



Aplicação de Gestão e Requisição de Equipamentos

Cesae Digital

2022 / 2023

1171060 André Silva

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

Aplicação de Gestão e Requisição de Equipamentos

Cesae Digital

2022 / 2023

1171060 André Silva



Licenciatura em Engenharia Informática

Setembro 2023

Orientador ISEP: **Diogo Martinho e Constantino Martins**

Supervisor Externo: **Helder Pinto**

À nossa

Agradecimentos

Agradeço à minha família por todo o apoio e suporte incondicional.

Aos meus amigos, pela compreensão durante o período deste projeto, em especial ao Pedro Martins pelos conselhos oferecidos.

Ao Professor Diogo Martinho, por se ter mostrado sempre disponível quando fosse necessário e pelo *feedback* dado.

Ao Professor Constantino Martins, por ter aceitado coorientar o projeto.

Por último, agradeço ao Helder Pinto por todo o apoio e *feedback* na realização do projeto e por garantir que o resultado obtido fosse sempre o melhor possível.

Resumo

Sistemas informáticos possuem uma função crucial nas organizações. Na organização Cesae Digital a requisição e gestão de equipamentos é realizada de forma manual. De modo a aumentar a eficiência das atividades e acelerar o processo de requisição e gestão de equipamentos é essencial criar um sistema informático que centralize a informação.

Neste relatório é descrito, em detalhe, o processo de desenvolvimento de uma aplicação *web* para a gestão e requisição de equipamentos. A aplicação possui funcionalidades como:

- Registo e login de utilizadores;
- Adição e gestão de equipamentos;
- Criação e gestão de requisições de equipamentos;
- Atualização automática dos stocks de equipamento disponível;
- Envio de notificações por parte do sistema.

A aplicação desenvolvida responde a todos os requisitos inicialmente propostos, com a adição de outras funcionalidades consideradas relevantes, oferecendo uma solução para a gestão de inventário e gestão e criação de requisições de equipamento do laboratório de multimédia do CESAE.

Palavras-chave (Tema): Gestão de requisições, Gestão de inventário, Cesae Digital

Palavras-chave (Tecnologias): Angular, .Net Core, SQL Server, C#, Typescript

Índice

1	Introdução	1
1.1	Enquadramento/Contexto.....	1
1.2	Descrição do problema.....	2
1.2.1	Objetivos	2
1.2.2	Abordagem	2
1.2.3	Contributos.....	3
1.2.4	Planeamento do trabalho.....	3
1.3	Estrutura do relatório	4
2	Estado da arte	5
2.1	Trabalhos relacionados.....	5
2.1.1	Sistemas de requisição de compras.....	5
2.1.1.1	Vantagens dos sistemas de requisição de compras	5
2.1.1.2	Desafios no uso de sistemas de requisição de compras	5
2.1.2	<i>Software</i> de gestão de requisições	6
2.1.2.1	Benefícios no uso de <i>software</i> de gestão de requisições.....	6
2.1.2.2	Exemplos <i>software</i> de gestão de requisições	6
2.1.3	Sistemas de gestão de inventário	9
2.1.3.1	Benefícios no uso de sistemas de gestão de inventário.....	9
2.2	Tecnologias existentes.....	10
2.2.1	.NET Core	10
2.2.2	Angular	10
2.2.3	SQL Server	11
2.2.4	.NET Core vs Node.js.....	11
2.2.5	Angular vs React	12
3	Análise e desenho da solução	14

3.1	Domínio do problema.....	14
3.2	Requisitos funcionais e não funcionais	15
3.2.1	Requisitos funcionais	15
3.2.2	Requisitos não funcionais	18
3.3	Desenho	20
3.3.1	Nível de sistema	21
3.3.2	Nível de contentores	21
3.3.3	Nível de componentes	22
3.3.4	Diagrama de classes	29
4	Implementação da Solução	30
4.1	Descrição da implementação	30
4.1.1	UC1 – Efetuar pedido de registo na aplicação	31
4.1.2	UC2 – Autenticar na aplicação	32
4.1.3	UC3 – Responder a pedido de registo	33
4.1.4	UC4 – Gerir utilizadores	35
4.1.5	UC5 – Criar categoria de equipamento.....	36
4.1.6	UC6 - Adicionar equipamento	38
4.1.7	UC7 - Listar equipamentos existentes	39
4.1.8	UC8 – Editar equipamento	40
4.1.9	UC9 - Eliminar equipamento.....	41
4.1.10	UC10 - Efetuar requisição de equipamentos	42
4.1.11	UC11 - Listar requisições de um utilizador.....	44
4.1.12	UC12 - Editar requisição	45
4.1.13	UC13 – Responder a pedido de requisição	46
4.1.14	UC15 - Listar requisições existentes	47
4.1.15	UC16 – Notificar utilizador de mudanças no estado da requisição.....	47

4.1.16	UC17/UC18 – Definir stock mínimo para categoria de equipamento e notificar em caso de stock reduzido	48
4.1.17	UC19 – Alterar estado da requisição	49
4.1.18	UC20 – Notificar atrasos na devolução da requisição.....	50
4.1.19	Bibliotecas e APIs relevantes	51
4.1.19.1	Angular Material	51
4.1.19.2	MailJet	51
4.2	Testes	51
4.2.1	Testes unitários	52
4.2.2	Testes de integração.....	54
4.2.3	Testes à API	56
4.3	Avaliação da solução	57
5	Conclusões	58
5.1	Objetivos concretizados	58
5.2	Limitações e trabalho futuro	58
5.3	Apreciação final	59
6	Referências.....	60

Índice de Figuras

Figura 1 - Diagrama de Gantt	4
Figura 2 - Pedido de requisição no Procurify (retirado de [13])	7
Figura 3 - Criar requisição no Tradogram (retirado de [15])	8
Figura 4 - Lista de requisições no Asset Infinity (retirado de [17])	9
Figura 5 - Web frameworks mais utilizadas [32]	13
Figura 6 - Modelo de domínio solução	14
Figura 7 - Diagrama de casos de uso	16
Figura 8 - Vista lógica (nível 1)	21
Figura 9 - Vista lógica (nível 2)	21
Figura 10 - Vista de implementação (nível 2)	22
Figura 11 - Vista lógica do Management (nível 3)	23
Figura 12 - UI vista lógica (nível 3)	24
Figura 13 - UC05 vista de processos front-end	25
Figura 14 - UC05 vista de processos back-end	26
Figura 15 - UC09 vista de processos UI	27
Figura 16 - UC09 vista de processos Management	28
Figura 17 - Diagrama de classes	29
Figura 18 - Menu da aplicação (função administrador)	30
Figura 19 - SSD UC1	31
Figura 20 - Página de registo versão mobile	32
Figura 21 - SSD UC2	32
Figura 22 - Página de login	33
Figura 23 - SSD UC3	33
Figura 24 - Página gestão de utilizadores	34
Figura 25 - Email de aceitação de registo	34

Figura 26 - SSD UC4 - delete user	35
Figura 27 - Página adicionar utilizador	35
Figura 28 - SSD UC5	36
Figura 29 - Página adicionar equipamento, selecionar categoria.....	36
Figura 30 - Página adicionar categoria de equipamento	37
Figura 31 - SSD UC6	38
Figura 32 - Adicionar equipamento (dados equipamento).....	38
Figura 33 - Adicionar equipamento (confirmação).....	39
Figura 34 - SSD UC7	39
Figura 35 - Página gerir equipamentos.....	40
Figura 36 - SSD UC8.....	40
Figura 37 - Página com informação de um equipamento.....	41
Figura 38 - SSD UC9	41
Figura 39 - Remover equipamento.....	42
Figura 40 - SSD UC10	42
Figura 41 - Página criar requisição (escolher equipamentos).....	43
Figura 42 - Página criar requisição (escolha de datas).....	44
Figura 43 - SSD UC11	44
Figura 44 - Página que mostra as requisições do utilizador com sessão iniciada	45
Figura 45 - SSD UC12	45
Figura 46 - SSD UC13	46
Figura 47 - Página gerir requisições.....	46
Figura 48 - Página gerir requisições (requisições canceladas)	47
Figura 49 - SSD UC15	47
Figura 50 - SSD UC16	47
Figura 51 - Email requisição cancelada.....	48
Figura 52 - SSD UC17	48

Figura 53 - Página detalhes de categoria de equipamento	48
Figura 54 - Email de alerta de stock reduzido.....	49
Figura 55 - SSD UC19	49
Figura 56 - Vista lógica nível 3 UC20.....	50
Figura 57 - Método para verificar atrasos nas requisições	50
Figura 58 - Método para enviar emails usando MailJet.....	51
Figura 59 - Teste unitário front-end	52
Figura 60 - Teste unitário back-end.....	53
Figura 61 - Teste de integração back-end	55
Figura 62 - Testes à API para criar requisição.....	56
Figura 63 - Testes no Postman para editar requisição.....	57

Índice de Tabelas

Tabela 1 - Comparação entre as características do .NET Core e Node.js	12
Tabela 2 - Comparação entre características do Angular e React	12
Tabela 3 - Testes unitários para a criação de uma requisição	53
Tabela 4 - Testes de integração para criação de uma requisição	55

Notação e Glossário

Angular	<i>Framework para o desenvolvimento de front-end</i>
API	<i>Application Programming Interface</i>
Back-end	Componente da aplicação relativa à estrutura e lógica de negócio
C#	Linguagem de programação
CESAE	Organização para o qual o projeto está a ser realizado
DOM	<i>Document Object Model</i>
Framework	Estrutura base da qual é possível construir <i>software</i>
Front-end	Componente da aplicação relativa à UI
HTTP	<i>Hypertext Transfer Protocol</i>
JavaScript	Linguagem de programação
.NET Core	<i>Framework para o desenvolvimento de back-end</i>
React	<i>Framework para o desenvolvimento de front-end</i>
SPA	<i>Single Page Application</i>
TypeScript	Linguagem de programação
UI	<i>User Interface</i>

1 Introdução

A introdução encontra-se dividida em três secções. Começa com uma perspetiva geral do problema em estudo com uma secção de enquadramento do problema e é descrita a motivação para a aceitação do desafio. Segue-se a descrição do problema, onde estão definidos os objetivos do projeto, a abordagem utilizada para solucionar o problema, os contributos que o projeto traz para a empresa e o planeamento do trabalho. Por último, é descrita a estrutura do relatório.

1.1 Enquadramento/Contexto

Sistemas informáticos possuem uma função crucial nas organizações. Eles ajudam organizações na tomada de decisões ao fornecer a informação necessária através do processamento dos dados fornecidos. Sistemas informáticos oferecem informação mais completa, rápida e organizada, permitindo às organizações operarem de maneira mais eficaz e reduzir os custos. [1]

Um uso importante de sistemas informáticos é a gestão de inventário. Manter informação correta e atualizada do stock existente de uma organização é essencial. Sistemas de gestão de inventário permitem às organizações monitorizar facilmente o stock disponível e a localização e estado dos seus produtos. [2], [3]

Também é possível usar sistemas informáticos para facilitar o processo de requisição de equipamento. Pedidos de requisição podem ser realizados e aprovados digitalmente, o seu estado e fluxo de aprovação gerido e seguido pelo sistema e, com isso, garantir o acesso a informação correta e atualizada. [4], [5]

A criação de um sistema informático para a gestão e requisição de equipamentos proporciona uma excelente oportunidade para tornar as atividades do laboratório de multimédia do CESAE mais eficientes onde, no momento de escrita do relatório, o processo de requisição e gestão de equipamento é realizado manualmente.

