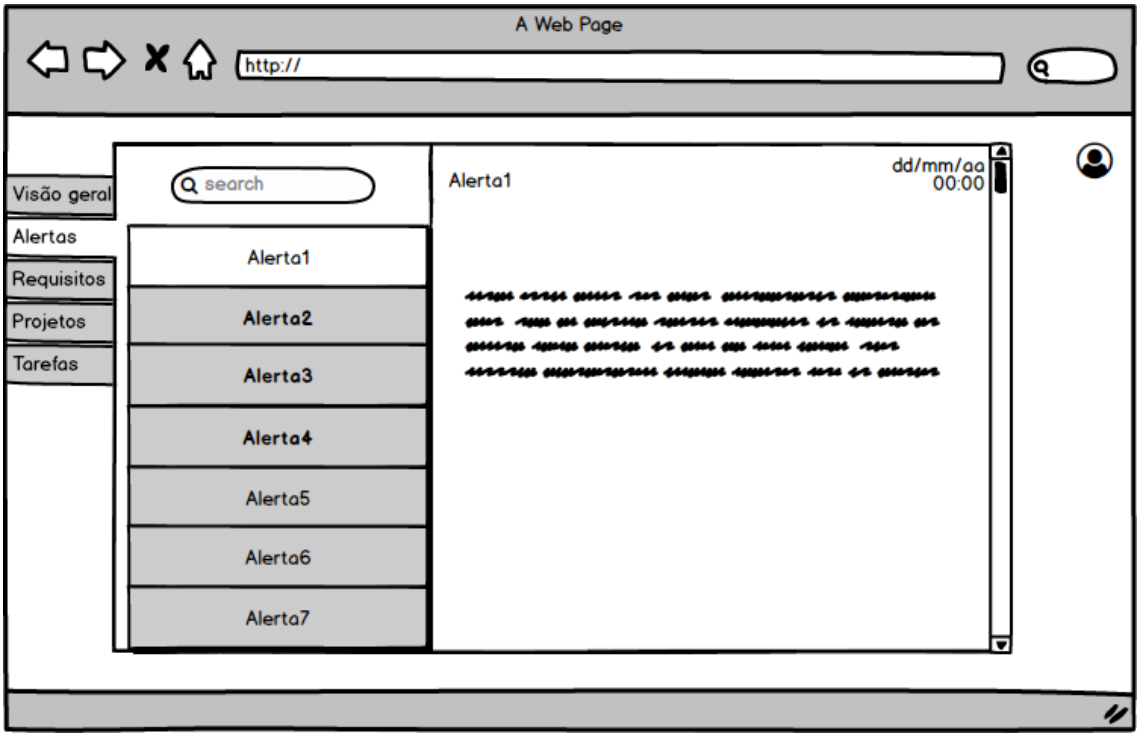


Figura 69: Protótipo para a vista de alertas/notificações.

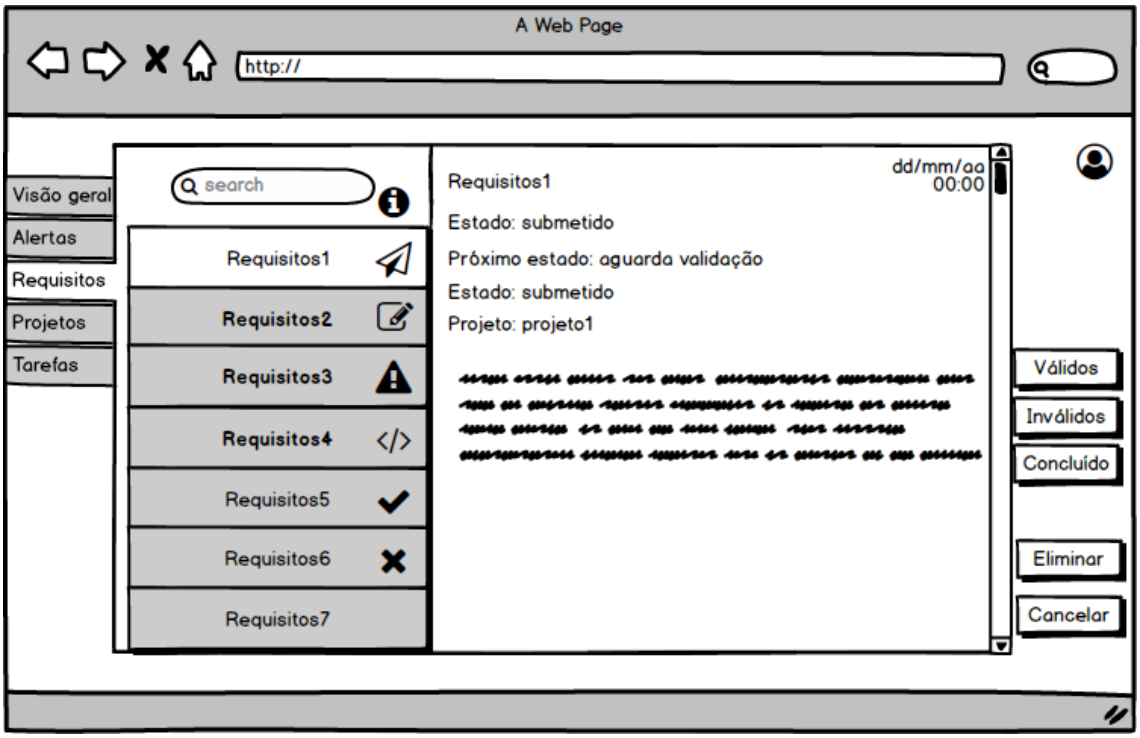


Figura 70: Protótipo para o líder de equipa gerir requisitos.

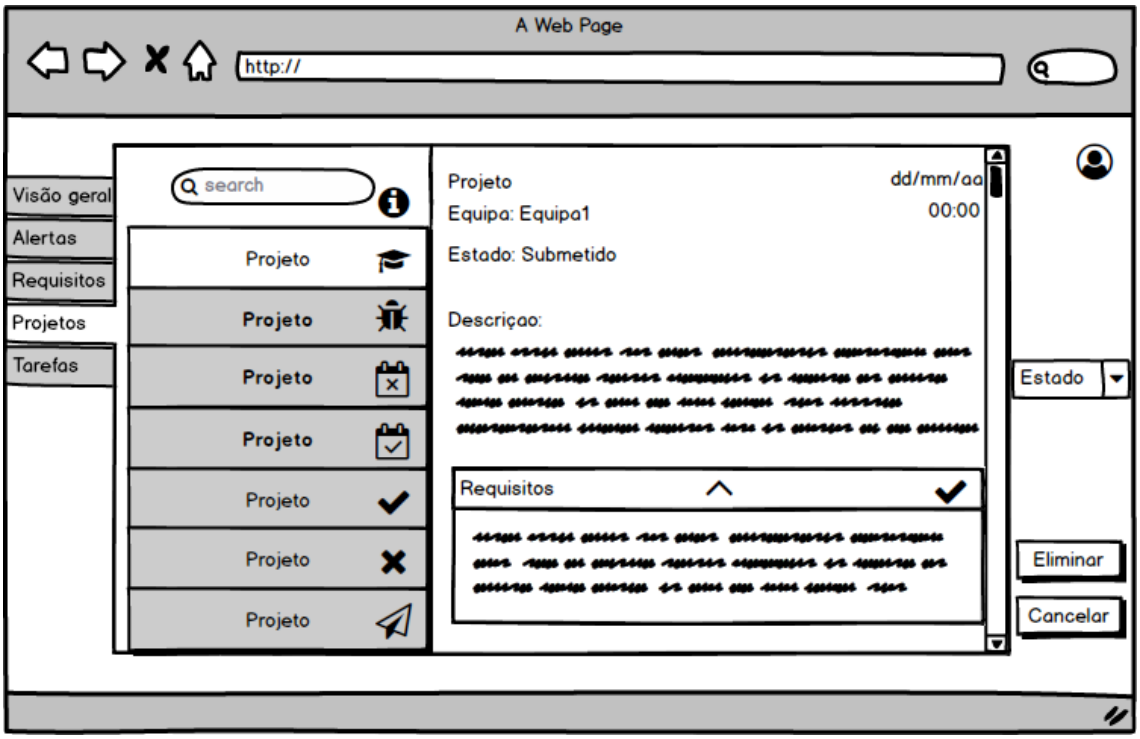


Figura 71: Protótipo para o líder de equipa gerir projetos.

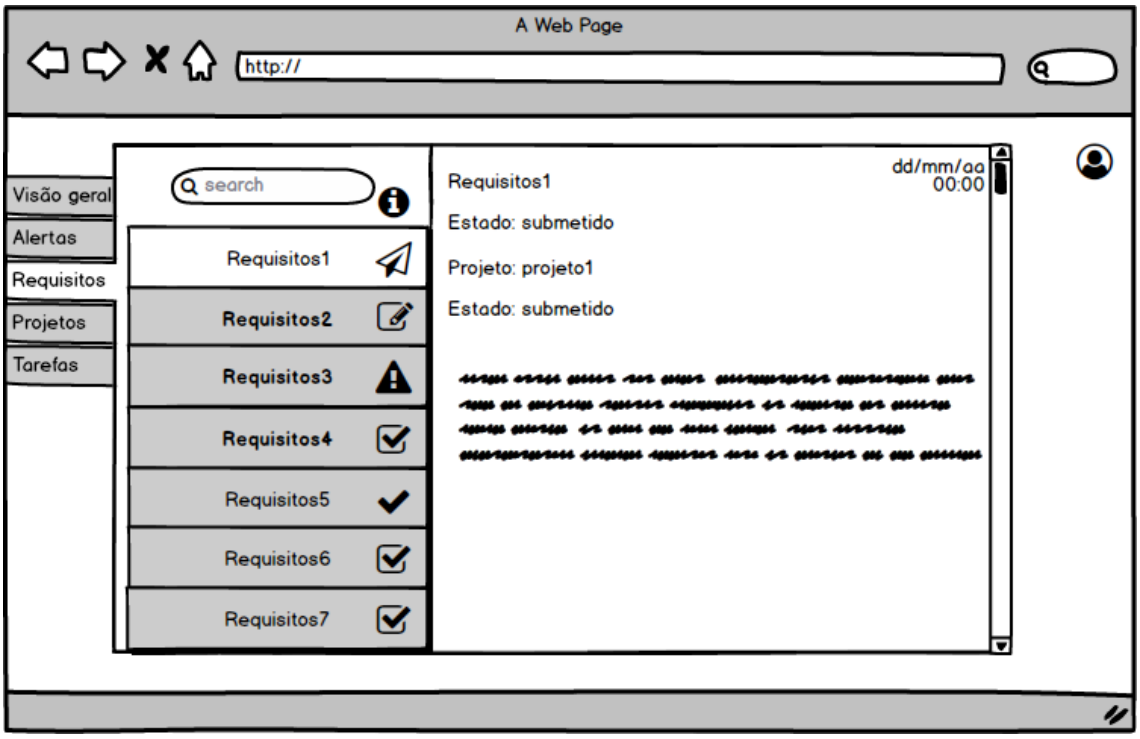


Figura 72: Protótipo para o membro de equipa ver os requisitos.

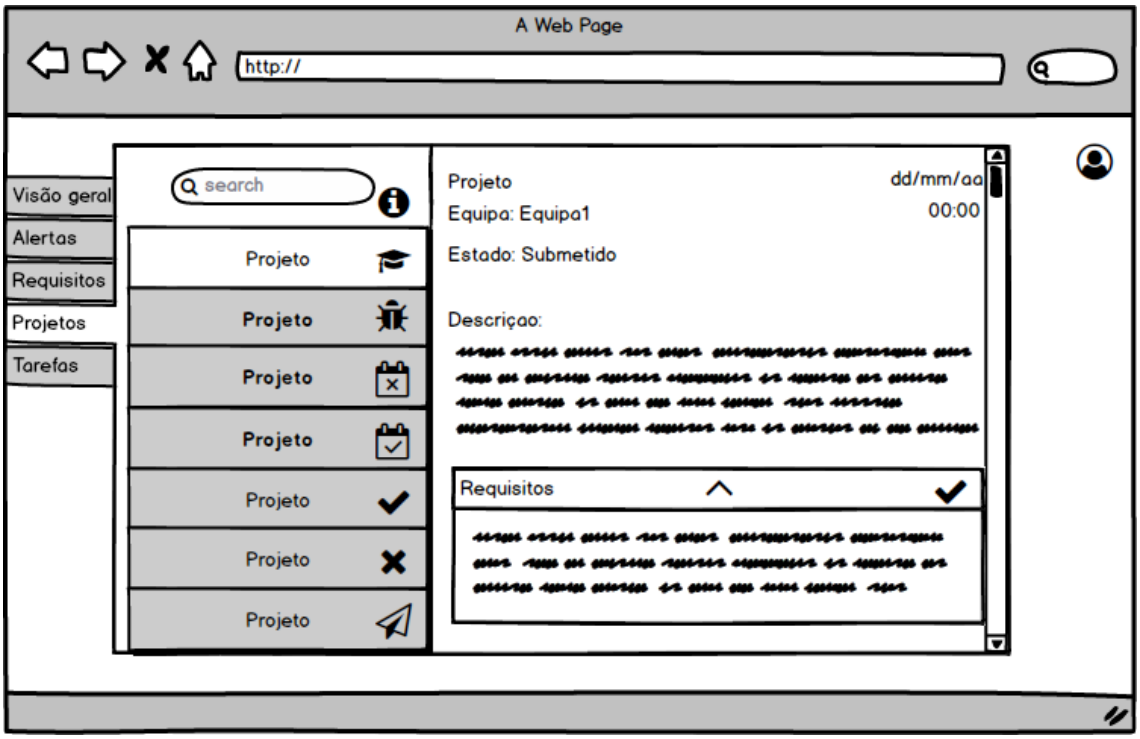


Figura 73: Protótipo para o membro de equipa ver os projetos.

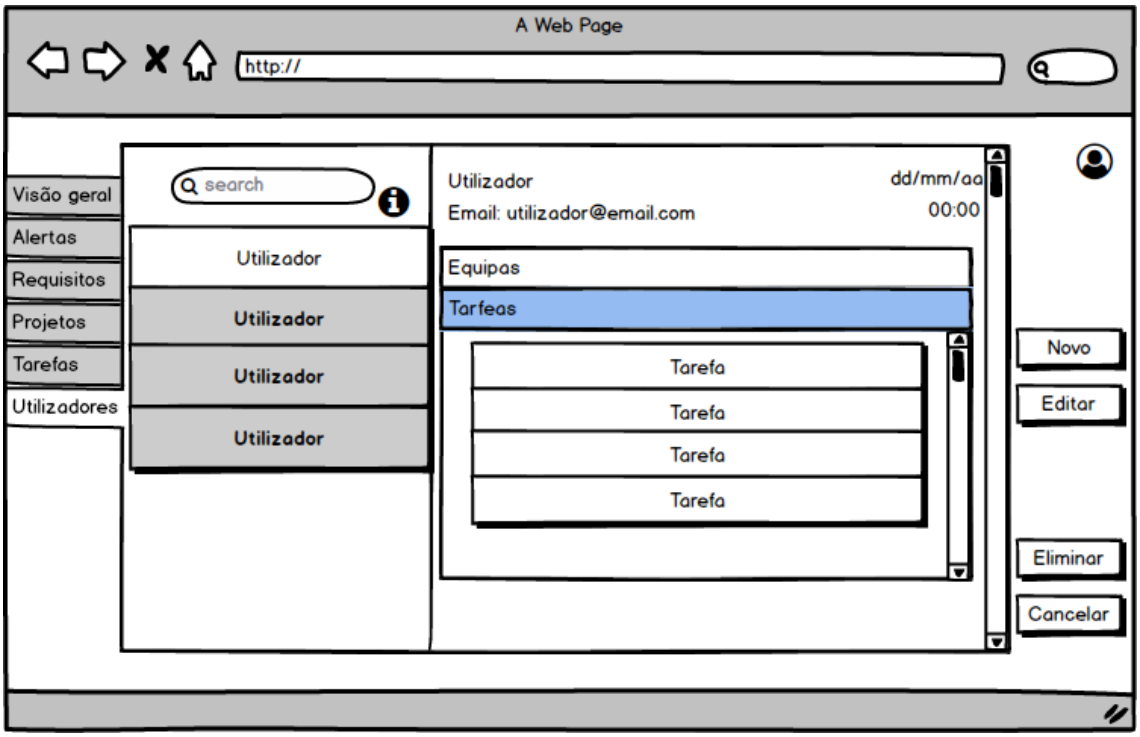


Figura 74: Protótipo da vista para ver os utilizadores.

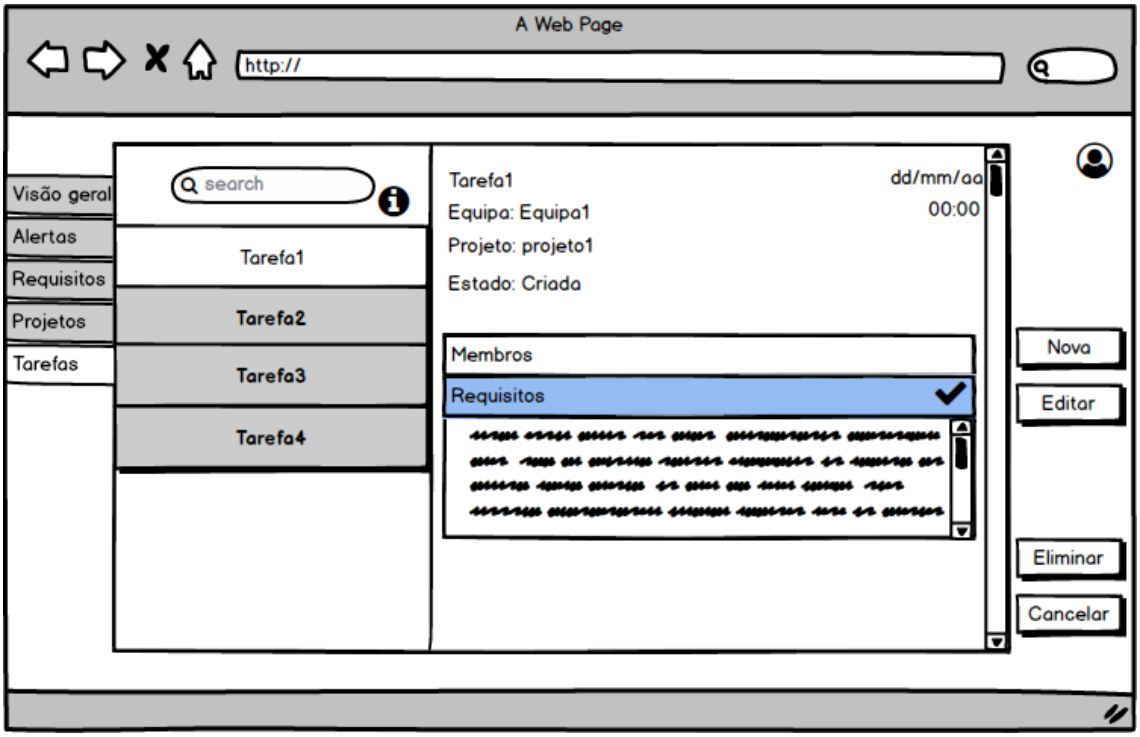


Figura 75: Protótipo para a gestão de tarefas.