A motivação pessoal para a escolha e aceitação deste projeto deve-se ao facto de, com este projeto, deixa de ser necessária a gestão manual da informação dos equipamentos de laboratório do CESAE. Com a automatização do sistema, o processo de requisição de equipamentos e gestão de informação deverá tornar-se mais fácil e rápido para todos os intervenientes, trazendo, deste modo, valor para a organização.

1.2 Descrição do problema

Gerir equipamentos e requisições de equipamento de forma manual é um processo demorado e pouco eficiente. Isto pode levar a registos de inventário incorretos devido a erro humano e dificuldades em aceder à informação. Adotar um sistema informático para gerir requisições e equipamentos pode resolver estes problemas e trazer vários benefícios. [6]

Através do uso de uma aplicação tecnológica é possível acelerar o processo descrito anteriormente pois, ao reduzir o número de tarefas manuais, os registos de inventário tornam-se mais credíveis com a diminuição do erro humano, passa também a ser possível notificar em tempo real os utilizadores através de alertas relevantes, o acesso à informação fica rápido e fácil e, ao acelerar todo o processo de gestão e requisição de equipamentos, aumenta a eficiência e reduz custos da organização. [4]

1.2.1 Objetivos

O principal objetivo deste projeto passa por automatizar o processo de requisição e gestão de equipamentos do laboratório de multimédia da empresa Cesae Digital, resolvendo, com isso, os problemas descritos na secção anterior (1.2). Foi então necessário a criação de uma aplicação para permitir aos técnicos gerir os equipamentos existentes e a criação de requisições de equipamento por parte dos vários utilizadores, assim como o acompanhamento dessas requisições. A aplicação *web* pretende atingir estas metas com funcionalidades como:

- Registo e login de utilizadores com diversas funções, funcionalidade fundamental para garantir a autenticidade dos utilizadores;
- Possibilidade de adicionar equipamentos e gerir equipamentos para, com isto, poderem ser requisitados pelos utilizadores e ser mantido um controlo de inventário.
- Efetuar e gerir requisições, permitindo aos utilizadores fazer uso dos equipamentos do laboratório de multimédia da organização.
- Atualização automática de stock de equipamentos disponível, certificando-se que todos os equipamentos são contabilizados;
- Envio de notificações por parte do sistema, alertando os utilizadores e técnicos de informação relevante com as suas requisições e equipamentos.

1.2.2 Abordagem

No seguimento da análise do problema em questão e contexto do projeto, a abordagem adotada focou-se no desenvolvimento de uma aplicação *web* através do uso de tecnologias como .Net Core e Angular. A aplicação possui como principais atores o administrador, os

técnicos e os diversos utilizadores pertencentes à organização do CESAE e funcionalidades que permitem a criação de requisições e a gestão de equipamentos.

A aplicação encontra-se dividida em duas componentes. A componente *front-end* onde se encontra a *interface* do utilizador e onde o utilizador comunica com o sistema, desenvolvida em Typescript. A componente *back-end* foi desenvolvida usando a linguagem C# e tem como principal função fazer a ligação do utilizador aos elementos guardados na base de dados.

1.2.3 Contributos

O contributo principal desta aplicação é facilitar e tornar mais eficiente o processo de gestão e requisição de equipamentos da organização CESAE. Ao passar de um processo manual para um processo digital, com componentes automatizados, todo o processo e arquivo de requisições e equipamento encontra-se centralizado na aplicação, trazendo então valor para a organização. A *web app* facilita também a comunicação entre os técnicos e os utilizadores ao tornar a informação das requisições mais acessível.

Este projeto trata-se de uma solução personalizada para uso interno da organização, contribuindo para o melhoramento das suas atividades diárias relacionadas com o seu laboratório de multimédia.

1.2.4 Planeamento do trabalho

O projeto foi inicialmente planeado após a definição concreta dos requisitos para a aplicação com a definição das principais tarefas e datas propostas para a sua realização.

A reunião de sessão de esclarecimento dos requisitos da aplicação foi realizada em 29 de março e foi criado um planeamento para a realização do projeto. Esse planeamento foi depois alterado na reunião de “Ponto de Situação”.

A versão final do planeamento inicial definia datas para as seguintes fases do projeto:

- Estudo do estado da arte: 21 abril – 10 maio;
- Planeamento e design da solução: 24 abril – 27 maio;
- Implementação da solução: 28 maio – 3 agosto;
- Testes e validação: 28 maio – 10 agosto;
- Escrita de Relatório: 21 abril – 24 agosto;
- Revisão do Relatório: 26 agosto - 7 setembro.

A figura 1 demonstra o diagrama de Gantt para o planeamento proposto.



Figura 1 - Diagrama de Gantt

Este projeto teve um acompanhamento regular do Supervisor da empresa através de reuniões recorrentes.

Foi usada a plataforma GitHub para a gestão dos avanços do projeto e controlo de versões.

1.3 Estrutura do relatório

Para além da introdução, este relatório contém mais 4 capítulos:

- Capítulo 2 - Estado da arte: é feito um estudo de sistemas e trabalhos relacionados com o tema deste projeto e são analisadas as tecnologias utilizadas no projeto e possíveis alternativas;
- Capítulo 3 – Análise e desenho da solução: é feita uma análise ao domínio do problema, são identificados os requisitos funcionais e não funcionais e é elaborado o desenho da arquitetura da aplicação;
- Capítulo 4 – Implementação da Solução: é descrita a implementação da solução preconizada no capítulo anterior e é feita a avaliação da mesma. São também descritos os testes efetuados;
- Capítulo 5 – Conclusão: é apresentado um resumo dos resultados obtidos em função dos objetivos inicialmente definidos, é feita uma análise às limitações e possível desenvolvimento futuro do projeto e é realizada uma apreciação final do trabalho.

2 Estado da arte

Neste capítulo é feito um estudo literário de sistemas e tecnologias relacionadas com a requisição de produtos e compras. De seguida, é feita uma revisão e documentação das tecnologias utilizadas no desenvolvimento do projeto, assim como algumas alternativas.

2.1 Trabalhos relacionados

Tratando-se o projeto em questão de uma plataforma para requisição de equipamentos de laboratório, foi estudado em que consistem os sistemas e protocolos existentes para a gestão e requisição de equipamentos e inventário.

2.1.1 Sistemas de requisição de compras

Um sistema de requisição de compras é um processo usado por organizações para solicitar, aprovar e obter bens e serviços. Quando um departamento necessita de novos equipamentos, é criada uma requisição de compra – um documento formal que captura os detalhes dos equipamentos que é necessário adquirir. Esta requisição é depois redirecionada, através dos canais apropriados, para as entidades responsáveis poderem aprovar a requisição. Depois de aprovada, torna-se numa ordem de compra que pode ser enviada aos fornecedores. [7]

2.1.1.1 Vantagens dos sistemas de requisição de compras

Usar um sistema de requisição de compras traz vários benefícios tais como:

- Cria transparência nos procedimentos de compra com a elaboração de um registo claro para esse efeito [8];
- Os atores interessados têm acesso à informação a respeito da requisição (produtos requisitados, quem aprovou, quem requisitou, etc.) [9];
- Ajuda a centralizar o processo de compra, aumentando o nível de eficiência da organização [10];
- Estabelecimento de níveis hierárquicos de aprovação, um sistema de requisição de compras bem definido incorpora níveis hierárquicos de aprovação pré-definidos baseados na estrutura e políticas organizacionais [9].

2.1.1.2 Desafios no uso de sistemas de requisição de compras

Para além das vantagens que oferece, este sistema também pode trazer alguns problemas. Existe a possibilidade de ser registada informação incompleta ou imprecisa,

resistência à mudança dos procedimentos da organização pelos colaboradores e falta de clareza e transparência do estado de um pedido de requisição ao longo do seu ciclo de vida. Sem mecanismos de monitorização adequados pode tornar-se difícil para o requisitante saber o estado do seu pedido. [11], [12]

2.1.2 Software de gestão de requisições

Um *software* de gestão de requisições vem responder a vários desafios presentes nos sistemas manuais de requisição de compras. Trata-se de uma ferramenta para automatizar o processo de requisição de equipamentos e produtos, desde a criação da requisição à aprovação e ordem de compra. Torna-se, assim, mais fácil, criar, processar, autorizar e acompanhar requisições dentro de uma organização, oferecendo uma maior visibilidade e controlo de todo o processo. [4], [5]

Com este software, os utilizadores podem criar digitalmente requisições detalhadas, os fluxos de aprovação são facilmente customizados, é possível um acompanhamento em tempo real das requisições e é possível gerir as permissões dos utilizadores com base na sua função [4], [5].

2.1.2.1 Benefícios no uso de *software* de gestão de requisições

Esta automatização traz benefícios significantes, tais como:

- Automatiza o processo de requisição, reduzindo, assim, o tempo e esforço necessários para criar e aprovar requisições [4];
- Acompanha e atualiza o estado das requisições, oferecendo, em tempo real, atualizações no seu progresso [4];
- Fornece a documentação completa de todas as requisições guardadas no sistema, tornando mais fácil visualizar o histórico de requisições efetuadas [4];
- Permite customizar e automatizar os fluxos de aprovação, diminuindo a necessidade de intervenção manual [5].

2.1.2.2 Exemplos *software* de gestão de requisições

Existem vários programas com funcionalidades para a gestão de requisições presentes no mercado. Procurify, Tradogram, e Asset Infinity são alguns dos programas que atuam nesta área. Todos eles oferecem a possibilidade de criar uma requisição selecionando produtos presentes numa lista/catálogo e aceitar ou recusar a requisição.

2.1.2.2.1 Procurify

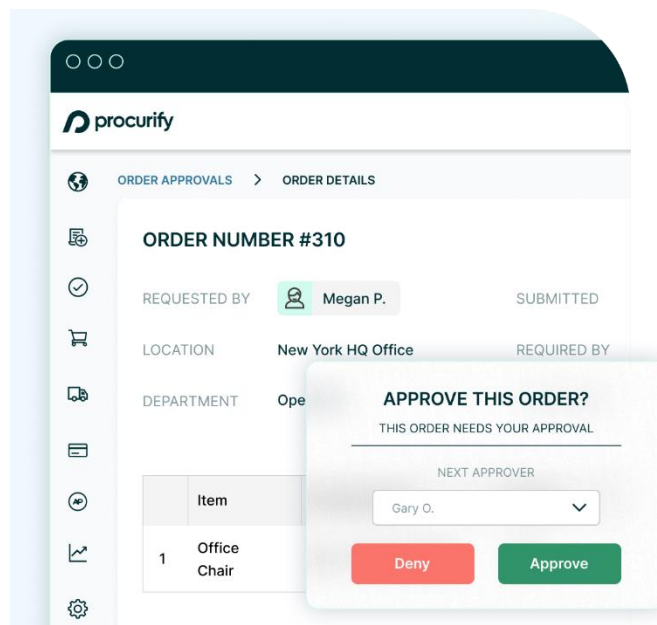


Figura 2 - Pedido de requisição no Procurify (retirado de [13])

O Procurify apresenta as seguintes funcionalidades, entre outras [14]:

- Formulários de requisição online. Isto permite a criação, aprovação e seguimento digital das requisições;
- Fluxos de aprovação. As requisições podem ser direcionadas para diferentes atores consoante as características da requisição, respeitando, assim, possíveis regras da organização;
- Atualizações em tempo real. Requisitantes e aprovadores podem ver o estado da requisição em tempo real. Também recebem notificações por email;
- Relatórios de requisições.

2.1.2.2.2 Tradogram

CREATE REQUISITION

Requisition Title: *
Supplies for new hire

Notes:
Need these items to be delivered to finance before the end of the month.

UPLOAD REQUISITION FILES:

Choose the Department: *
Finance

Associated Project:
None

Distribution Address:
None

ATTACH FILLABLE FORM:

Currency: USD

Due Date:

Item Name	Item Number	Description	Custom Fields	Spec. Files	U.O.M.	Qty	Estimated Unit Price	Subtotal
Apple iPad	77832UO				Unit, each	1	440	440.00
Apple iMac Desktop	192839 MK				Unit, each	1	1400	1,400.00
Stapler	492020				each	1	15.99	15.99
TOTAL:								USD \$1,855.99

ADD DATABASE ITEMS **ADD EXTERNAL ITEMS** **CONFIRM** **SAVE AS DRAFT** **SUBMIT REQUISITION**

Figura 3 - Criar requisição no Tradogram (retirado de [15])

Algumas das funcionalidades oferecidas pelo Tradogram [16]:

- Criação de orçamentos para prevenir uso excessivo de fundos;
- Gestão de aprovação. Requisições requerem aprovação de utilizadores específicos definidos previamente;
- Envio de notificações;
- Geração automática de ordens de compra;
- Acompanhamento do estado das requisições. Utilizadores podem seguir online o estado das suas requisições quando pretenderem;
- Geração de relatórios relacionados com requisições.

2.1.2.2.3 Asset Infinity

	Request Type	Location	Requisition No	Status	Pending Approver
<input type="checkbox"/>	Deo-Item	Abcdef loc	R1220222356	Completed	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220222354	Pending	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220222351	Pending	
<input type="checkbox"/>	Deo-Item	Abcdef loc - 2nd cross	R1220222350	Completed	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220222349	Pending	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220222348	Pending	
<input type="checkbox"/>	Pending Allocation Check	Building2	R1220222345	Pending	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220221336	Completed	
<input type="checkbox"/>	Pending Allocation Check	Abcdef loc	R1220221335	Pending	
<input type="checkbox"/>	Deo-Cat	Abcdef loc	R1220221334	Completed	

Figura 4 - Lista de requisições no Asset Infinity (retirado de [17])

As principais funcionalidades do Asset Infinity são [17]:

- *Templates* de requisição. Itens e serviços requisitados múltiplas vezes podem ser definidos como *template* para facilitar o processo de requisição;
- Criação de orçamentos para prevenir uso excessivo de fundos;
- Conversão automática para ordens de compra;
- Relatórios;
- Notificações;
- Gestão de inventário.

2.1.3 Sistemas de gestão de inventário

Uma gestão de inventário eficaz é, também, um aspeto importante para a organização. No seguimento da aprovação de uma requisição os itens requeridos devem ser atualizados no inventário, reduzindo, assim, o risco de excesso ou escassez de stock. Uma boa gestão de inventário ajuda as organizações a otimizar os seus níveis de stock, reduzir custos e aumentar a eficiência. [2]

2.1.3.1 Benefícios no uso de sistemas de gestão de inventário

Existem múltiplos benefícios em usar um sistema de gestão de inventário:

- Ajuda organizações a controlar melhor o seu inventário, reduzindo o risco de excesso e escassez de stock [2];

- Reduz custos ao otimizar os níveis de stock e diminui a necessidade de stock de reserva [2];
- Aumento da eficiência ao reduzir o tempo e esforço necessário para gerir o inventário [2], [3];
- Melhora a experiência dos utilizadores ao garantir a disponibilidade dos produtos [3].

2.2 Tecnologias existentes

Para construir este projeto foi necessária a utilização de tecnologias orientadas ao desenvolvimento de aplicações *web*. Nesta secção essas tecnologias são descritas e analisadas.

O projeto foi desenvolvido no *Visual Studio Code* utilizando a linguagem C# para o desenvolvimento do *back-end* e, no *front-end*, Typescript, HTML e CSS. A *framework* utilizada no *back-end* foi o .NET Core e, no *front-end*, foi usado Angular. A base de dados escolhida para a persistência de dados foi o SQL Server. Optou-se por estas tecnologias pois foram as tecnologias sugeridas pela empresa.

2.2.1 .NET Core

.NET Core é uma *framework* desenvolvida pela Microsoft com o objetivo de construir aplicações modernas, incluindo aplicações *web*. Foi lançada em 2016 como um *re-design* das versões anteriores de .NET. Ao contrário das versões anteriores esta trata-se de uma *framework open-source* e compatível com Windows, Linux e macOS. Aplicações usando .NET Core podem ser escritas usando C#, F# ou Visual Basic. [18], [19]

Existem várias vantagens em usar .NET Core sendo que a principal se encontra na sua performance. Isto deve-se às suas características, incluindo suporte para programação assíncrona e pedidos HTTP leves e de alto desempenho que são características extremamente importantes no desenvolvimento de aplicações *web*. [18], [20]

2.2.2 Angular

Desenvolvido e mantido pela Google, o Angular é uma *framework* única para a construção de aplicações *front-end* escrita em TypeScript. Foi construído para simplificar o desenvolvimento de *single-page applications* (SPA) e proporcionar uma estrutura de código modular e organizada. [21]

Algumas das suas principais características são:

- Ao ser uma *framework* baseada em componentes, é possível construir aplicações através do uso de componentes reutilizáveis que contêm tanto o HTML *template* como a lógica em TypeScript, oferecendo, assim, uma alta escalabilidade [22];
- Suporta *data binding* bidirecional, permitindo que o estado do *data model* se altere automaticamente quando são efetuadas mudanças na UI, e vice-versa, aumentando o dinamismo e interatividade da aplicação [23];
- Suporta injeção de dependências, aumentando a modularidade e flexibilidade das aplicações [22].

Trata-se de uma das *frameworks* mais utilizadas, sendo usada por empresas como BMW, Forbes e Xbox [24].

2.2.3 SQL Server

Base de dados é uma coleção organizada de informação estruturada normalmente guardada num sistema computacional [25]. As bases de dados podem ser divididas em relacionais e não relacionais. Bases de dados relacionais guardam os dados em tabelas compostas por linhas e colunas [25]. A relação entre tabelas e o formato dos dados é chamado de *schema*. Para bases de dados relacionais, o *schema* deve estar definido claramente [26]. Por outro lado, uma base de dados não relacional guarda dados de maneira a permitir uma estrutura de dados flexível, não necessitando de um *schema* pré-definido para adicionar ou remover dados [27].

Uma das base de dados mais famosas é o SQL Server [26]. SQL Server é uma base de dados relacional desenvolvida pela Microsoft. Trata-se de um sistema que oferece velocidade, elevada performance, excelente restauro de dados e segurança [28].

2.2.4 .NET Core vs Node.js

Existem várias tecnologias usadas no desenvolvimento de REST APIs e aplicações *back-end*. Duas das mais populares são .NET Core e Node.js. Ao contrário do .NET, Node.js é mais maturo tendo sido lançado a sua primeira versão em 2009. Ambos são conhecidos pela sua velocidade, desempenho e alta escalabilidade. [29]

O Node.js permite a escrita de código *server-side* usando JavaScript [30]. Uma das suas principais características é o facto da sua execução ser *single-thread*, ou seja, ao contrário das outras *frameworks*, apenas uma *thread* é responsável por executar o código Javascript da aplicação [30].

Ambas as tecnologias apresentam diferenças e semelhanças. Foi então feita uma comparação entre elas presente na tabela 1.

Tabela 1 - Comparação entre as características do .NET Core e Node.js

<i>Caraterística</i>	<i>.NET CORE</i>	<i>Node.js</i>
<i>Linguagem</i>	C#, F#, Visual Basic	JavaScript
<i>Plataforma de Suporte</i>	Windows, Linux, macOS	Windows, Linux, macOS
<i>Programação Assíncrona</i>	Sim	Sim
<i>Thread</i>	<i>Multi-Thread</i>	<i>Single-Thread</i>
<i>Popularidade</i>	Popular	Extremamente popular
<i>Familiaridade com a linguagem</i>	Sim	Sim

2.2.5 Angular vs React

Uma das alternativas ao uso de Angular para o desenvolvimento da aplicação *front-end* era o React.

React é uma biblioteca em JavaScript extremamente popular construída pelo Facebook. Funciona através da utilização de um *DOM* virtual, ou seja, sempre que o estado de um componente sofre alterações é criado um novo *DOM*, virtual. Este último é comparado com o anterior e o *DOM* real apenas é modificado onde existirem diferenças. React é conhecido pela sua performance, escalabilidade e facilidade na reutilização de componentes UI. [31]

Foi realizada uma comparação entre as características do Angular e do React:

Tabela 2 - Comparação entre características do Angular e React

<i>Caraterística</i>	<i>Angular</i>	<i>React</i>
<i>Linguagem</i>	TypeScript	JavaScript
<i>Data binding</i>	Bidirecional	Unidirecional
<i>Framework vs Library</i>	<i>Framework</i>	<i>Library</i>
<i>Dom</i>	Incremental	Virtual
<i>Performance</i>	Mais lenta	Mais rápida

Familiaridade com a
tecnologia

Sim	Não
-----	-----

Nos dias atuais, React é uma das *frameworks* mais utilizadas para desenvolvimento *Web*, sendo a mais usada no desenvolvimento de aplicações *front-end* pelos utilizadores do StackOverflow (Figura 5) [32]. É usada por empresas como Facebook, Netflix e Uber [33].