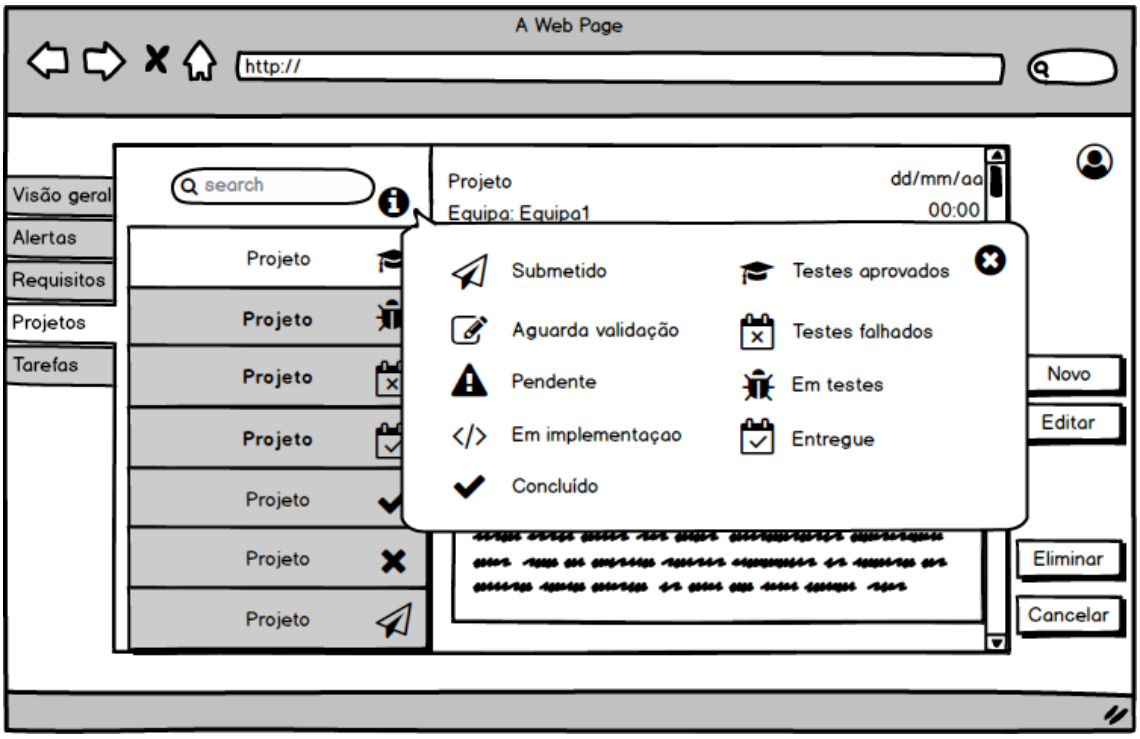


Figura 76: Protótipo para ver a legenda de ícones dos projetos.

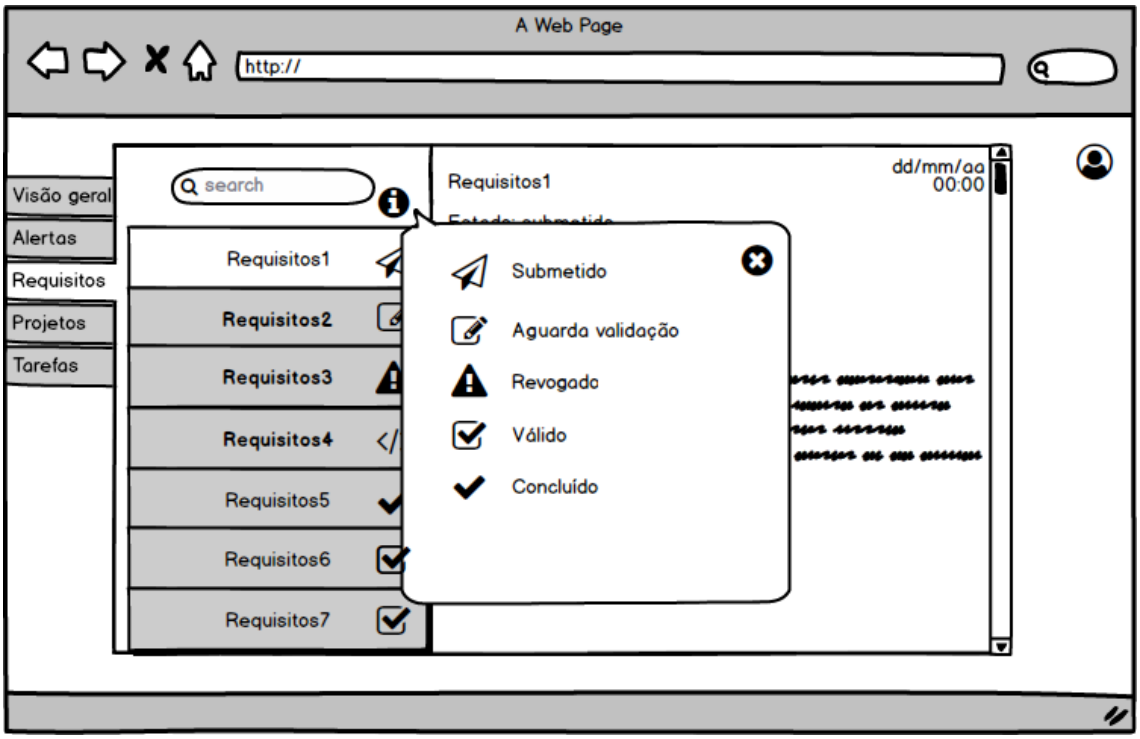


Figura 77: Protótipo para ver os ícones de ajuda dos requisitos.

Anexo E – Padrão arquitetural de produtores e consumidores

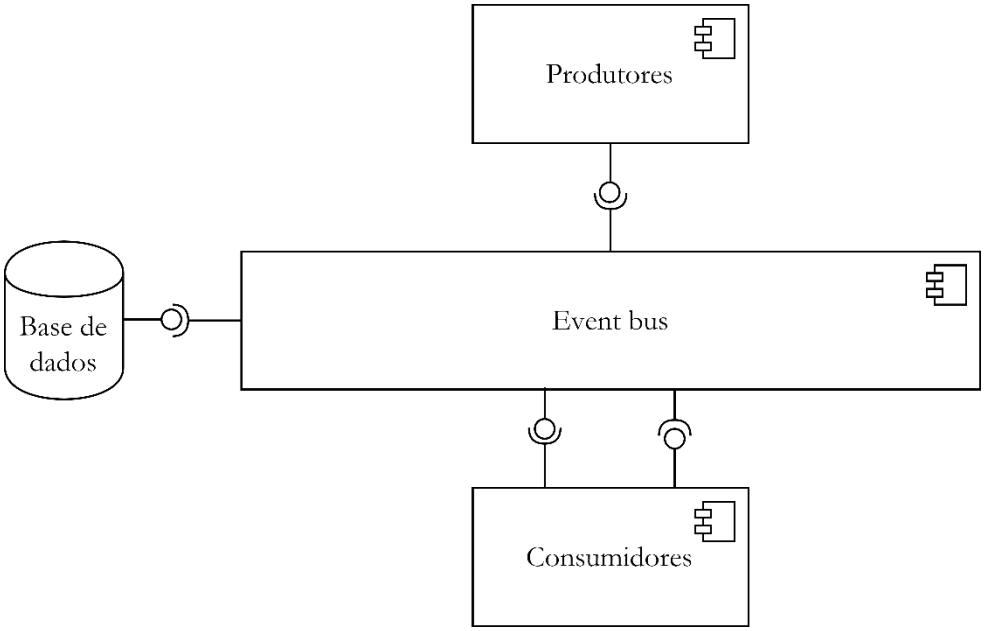


Figura 78: Padrão arquitetural de produtores e consumidores.

Anexo F – Vistas da ferramenta Odkas Factory

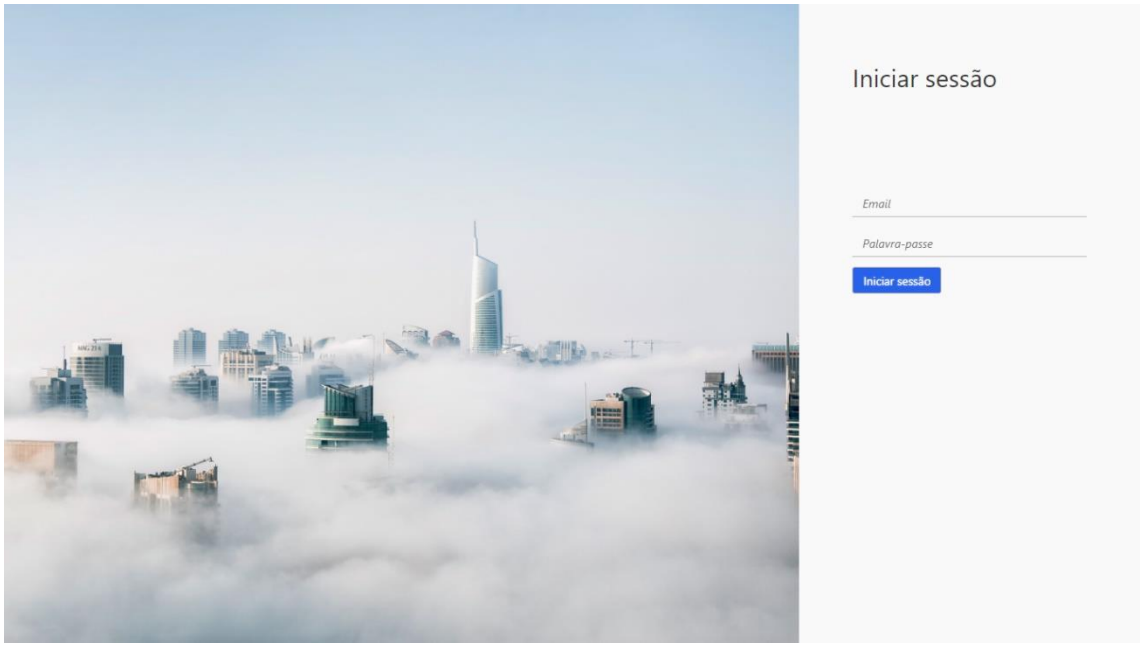


Figura 79: Vista de iniciar sessão.