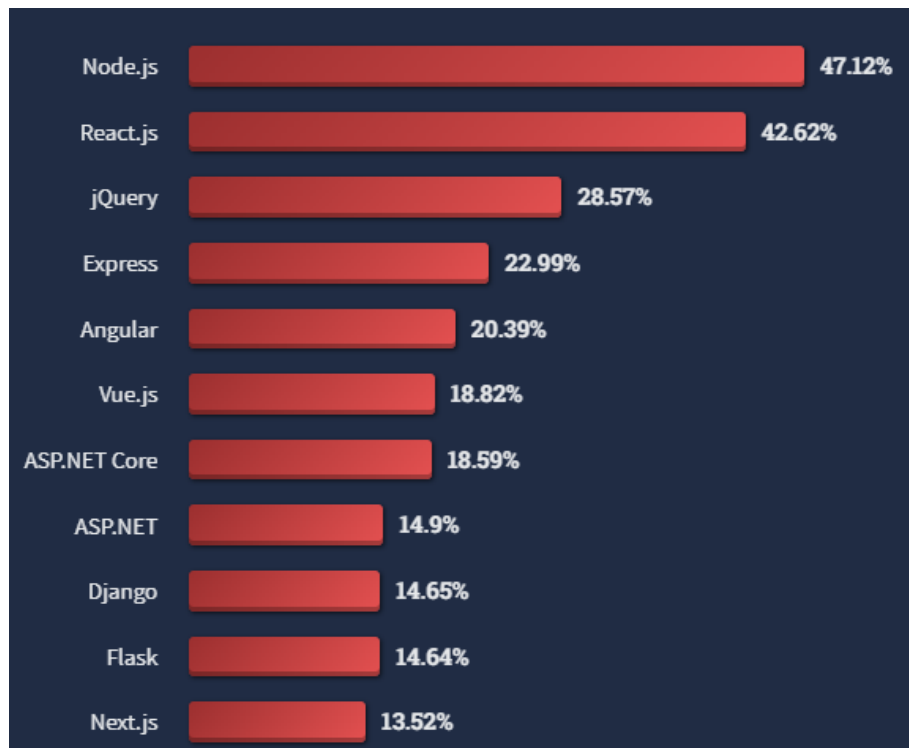


Figura 5 - Web frameworks mais utilizadas [32]

3 Análise e desenho da solução

Este capítulo tem como foco a análise e design da solução implementada. Na parte da análise, são especificados os conceitos de domínio do problema e descritos os requisitos funcionais e não funcionais do sistema. No design, é apresentada a arquitetura global da solução assim como o desenho de algumas das funcionalidades mais importantes.

3.1 Domínio do problema

Após o levantamento de requisitos e elaboração da descrição dos casos de usos necessários ao desenvolvimento da aplicação, foi criado o modelo de domínio para a solução proposta.

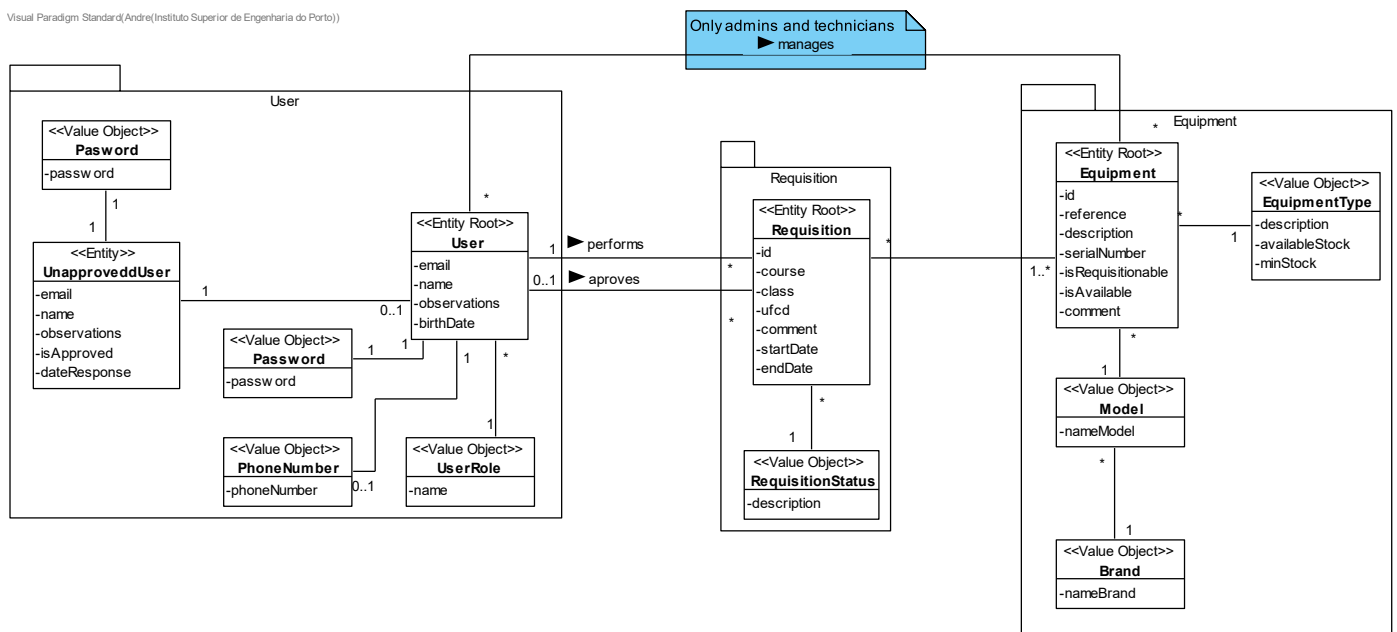


Figura 6 - Modelo de domínio solução

Foram identificados os conceitos presentes na figura 6:

- *User* refere-se a um utilizador da aplicação, podendo ele ser um utilizador com permissões base que deseja efetuar uma requisição, um técnico ou um administrador, distinguido pela *UserRole*. Um *User* é identificado pelo seu email e faz *log in* na aplicação através da combinação email-password. Possui ainda um nome, número de telefone, data de nascimento e um campo livre de observações.
- *UnapprovedUser* trata-se de um utilizador que efetuou pedido de registo na aplicação, mas ainda não foi aprovado por um técnico ou administrador
- *Equipment* refere-se a um equipamento. Um equipamento possui um id, uma referência, uma descrição, um número de série, uma marca, um modelo, um

comentário, um tipo de equipamento (câmara, microfone, etc.), um atributo para saber se o equipamento se encontra em condições aceitáveis para ser requisitado e um atributo para saber se o equipamento se encontra disponível de momento.

- Uma *Requisition* trata-se de uma requisição de equipamentos e é o conceito mais importante da aplicação. É identificada por um id e possui o curso do requisitante, a turma, a UFDC, a data de início e de fim, a lista de equipamentos pretendida, um estado e o utilizador que a efetuou. Esta requisição é depois aprovada ou negada por um técnico e, aquando da devolução da requisição, o técnico pode escrever um comentário.

3.2 Requisitos funcionais e não funcionais

Através dos objetivos presentes na proposta da empresa e uma reunião de esclarecimento de dúvidas com o Supervisor da empresa, foram definidos os requisitos funcionais e não funcionais para o projeto em questão. Fazendo uso de reuniões regulares, alguns dos requisitos iam sendo ajustados e aprofundados até atingir o seu estado final.

3.2.1 Requisitos funcionais

Foram identificados quatro atores intervenientes na aplicação:

- O utilizador, que efetua pedidos de requisição
- O técnico, que gere os equipamentos e as requisições.
- O administrador, que gere todos os utilizadores e as propriedades do sistema. Também tem acesso a todas a funcionalidade do técnico.
- O utilizador não registado, que efetua pedido de registo para se tornar num utilizador com permissões base.

Considerando os atores, o problema e o esclarecimento de dúvidas, foram definidos os casos de uso presentes da figura 7.

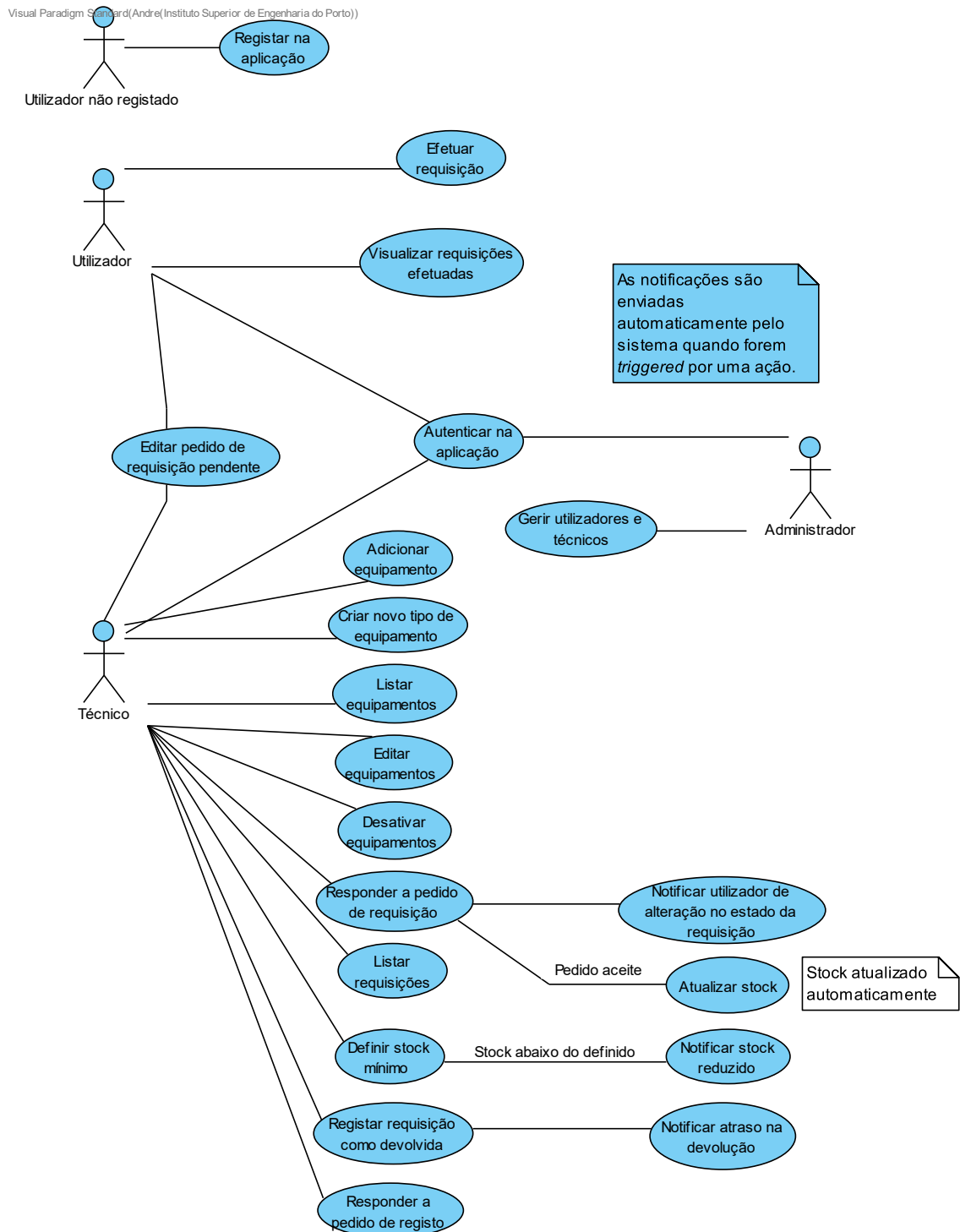


Figura 7 - Diagrama de casos de uso

Foram também descritas as *user stories* e os requisitos correspondentes a cada uma:

US1. Como utilizador não registado pretendo registar-me na aplicação.

- O sistema deve permitir o pedido de registo de novos utilizadores.
- O sistema deve apresentar um formulário de registo.
- O sistema deve enviar um email de aceitação/recusa do pedido de registo.

- US2. Como utilizador, técnico ou administrador, pretendo autenticar-me na aplicação para ter acesso às suas funcionalidades.
 - O sistema deve permitir o login através da introdução do email e password.
- US3. Como técnico, pretendo responder a pedidos de registo de utilizadores.
 - O sistema deve permitir ao técnico visualizar pedidos de registo pendentes.
 - O sistema deve permitir ao técnico aceitar ou recusar pedidos de registo.
- US4. Como administrador, pretendo gerir os utilizadores e técnicos.
 - O sistema deve permitir ao administrador eliminar utilizadores.
 - O sistema deve permitir ao administrador criar técnicos ou outros administradores.
- US5. Como técnico, pretendo criar um novo tipo de equipamento.
 - O sistema deve permitir ao técnico a criação de um novo tipo de equipamento.
 - O sistema deve permitir ao técnico a eliminação de um tipo de equipamento que não possua nenhum equipamento associado.
- US6. Como técnico, pretendo introduzir equipamentos na aplicação.
 - O sistema deve permitir a criação de equipamentos.
 - O sistema deve apresentar um formulário para introdução de informação sobre o equipamento.
- US7. Como técnico, pretendo listar os equipamentos existentes.
 - O sistema deve permitir a listagem de equipamentos.
 - O sistema deve permitir a utilização de filtros na listagem de equipamentos.
- US8. Como técnico, pretendo editar equipamentos existentes.
 - O sistema deve permitir a edição de informação sobre os equipamentos.
- US9. Como técnico, pretendo eliminar equipamentos.
 - O sistema deve permitir a eliminação de equipamentos.
- US10. Como utilizador, pretendo fazer uma requisição de equipamentos.
 - O sistema deve permitir a criação de um pedido de requisição de equipamentos disponíveis.
 - O sistema deve permitir ao utilizador filtrar os equipamentos.
 - O sistema deve permitir ao utilizador seleccionar os equipamentos pretendidos e as suas quantidades.
 - O sistema deve permitir ao utilizador seleccionar a data de início e fim da requisição garantindo que os equipamentos se encontram disponíveis.
- US11. Como utilizador, pretendo visualizar as minhas requisições.
 - O sistema deve permitir ao utilizador a listagem das requisições efetuadas.
 - O sistema deve permitir ao utilizador filtrar as requisições efetuadas pela sua aceitação ou recusa.
- US12. Como utilizador, pretendo editar um pedido de requisição pendente.
 - O sistema deve permitir ao utilizador a edição de um pedido de requisição pendente.
 - O sistema deve permitir ao utilizador a eliminação de um pedido de requisição pendente.
- US13. Como técnico, pretendo responder um pedido de requisição.
 - O sistema deve permitir ao técnico visualizar os pedidos de requisição pendentes.
 - O sistema deve permitir ao técnico aceitar ou recusar um pedido de requisição.
- US14. Como técnico, pretendo editar um pedido de requisição pendente.

- O sistema deve permitir ao técnico editar um pedido de requisição pendente.
- US15. Como técnico, pretendo visualizar as requisições efetuadas.
 - O sistema deve permitir ao técnico visualizar as requisições efetuadas.
 - O sistema deve permitir ao técnico filtrar as requisições efetuadas por utilizador, equipamento, data e estado de aceitação.
- US16. Como utilizador, pretendo ter conhecimento de alterações no estado do meu pedido de requisição.
 - O sistema deve enviar uma notificação por email ao utilizador com o estado do pedido de requisição mal ele seja alterado.
- US17. Como técnico, pretendo definir o stock mínimo para cada tipo de equipamento.
 - O sistema deve permitir ao técnico definir o stock mínimo para cada tipo de equipamento.
- US18. Como técnico, pretendo ser notificado num caso de stock reduzido.
 - O sistema deve notificar o técnico por email no caso de o stock para um tipo de equipamento seja inferior ao stock mínimo.
- US19. Como técnico, pretendo registar uma requisição como devolvida.
 - O sistema deve permitir ao técnico registar uma requisição como devolvida.
 - O sistema deve permitir ao técnico escrever comentários sobre os equipamentos da requisição.
- US20. Como técnico, pretendo ser notificado no caso de atraso numa devolução da requisição.
 - O sistema deve notificar o técnico caso a devolução de uma requisição não seja efetuada a tempo.
- US21. Como técnico, pretendo que o stock de cada tipo de equipamento atualize automaticamente.
 - O sistema deve atualizar o stock de cada tipo de equipamento quando houver uma requisição ou devolução da mesma.

3.2.2 Requisitos não funcionais

Para definir os requisitos não funcionais foi adotado o modelo FURPS+. FURPS é um acrónimo para classificar os requisitos em várias categorias funcionalidade (*functionality*), usabilidade (*usability*), confiabilidade (*reliability*), desempenho (*performance*) e suportabilidade (*supportability*) [34]. O modelo FURPS+ é uma extensão do FURPS em que são adicionadas limitações de desenho, implementação, interface e físicas [34].

Usabilidade

A usabilidade refere-se à interface gráfica, tanto na parte estética como na usabilidade, consistência e documentação [34].

Foram, então, definidos os seguintes requisitos:

- A aplicação deve ser fácil de usar, sendo acessível a qualquer utilizador.
- O tema (cores, tamanho e tipo de letra) deve ser consistente em toda a interface.

- Erros, como pedidos para o *back-end* sem resposta, não devem impedir o funcionamento da aplicação.

Confiabilidade

A confiabilidade inclui aspetos como a disponibilidade, recuperação em caso de falha e risco de falha [35].

Foi definido o requisito:

- A plataforma deve permanecer operacional na ocorrência de erros ou falhas.

Desempenho

O desempenho envolve aspetos como velocidade de resposta do sistema, tempo de arranque do sistema e uso de recursos [34], [35].

O requisito correspondente identificado foi:

- A aplicação deve oferecer respostas rápidas às ações dos utilizadores.

Suportabilidade

A suportabilidade agrupa várias características como testabilidade, adaptabilidade, manutenibilidade, compatibilidade, escalabilidade, entre outros [34].

Os requisitos destacados foram:

- A aplicação deve ser compatível com *browsers* de internet, tanto em computador como em mobile.
- O código deve respeitar padrões como SOLID, injeção de dependências e GRASP.

Limitações de desenho

As limitações de desenho, como o nome indica, limitam o desenho da aplicação [34].

Foram identificados os seguintes requisitos:

- A aplicação deve seguir o modelo Cliente-Servidor.
- A aplicação deve seguir uma arquitetura Onion.

Limitações de interface

As limitações de interface referem-se a interações entre sistemas, sejam eles internos ou externos [36].

Foram destacados os seguintes requisitos:

- O *front-end* deve comunicar com o *back-end* através de uma REST API para garantir que os dados são transferidos de maneira segura e eficiente.
- O *back-end* deve comunicar com a base de dados SQL Server através da SQL Server API.

3.3 Desenho

Para desenhar a aplicação foi usada a combinação de dois modelos de representação arquitetural: C4 e 4+1.

O modelo C4 permite descrever o sistema e o ecossistema que o rodeia através de quatro níveis de abstração hierárquica [37]. Cada nível apresenta maior detalhe do que o nível que o antecede [38], [39]:

- Nível 1 – descrição do sistema como um todo.
- Nível 2 – descrição de contentores do sistema.
- Nível 3 – descrição de componentes dos contentores.
- Nível 4 – descrição do código.

Este subcapítulo encontra-se dividido pelos níveis de abstração, à exceção do nível referente ao código, finalizando com o diagrama de classes.

O modelo de vistas 4+1 descreve o software com base em quatro diferentes visões do sistema [40]. Utilizadores, programadores, administradores de sistemas e *project managers* olham para o sistema de diferentes pontos de vista [40]–[42]:

- Vista lógica – descreve os aspetos do software que procuram responder aos desafios do negócio.
- Vista de processos – descreve o fluxo de uma funcionalidade e interações no sistema.
- Vista de implementação – descreve a organização do software no seu ambiente de desenvolvimento.
- Vista física – descreve o sistema da perspetiva de um engenheiro de sistemas, ou seja, onde é executado o software.
- Vista de cenários – associa os processos de negócio com os atores que os realizam.

3.3.1 Nível de sistema

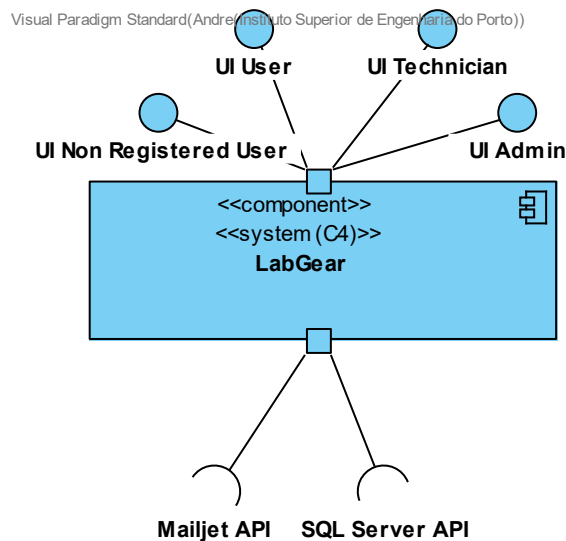


Figura 8 - Vista lógica (nível 1)

Foi desenhada a vista lógica, presente na figura 8, que mostra todos os utilizadores do sistema, já definidos anteriormente, e os sistemas externos com que existe interação. É chamada a SQL Server API para a persistência, recolha e alteração dos dados e Mailjet API para envio de notificações por email necessárias.

3.3.2 Nível de contentores

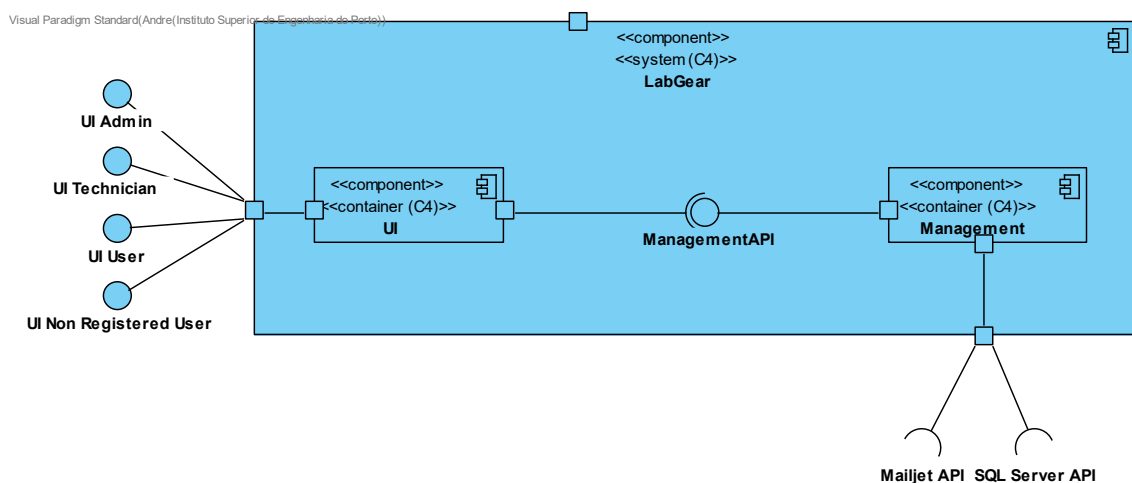


Figura 9 - Vista lógica (nível 2)

A aplicação encontra-se dividida em *front-end* e *back-end*, representados na figura 9 e figura 10, com os nomes UI e Management, respetivamente. A UI comunica com o Management através da Management API que, por sua vez, comunica com a API da base de dados (SQL Server API) e a API para enviar emails (Mailjet API).