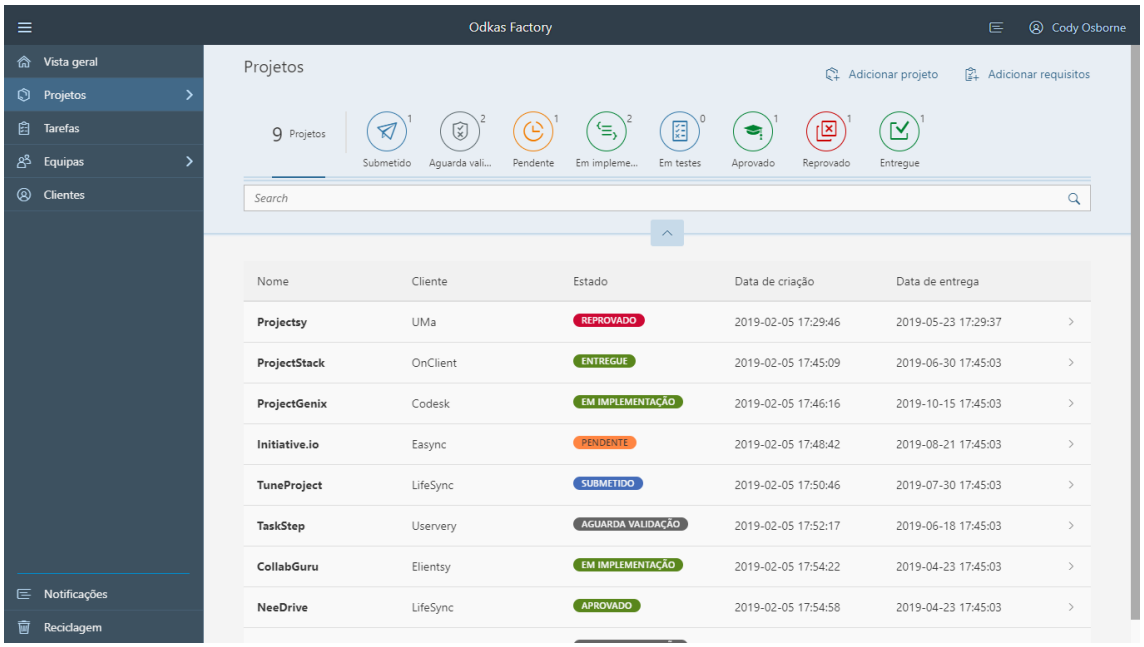


Figura 80: Vista para ver os projetos.

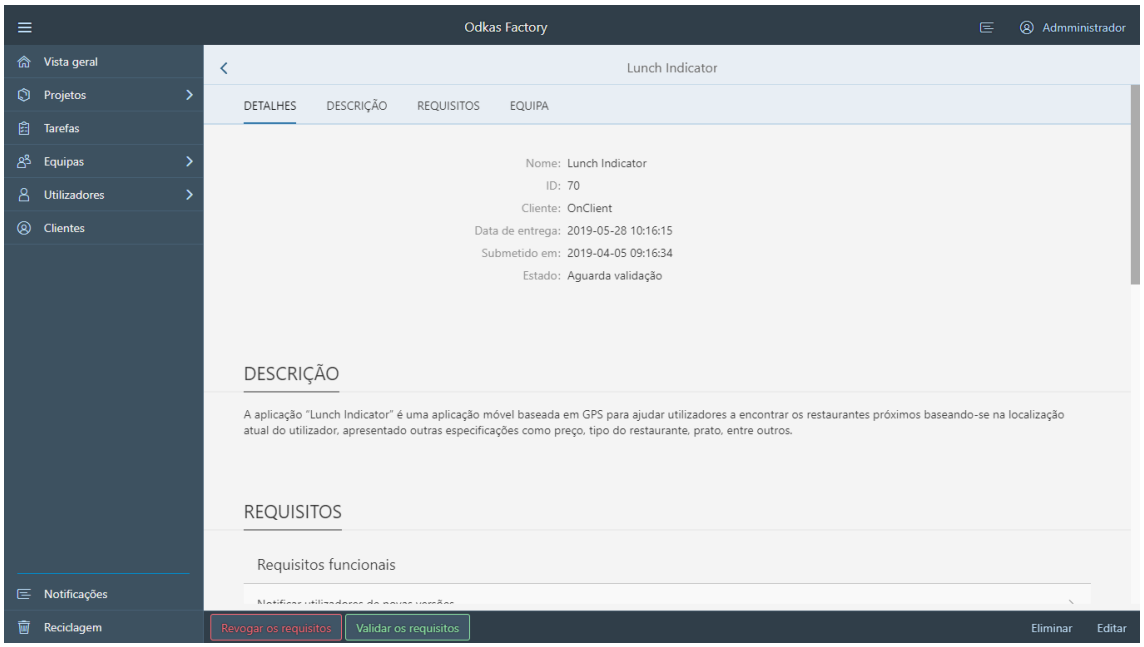


Figura 81: Parte superior da página de detalhes de um projeto.

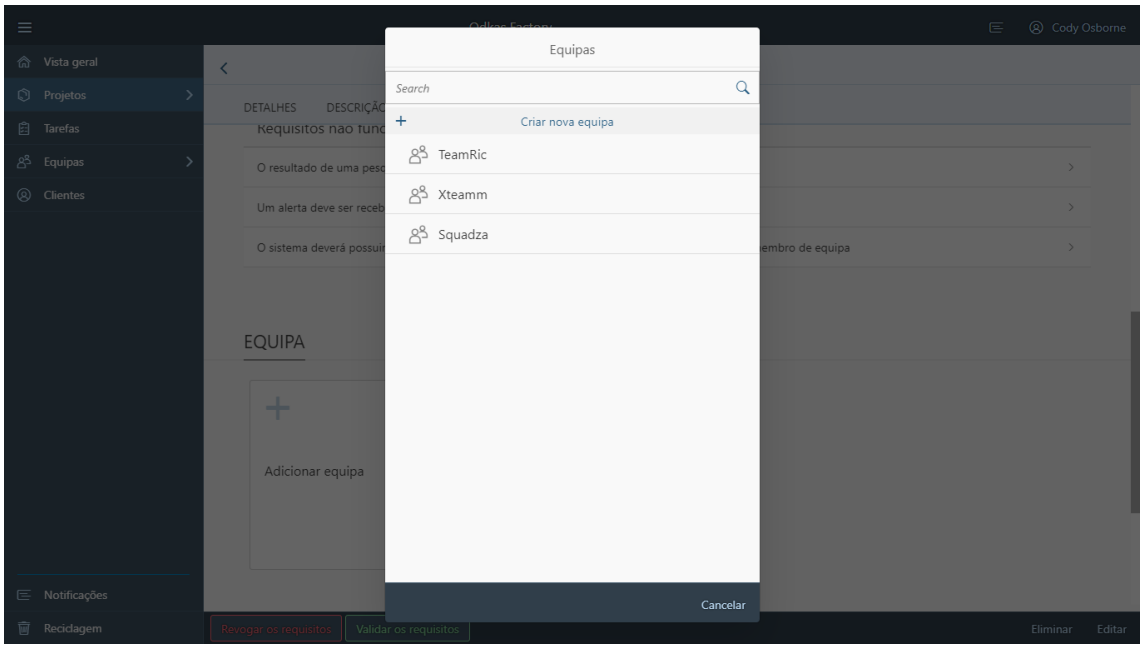


Figura 82: Parte inferior da página de detalhes de um projeto com a opção para associar uma equipa.

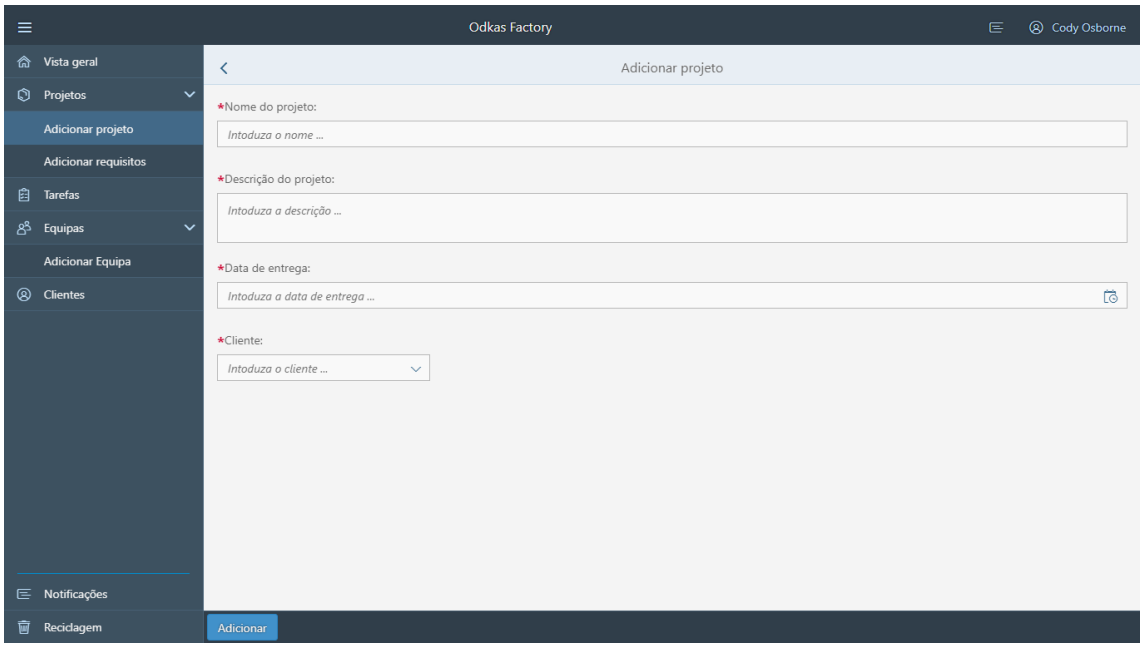


Figura 83: Vista para adicionar um projeto.