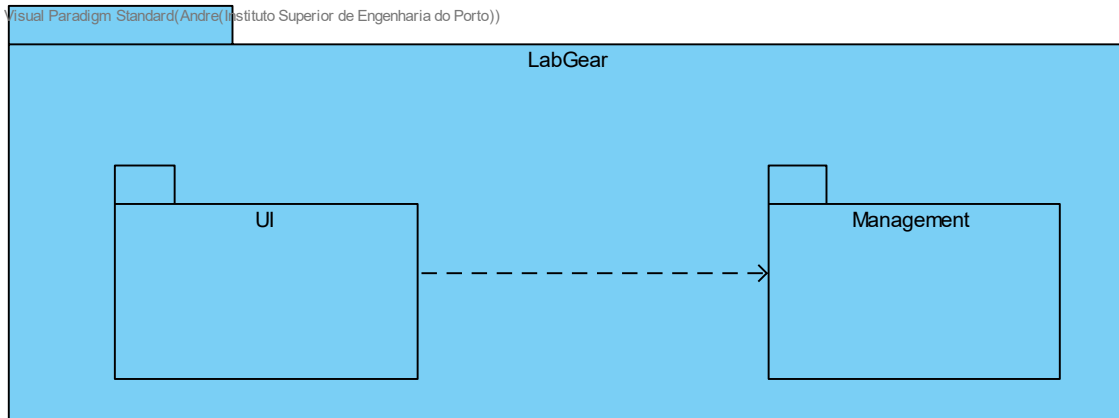


Figura 10 - Vista de implementação (nível 2)

3.3.3 Nível de componentes

O contentor Management é composto por vários componentes usando uma arquitetura Onion (figura 11). Neste tipo de arquitetura cada componente possui funções específicas e têm responsabilidades distintas [43]. A Management API comunica com a componente *Routing* para enviar o pedido recebido para o *Controller* correto. O *Controller* recebe o pedido HTTP, extrai a informação relevante e delega o processamento para a camada dos serviços. A componente dos serviços é responsável pelas funcionalidades essenciais do negócio. Recebe pedidos do *Controller*, realiza as operações necessárias e comunica com a camada *Repository* para ir buscar, modificar ou acrescentar dados presentes na base de dados. A componente *Repository* é responsável pelo acesso aos dados e a sua persistência. O *Domain Model* representa as entidades e objetos com que a aplicação trabalha. Define a estrutura e comportamento das entidades e é usado pelo *Service* para operações de lógica de negócio. Finalmente, DTO serve para transferir dados entre diferentes componentes da aplicação.

Por sua vez, o contentor UI é composto por quatro componentes (figura 12). A *View* é responsável pela interação com o utilizador e recebe os seus inputs, direcionando-os para o *Controller* se necessário. O *Controller* é responsável pelo controlo das funcionalidades apresentadas pela *View*, servindo como ligação entre a *View* e os *Services*. O componente *Service* é responsável pela comunicação o *back-end*. Por último, os DTOs servem para organizar a informação, tornando mais fácil a comunicação com os restantes componentes.

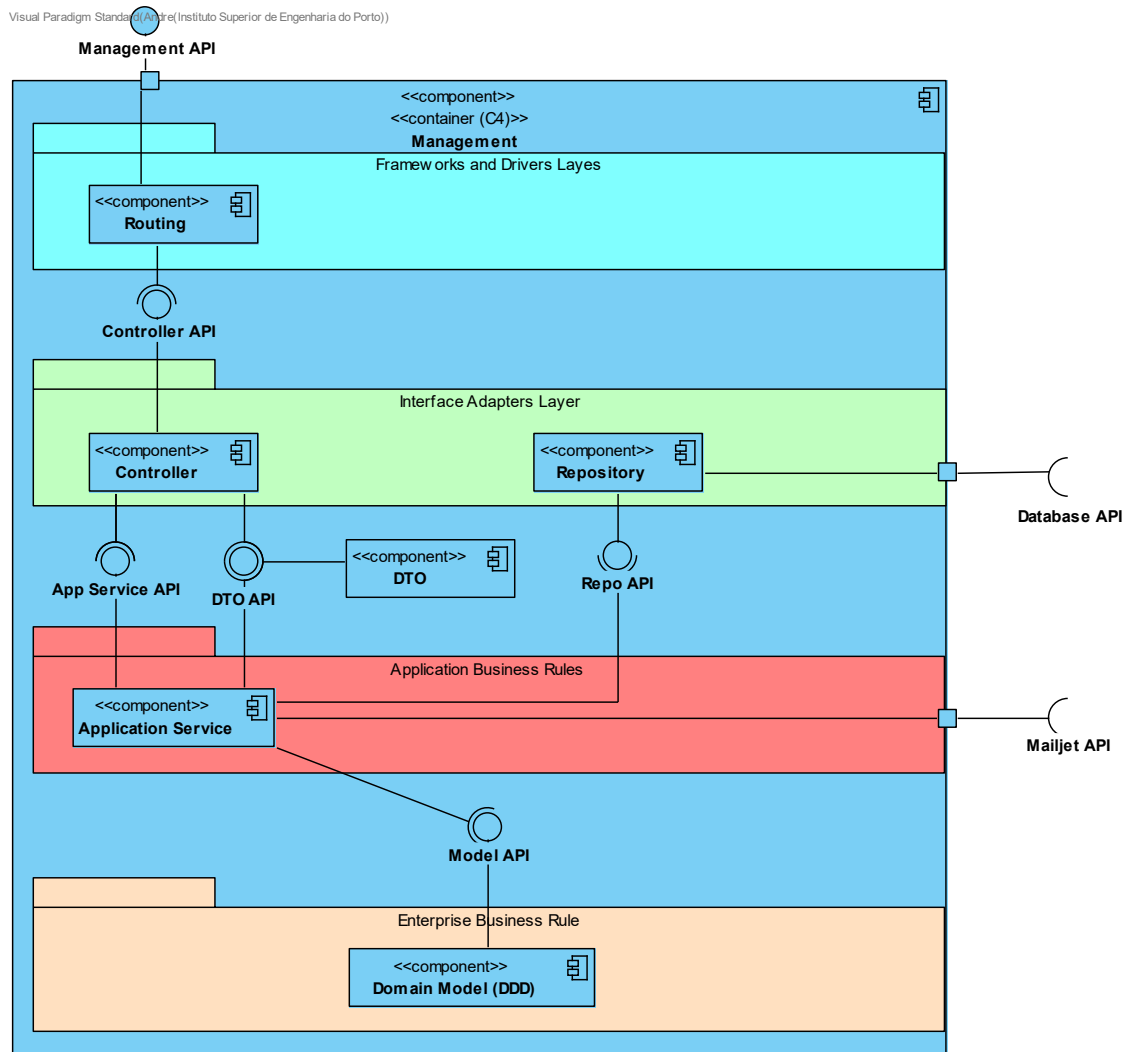


Figura 11 - Vista lógica do Management (nível 3)

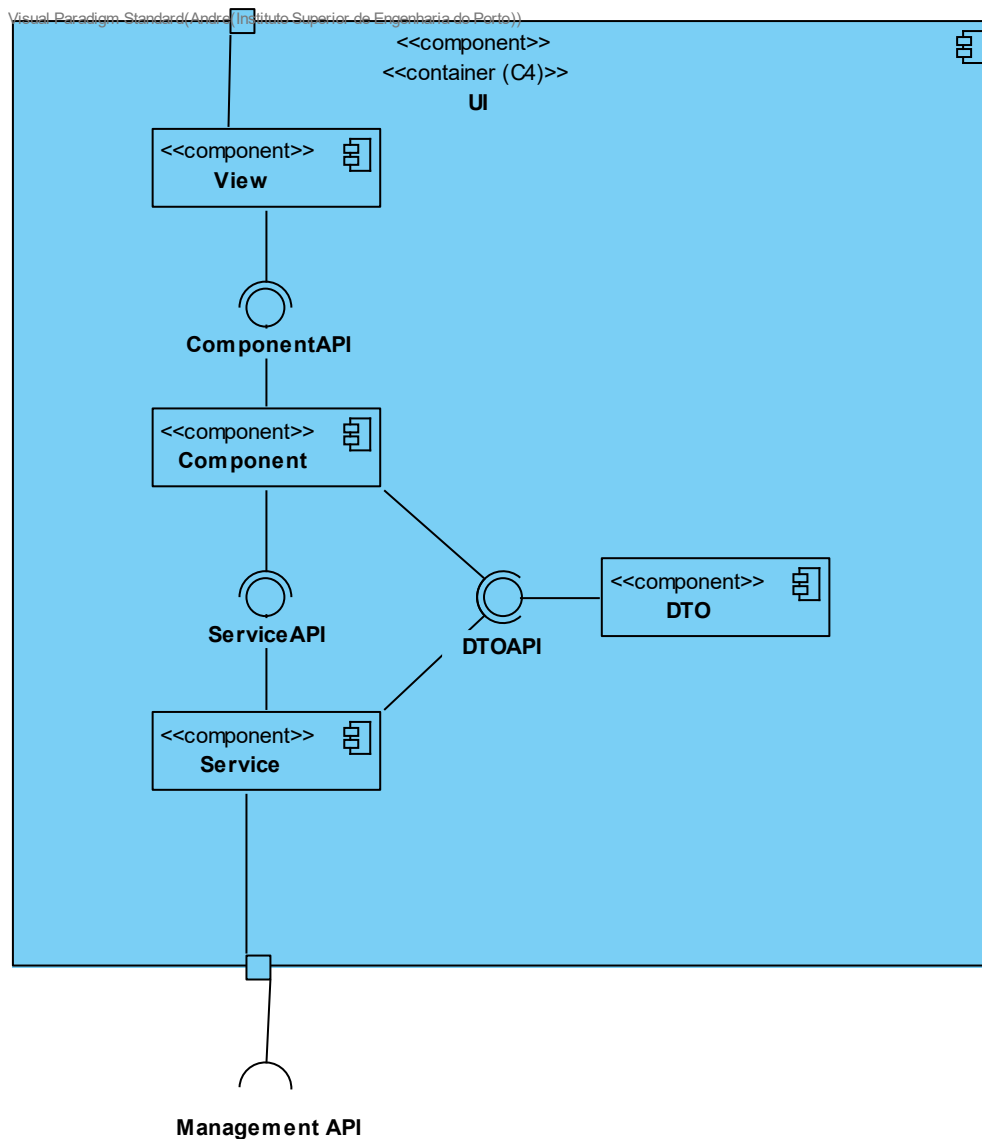


Figura 12 - UI vista lógica (nível 3)

Foi também desenhada a vista de processos para todos os casos de uso, sendo duas das funcionalidades mais importantes: adicionar equipamento (UC05) e efetuar requisição (UC09).

No caso de uso de adicionar equipamento representado pelas figuras 13 e 14, o Técnico pretende inserir um equipamento na aplicação. É feito um pedido GET ao Management que retorna todas as categorias de equipamentos existentes. Caso a categoria de equipamento pretendida ainda não existir, então o Técnico necessitará de a criar primeiro (caso de uso 5). O Técnico seleciona a categoria de equipamento correspondente e é-lhe pedido para introduzir os restantes dados correspondentes ao equipamento.

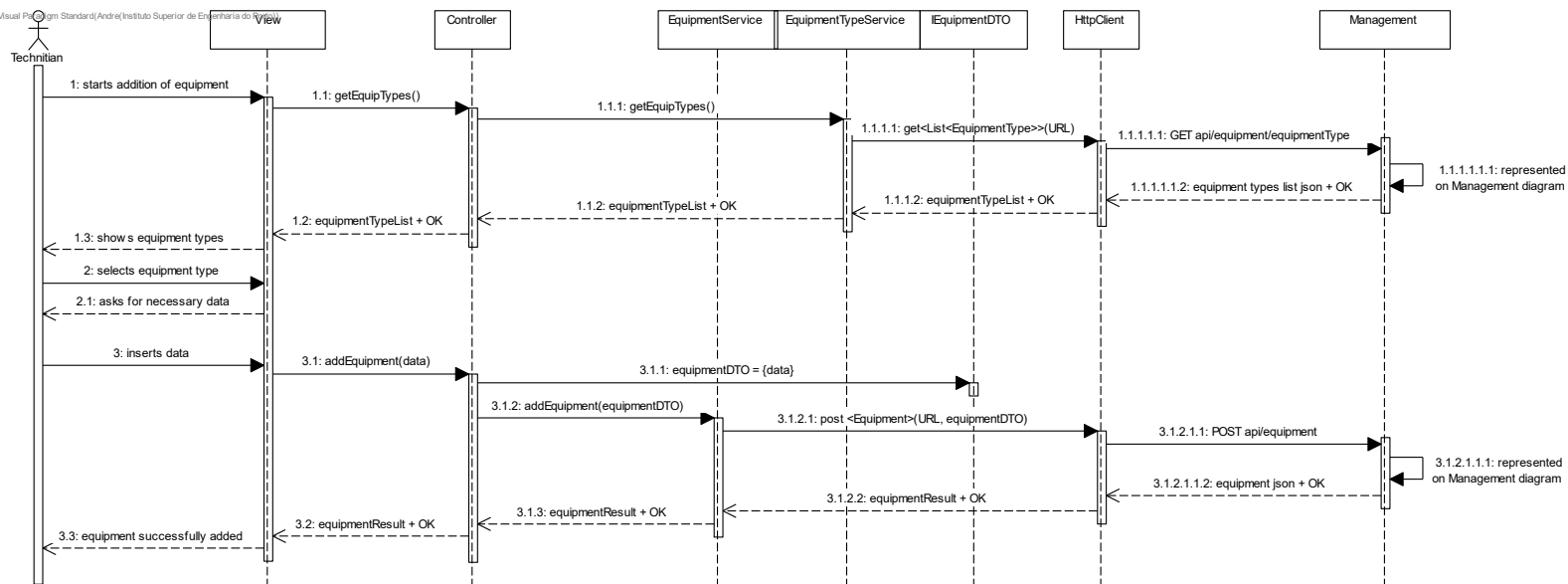


Figura 13 - UC05 vista de processos front-end

No *front-end* é criado um DTO no *Controller* que é passado para o *Service*, onde é feito um pedido HTTP POST ao *back-end*. O *Management* recebe o pedido através do *Controller* o qual mapeia os dados recebidos para um DTO do equipamento e chama o *Service*. No *Service* é criado um objeto de domínio *Equipment* que é passado ao repositório para o adicionar à base de dados e é chamado o *UnitOfWork* para fazer *commit* e persistir os dados. Por fim, o equipamento adicionado é passado para um DTO e retornado até ao técnico que é informado do sucedido da operação.

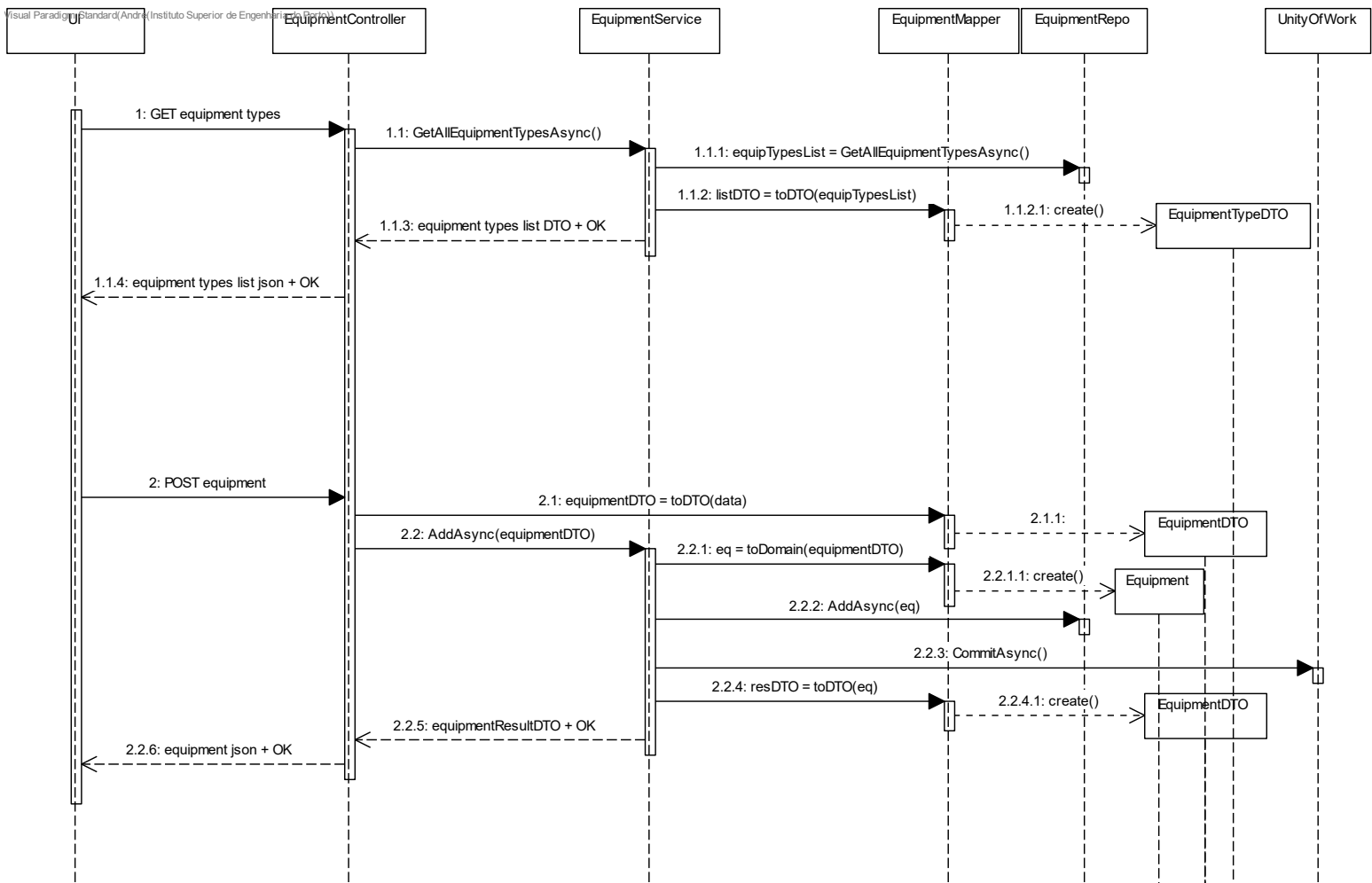


Figura 14 - UC05 vista de processos back-end

O caso de uso de efetuar uma requisição segue uma abordagem semelhante ao anterior, representado nas figuras 15 e 16. O Utilizador pretende efetuar uma requisição de equipamentos. A UI começa por fazer um pedido GET ao Management para ir buscar uma lista de todos os equipamentos requisitáveis e mostra-os ao Utilizador. É possível filtrar os equipamentos por marca e categoria, sendo que essa funcionalidade pertence ao domínio do UC07.

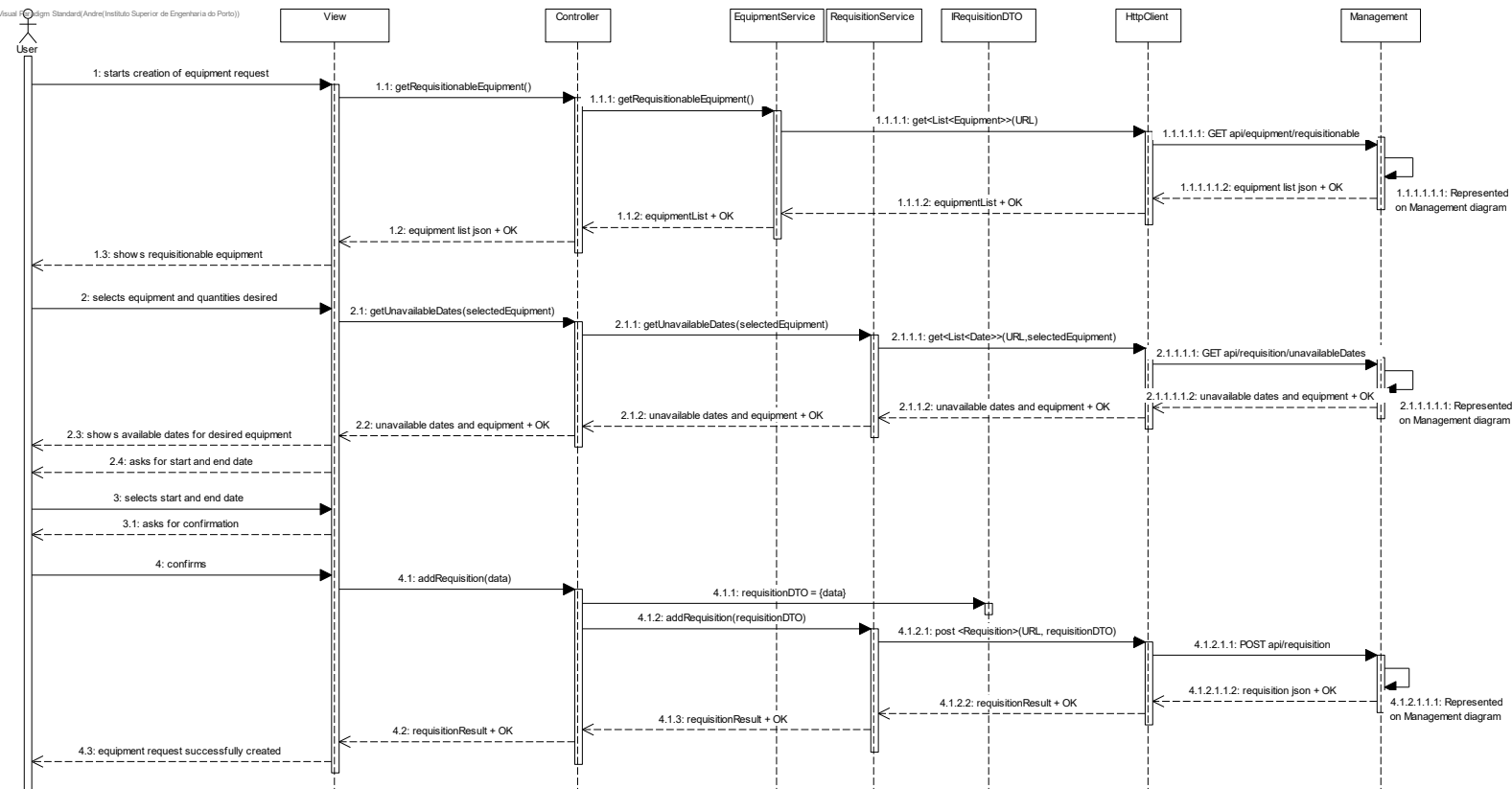


Figura 15 - UC09 vista de processos UI

O Utilizador escolhe os equipamentos desejados e é efetuado um pedido GET com os equipamentos como parâmetro que retorna uma lista de datas em que o conjunto de equipamentos pretendidos se encontra indisponível. É pedido ao Utilizador para seleccionar uma data de início e data de fim da requisição e é realizado um pedido POST normal para guardar o pedido de requisição na base de dados.