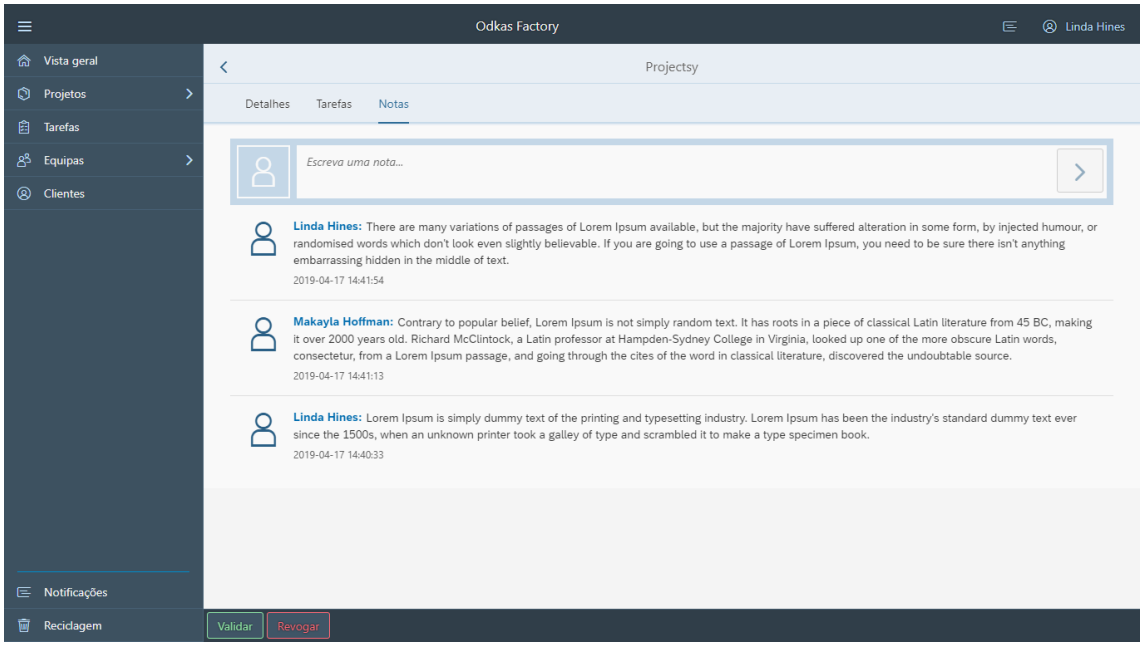


Figura 84: Vista para adicionar uma nota/comentário a um requisito.

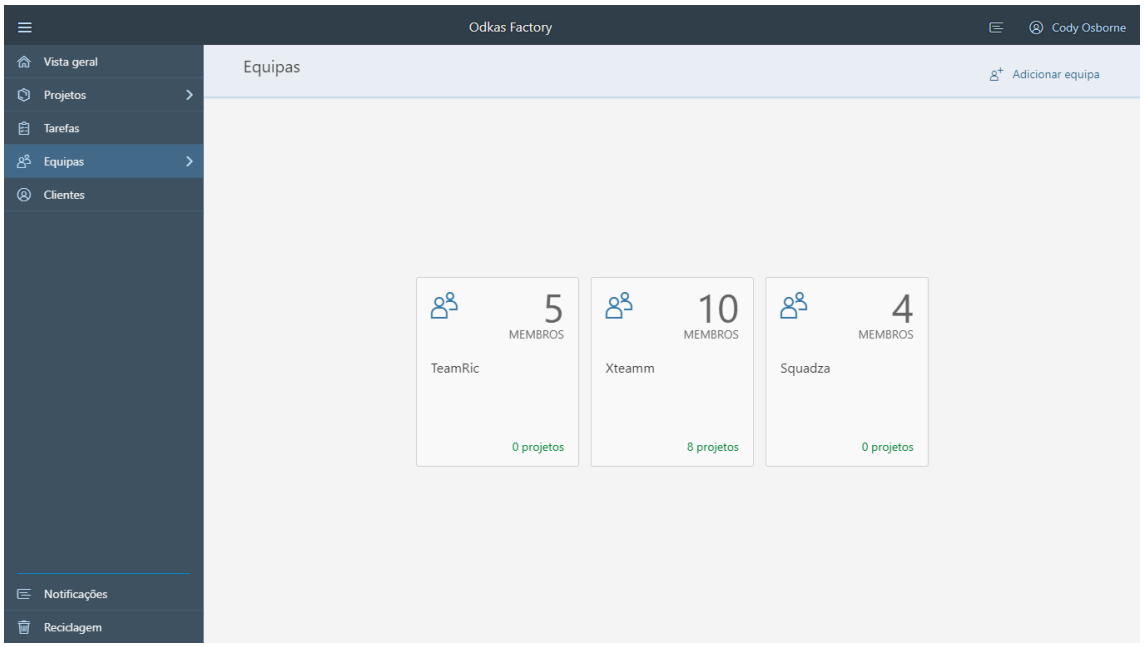


Figura 85: Vista para ver as equipas.

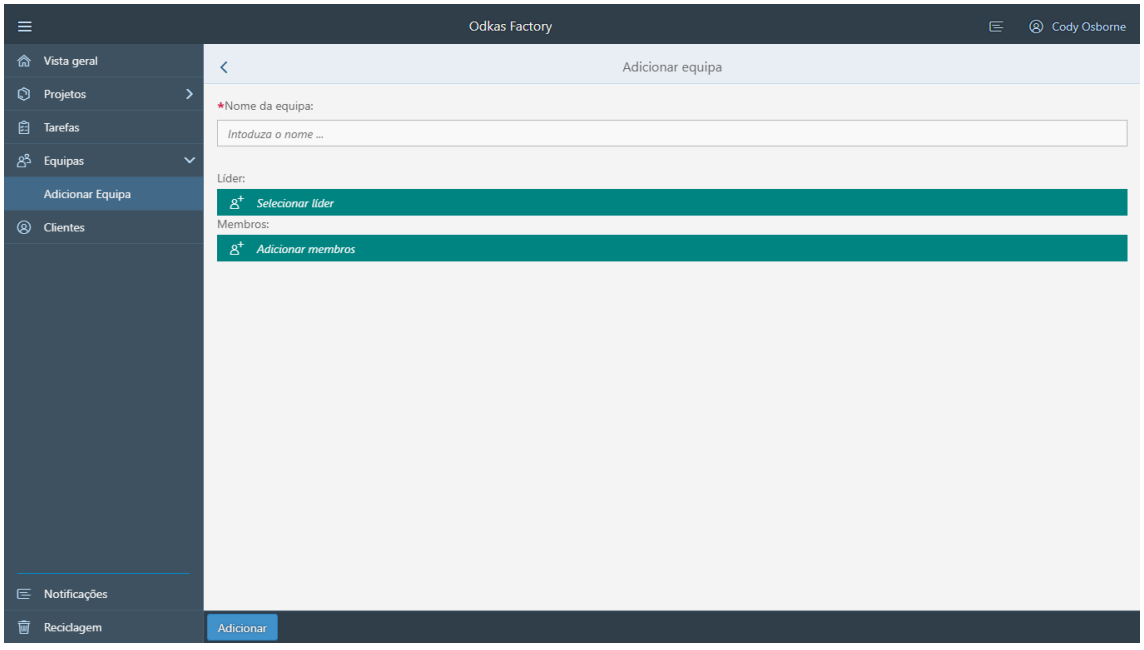


Figura 86: Vista para adicionar uma nova equipa.

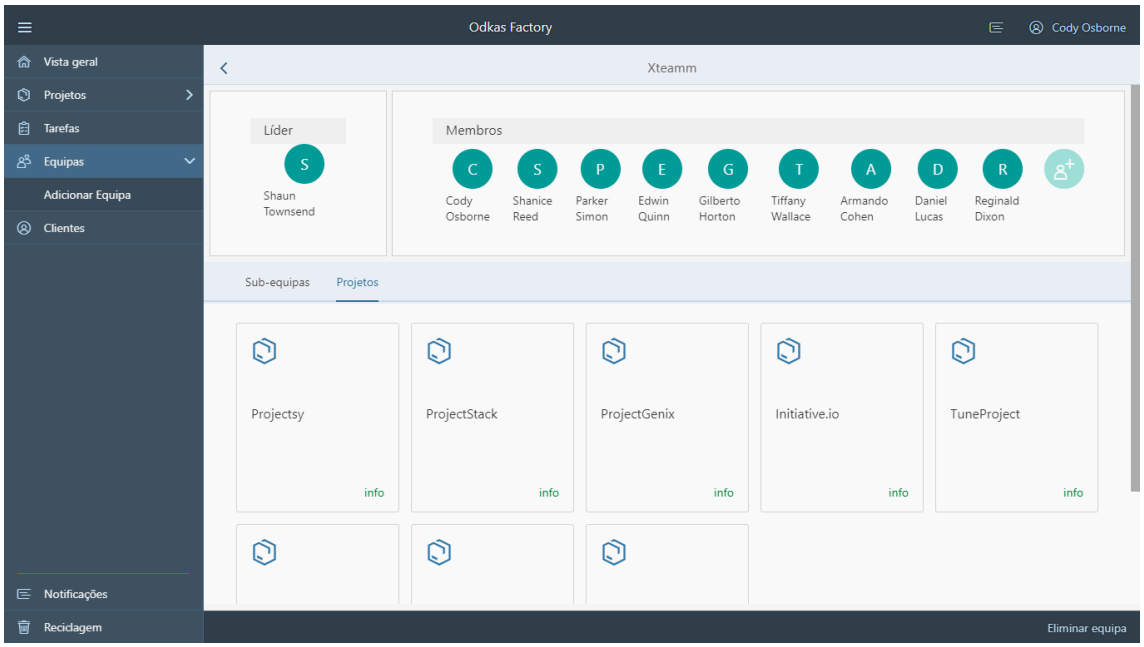


Figura 87: Vista para ver os detalhes de uma equipa.

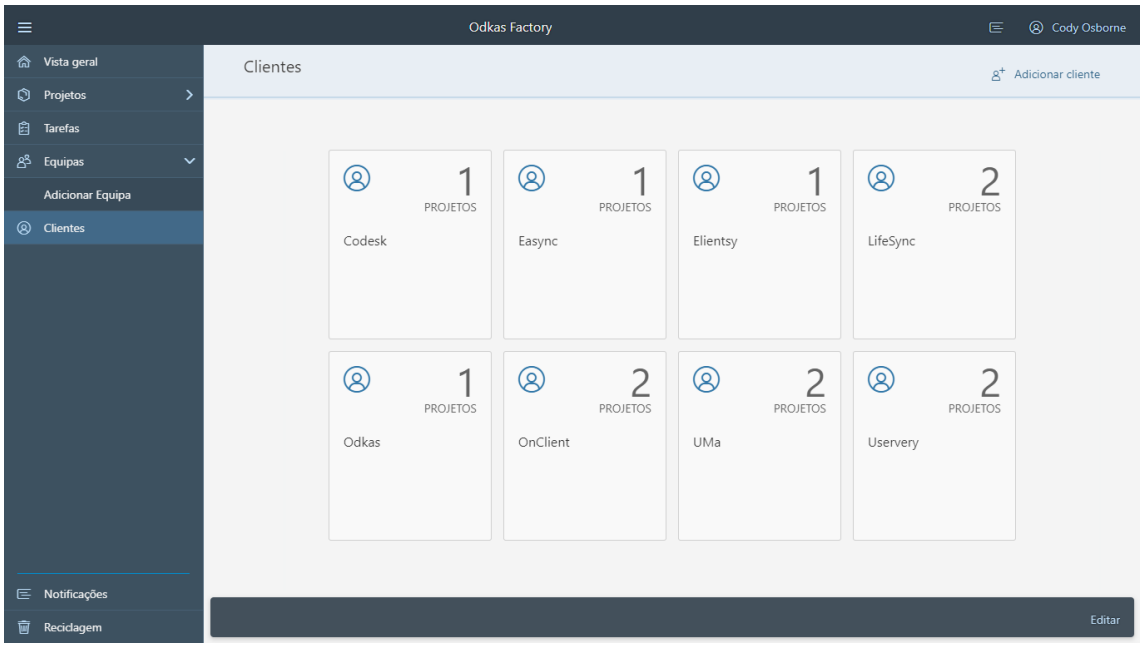


Figura 88: Vista para gerir clientes

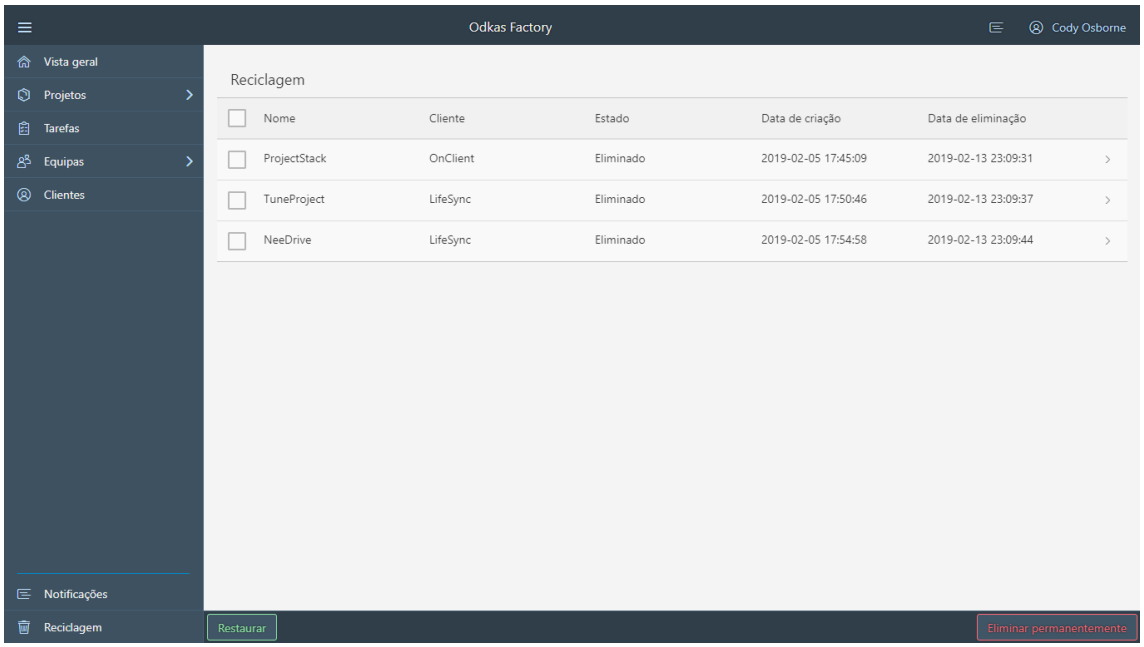


Figura 89: Vista da reciclagem.

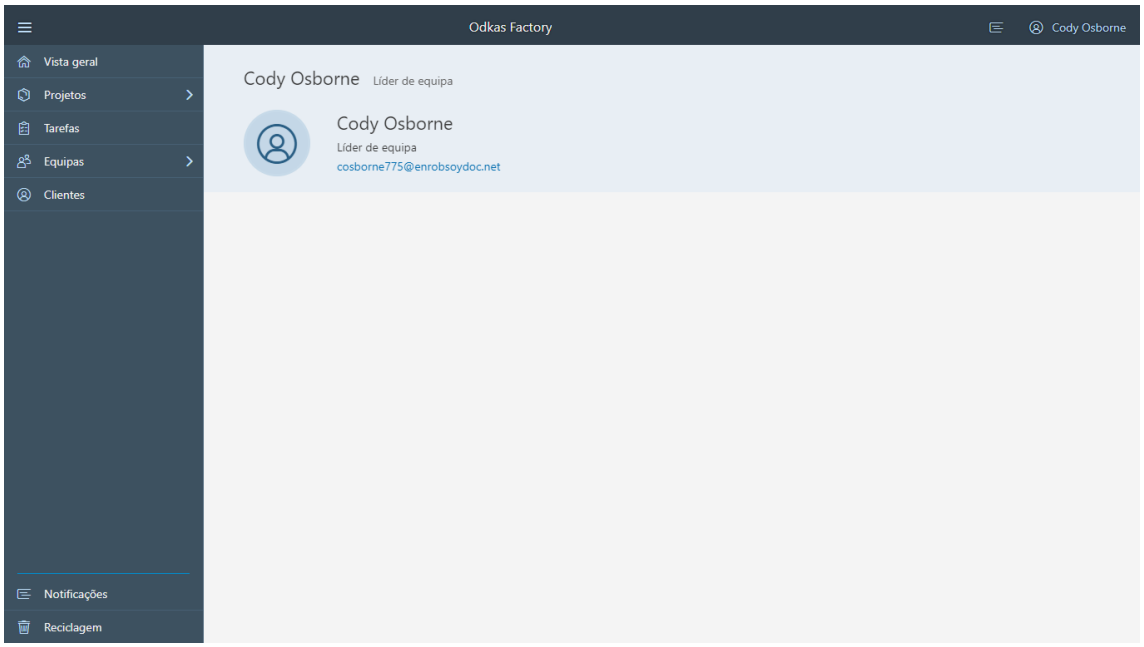


Figura 90: Vista da página pessoal.

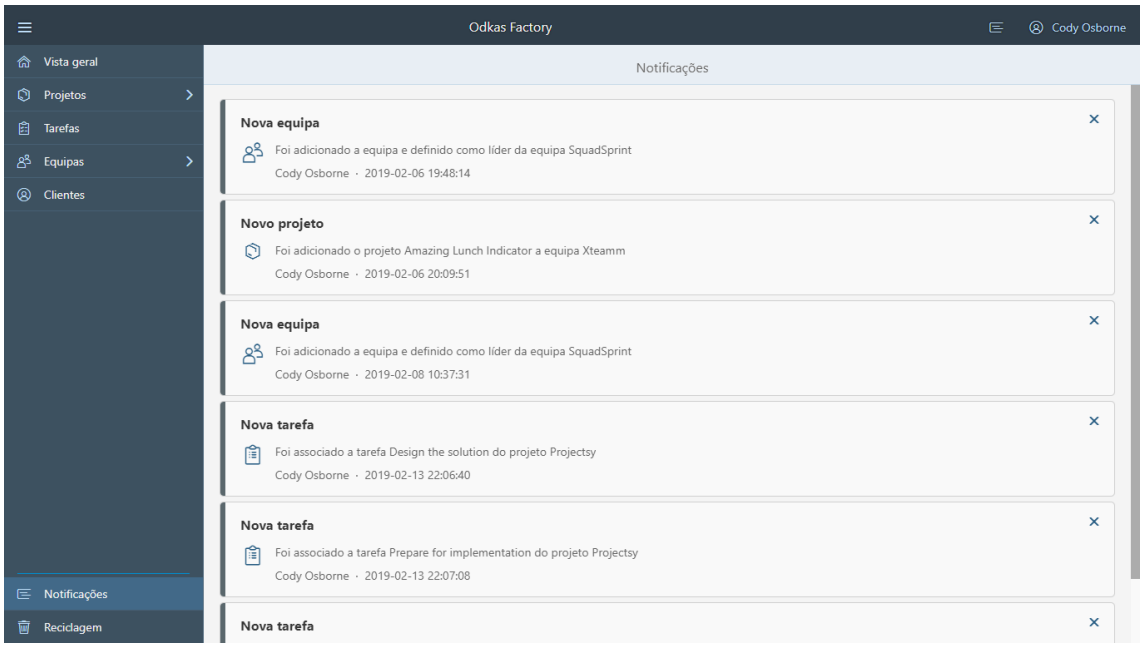


Figura 91: Vista das notificações.

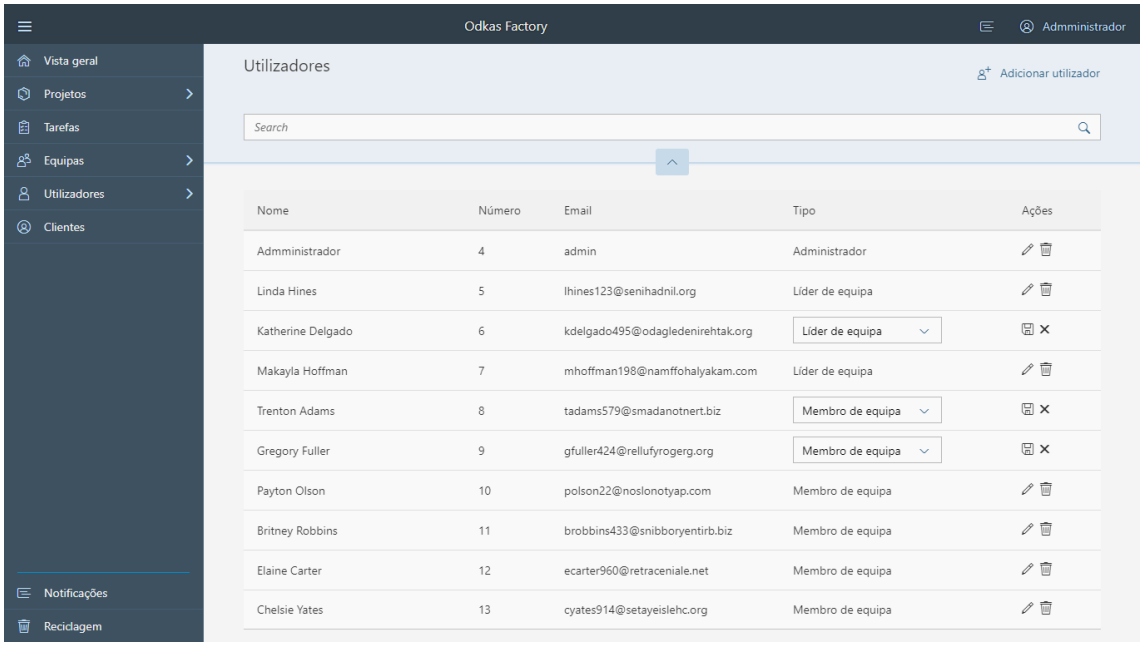


Figura 92: Vista para ver os utilizadores.

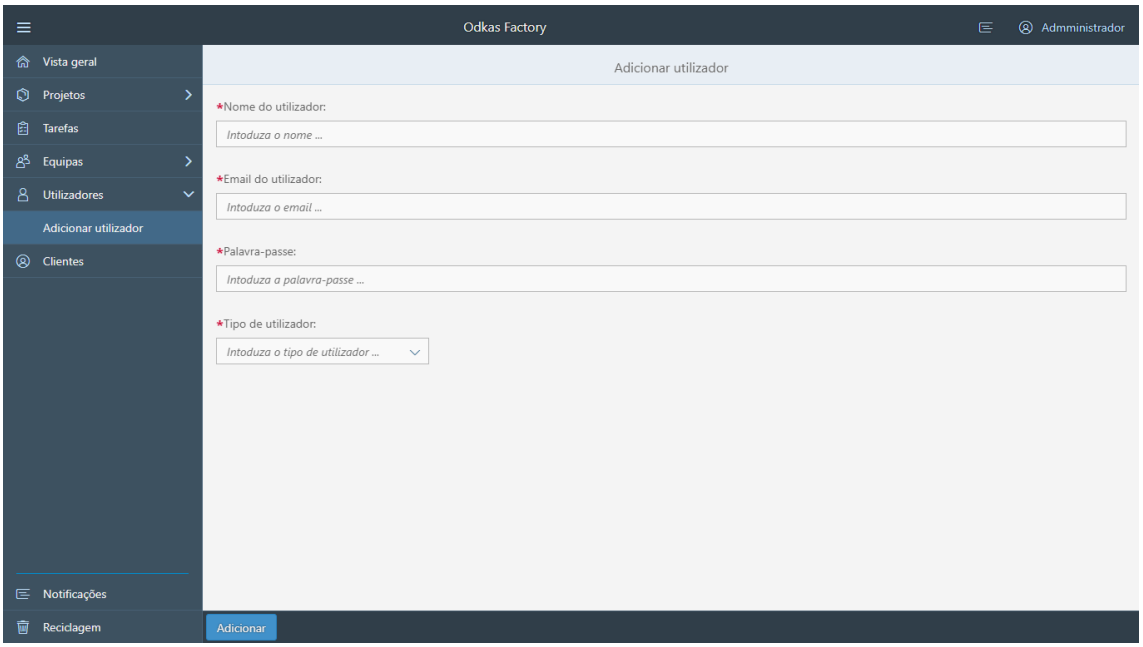


Figura 93: Vista para adicionar utilizadores.

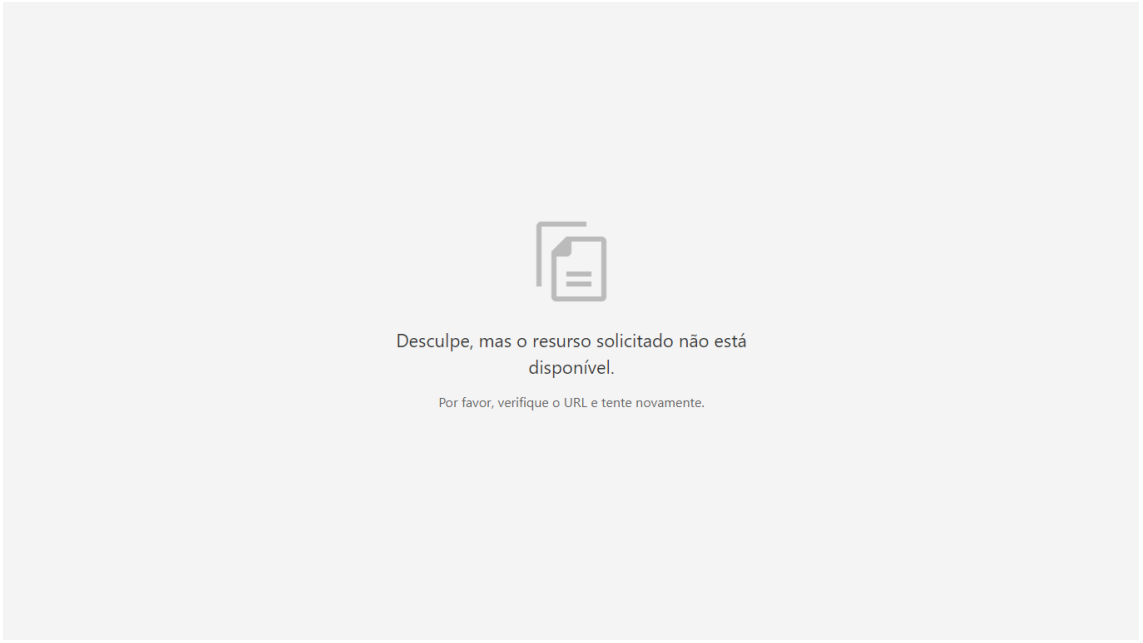


Figura 94: Vista de aviso de erro de rota inválida.

Anexo G – Teste de usabilidade

O teste seguinte desempenha as funções de líder de equipa:

1. Aceder a página https://factoryodkas.azurewebsites.net
2. Iniciar sessão com as credenciais:
    - Email: cosborne775@enrobsoydoc.net
    - Palavra-passe: cody

3. Criar Projeto:
    3.1. Adicionar o projeto "Lunch Indicator"[3] com as seguintes caraterísticas:
    - **Descrição:** A aplicação "Lunch Indicator" é uma aplicação móvel baseada em GPS para ajudar utilizadores a encontrar os restaurantes próximos baseando-se na localização atual do utilizador, apresentado outras especificações como preço, tipo do restaurante, prato, entre outros.
    - **Data de entrega:** 28/05/2019;
    - **Cliente:** Codesk;

    3.2. Adicionar ao projeto "Lunch Indicator" os requisitos funcionais e respetivas justificações:

Requisito funcional	Justificação
Notificar utilizadores de novas versões.	A fim de o utilizador poder descarregar as versões mais recentes da aplicação.
Recuperar palavra-passe.	
Possuir campos de procura.	Para que seja possível procurar por um determinado restaurante.
As mensagens entre utilizadores devem ser encriptadas.	

    3.3. Adicionar ao projeto "Lunch Indicator" os requisitos não funcionais e respetivas justificações:

Requisito não funcional	Justificação
O recurso de pesquisa deve ser proeminente e fácil de encontrar para o utilizador.	

[3] N. Sahlin, «Software Requirements Specification», p. 58.

A aplicação deve ser fácil de estender.	Para que seja fácil adicionar novas funcionalidades.

3.4. Mude o requisito "As mensagens entre utilizadores devem ser encriptadas" para a tabela de requisitos não funcionais.

3.5. Adicione os requisitos.

3.6. Valide todos os requisitos do projeto "Lunch Indicator";

3.7. Revogue o requisito "Recuperar palavra-passe.";

4. Criar Equipa:
   4.1. Crie uma equipa com o nome "SquadSprint";
   4.2. Adicione como líder o utilizador Cody Osborne;
   4.3. Adicione como membros os utilizadores Parker Simon, Tiffany Wallace e Daniel Lucas;
   4.4. Adicione a equipa;
5. Associar equipa a um projeto:
   5.1. Associe a equipa "SquadSprint" ao projeto "Lunch Indicator";
6. Adicionar tarefas
   6.1. Crie uma lista de tarefas "Por fazer".
   6.2. Na lista "Por fazer" adicione a tarefa "Criar perfil do utilizador";
   6.3. Adicione a seguinte descrição a tarefa criada: "Todos os utilizadores devem possuir a sua página de detalhes pessoais";
   6.4. Adicione o intervalo de datas: 20/02/2019 até 15/03/2019;
   6.5. Associe a tarefa aos requisitos "Recuperar a palavra passe" e "Possuir campos de procura";
   6.6. Remova o requisito "Possuir campos de procura." da tarefa;
   6.7. Atribua a tarefa aos membros Cody Osborne e Parker Simon;
   6.8. Visualize a notificação da nota tarefa;
   6.9. Visualize esta tarefa no planeamento;
   6.10. Identifique, na aplicação, onde se encontra a lista com as suas tarefas que estão no estado "Em execução".

Anexo H – Questionário e resultados

Ferramenta de gestão de projetos Odkas Factory

*Obrigatório

1. **Género:** *
   Marcar apenas uma oval.
   - ◯ Feminino
   - ◯ Masculino
   - ◯ Outra: _____

2. **Indique a sua faixa etária:** *
   Marcar apenas uma oval.
   - ◯ 18 ou menos
   - ◯ 19 a 29
   - ◯ 30 a 39
   - ◯ 40 a 49
   - ◯ 50 a 59
   - ◯ 60 ou mais

3. **Acho que gostaria de utilizar este produto com frequência:** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

4. **Considerei o produto mais complexo do que necessário:** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

5. **Achei o produto fácil de utilizar:** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

6. **Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto:** *
   Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

7. **Considerei que as várias funcionalidades deste produto estavam bem integradas:** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

8. **Achei que este produto tinha muitas inconsistências:** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

9. **Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto:** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

10. **Considerei o produto muito complicado de utilizar:** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

11. **Senti-me muito confiante a utilizar este produto:** *
Marcar apenas uma oval.

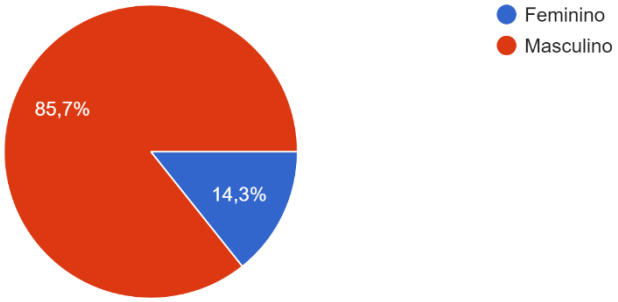
	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

12. **Tive que aprender muito antes de conseguir lidar com este produto:** *
Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo fortemente

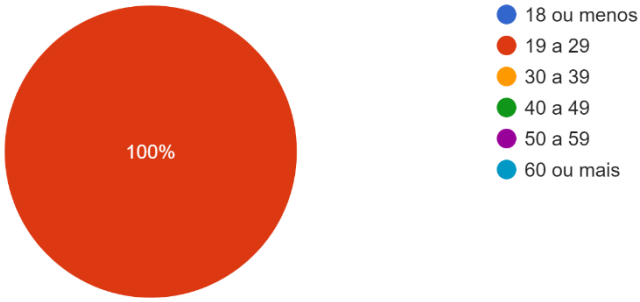
Género:

14 respostas



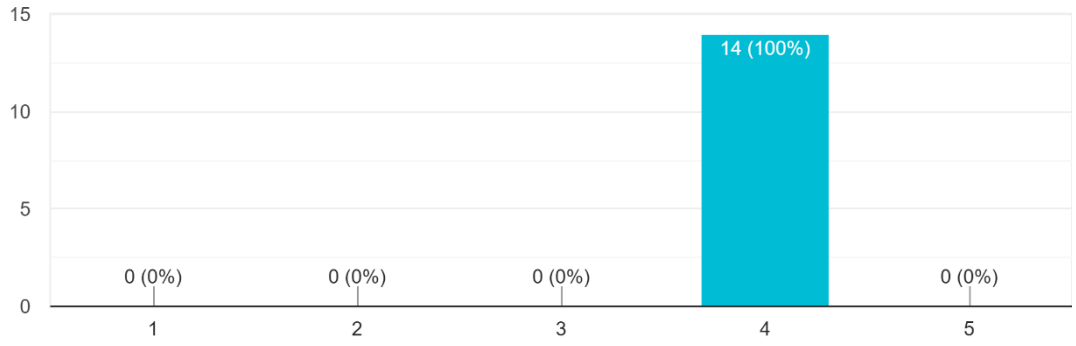
Indique a sua faixa etária:

14 respostas



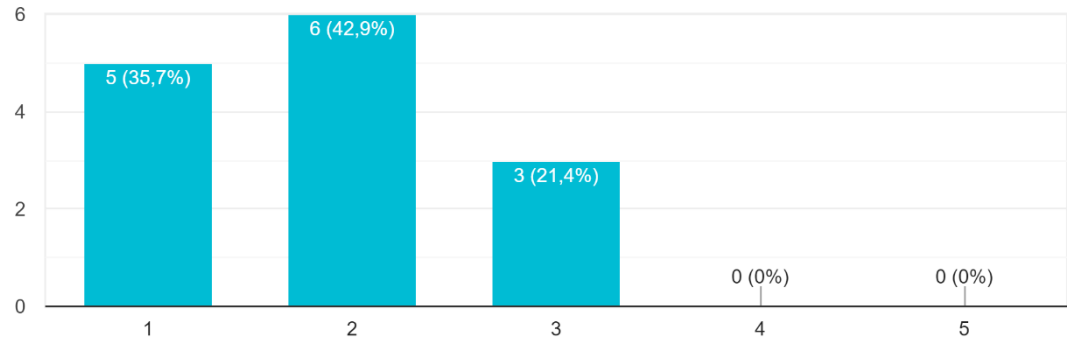
Acho que gostaria de utilizar este produto com frequência:

14 respostas



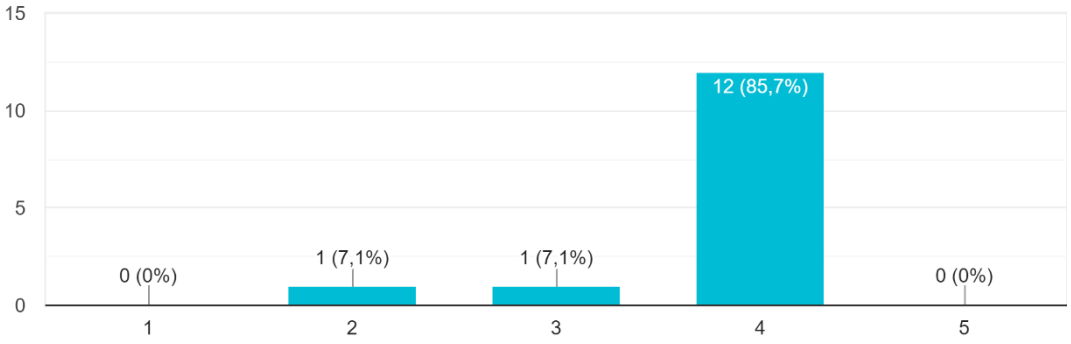
Considerei o produto mais complexo do que necessário:

14 respostas



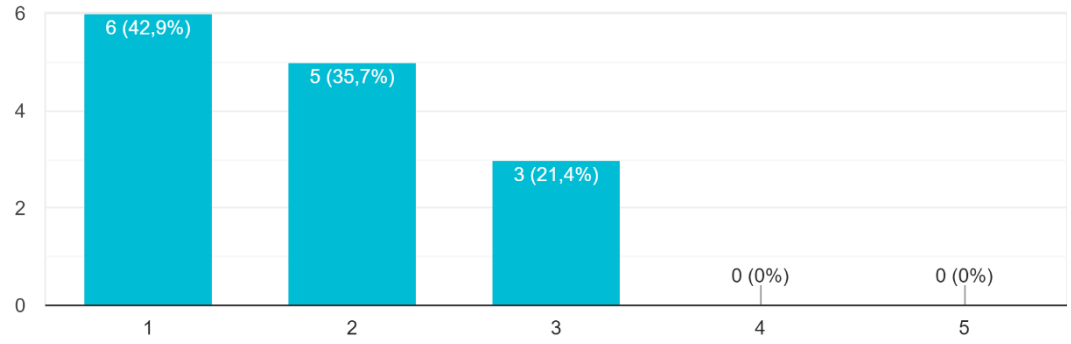
Achei o produto fácil de utilizar:

14 respostas



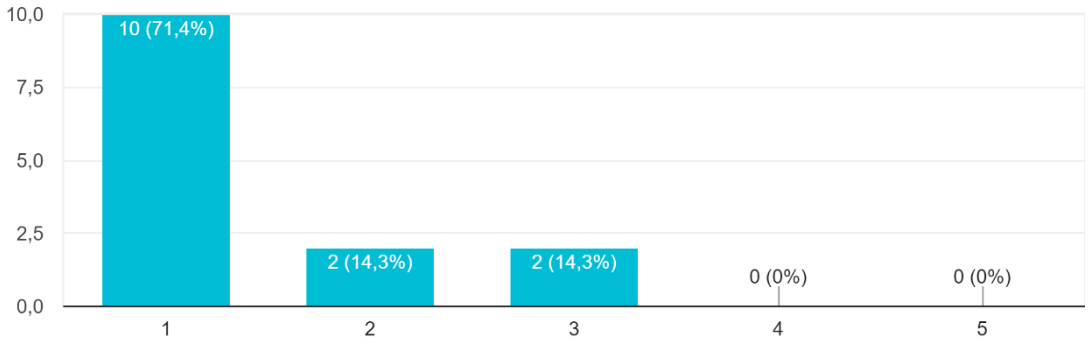
Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto:

14 respostas



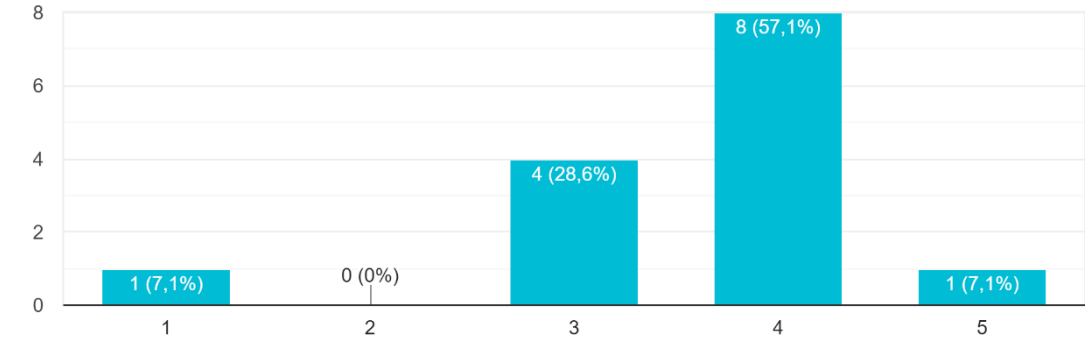
Considerei o produto muito complicado de utilizar:

14 respostas



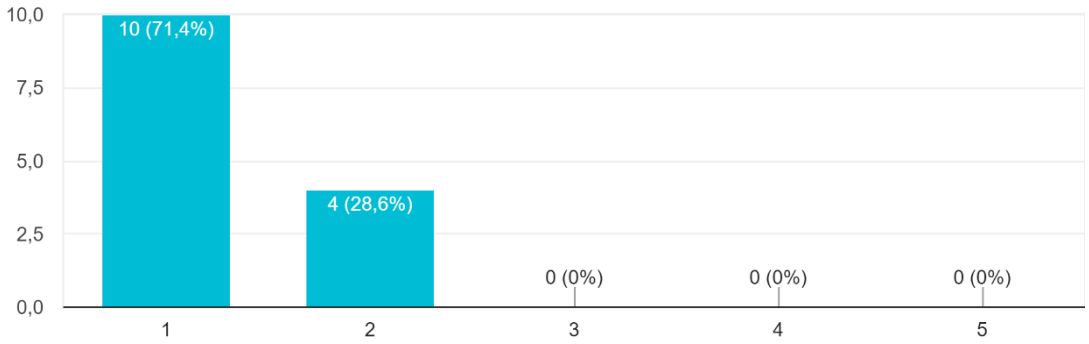
Senti-me muito confiante a utilizar este produto:

14 respostas



Tive que aprender muito antes de conseguir lidar com este produto:

14 respostas



Página em branco deixada propositadamente