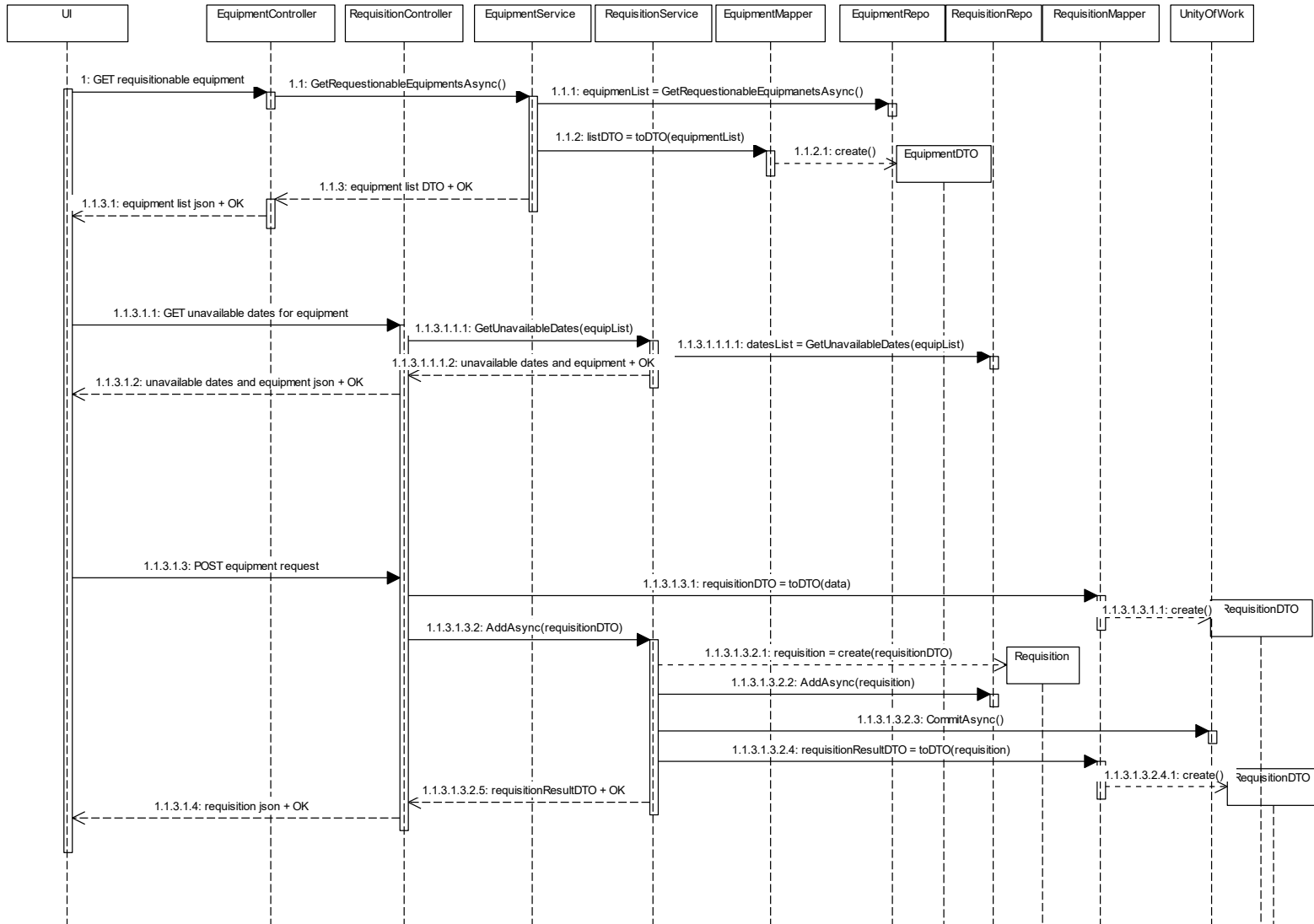


Figura 16 - UC09 vista de processos Management

3.3.4 Diagrama de classes

Foi elaborado um diagrama de classes para representar a estrutura dos elementos do domínio na base de dados e as ligações existentes entre eles.

Visual Paradigm Standard (André (Instituto Superior de Engenharia do Porto))

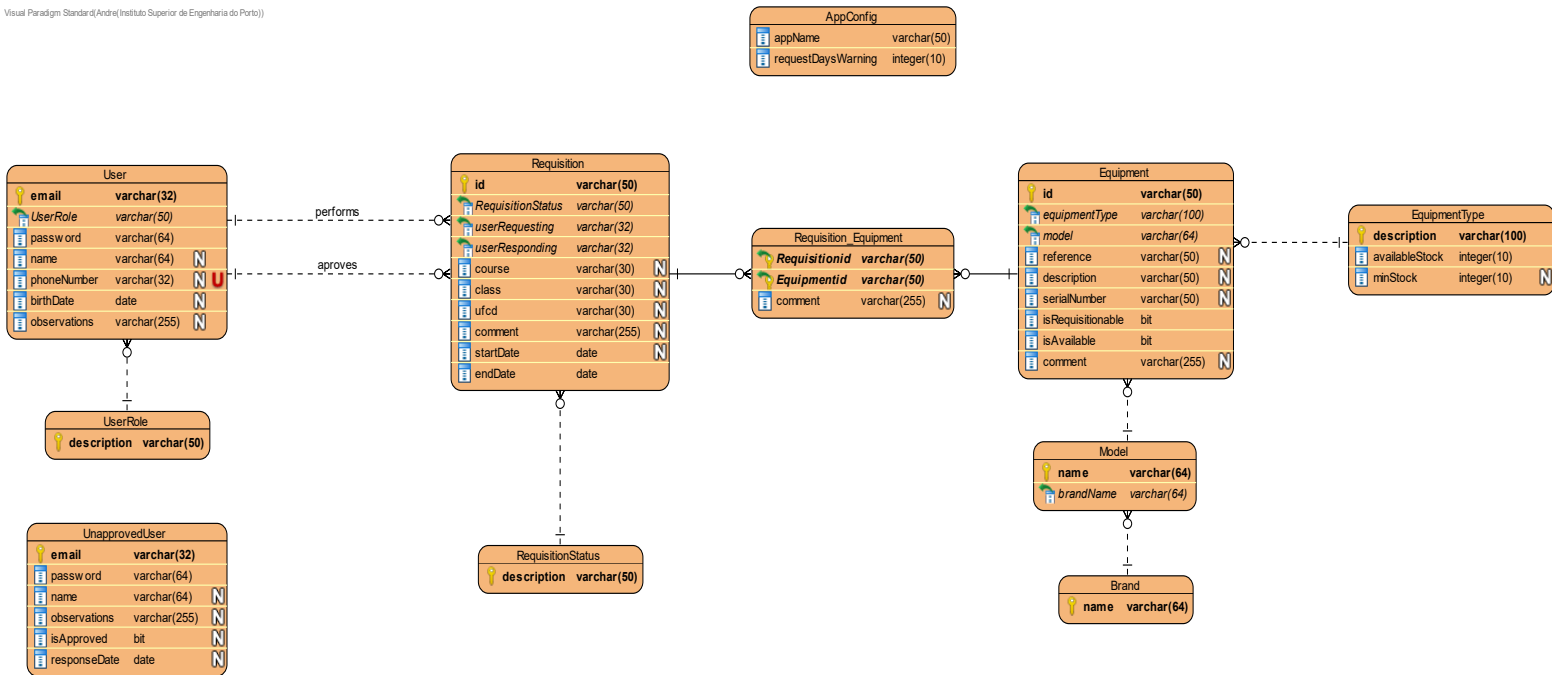


Figura 17 - Diagrama de classes

Como as requisições e os equipamentos possuem uma relação de muitos para muitos, isto é, uma requisição pode ter múltiplos equipamentos e um equipamento pode encontrar-se em várias requisições, foi criada uma tabela chamada “Requisition_Equipment” que contém os equipamentos presentes em cada requisição.

A requisição guarda, também, o email do utilizador que a efetua e o email do técnico que a aceitar ou recusar.

4 Implementação da Solução

Neste capítulo é realizada um resumo da implementação da solução. Primeiro é descrita a implementação feita através de evidencias do mesmo. É, depois, feita uma descrição dos testes realizados. Por fim, é descrita a abordagem para avaliar a solução.

4.1 Descrição da implementação

Neste subcapítulo são descritas as funcionalidades implementadas, referindo os seus fluxos de execução e bibliotecas e APIs usadas nos diversos casos de uso.



Figura 18 - Menu da aplicação (função administrador)

Para utilizar as funcionalidades presentes na aplicação, existe um menu de navegação presente na UI. Este menu apresenta diferenças consoante a função do utilizador que esteja com sessão iniciada no sistema, podendo aparecer um número maior ou menor de opções. O menu do administrador apresenta todas as opções e dá acesso a todas as funcionalidades implementadas, os técnicos apresentam limitações na gestão de contas e os restantes

utilizadores apresentam as mesmas restrições que os técnicos e não possuem acesso à gestão de equipamentos. Estas restrições são executadas através de verificações da sessão do utilizador, tanto a nível de *front-end*, não deixando os utilizadores acederem ou visualizarem componentes que não deviam ter acesso, como no *back-end*, protegendo os pedidos HTTP com a verificação de os utilizadores possuírem a autorização necessária para realizar os pedidos.

4.1.1 UC1 – Efetuar pedido de registo na aplicação

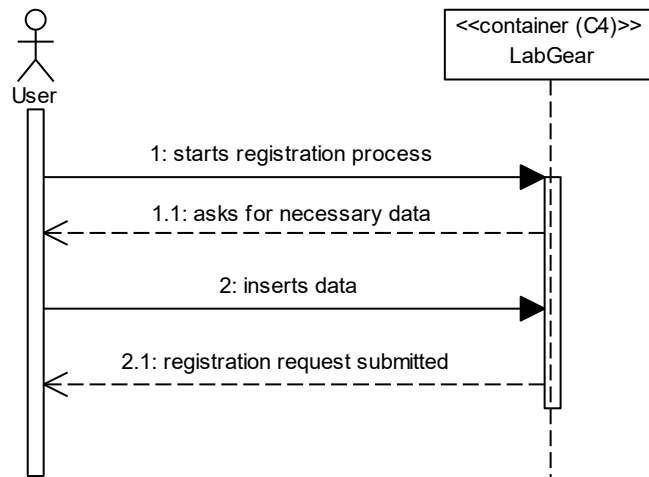


Figura 19 - SSD UC1

Neste caso de uso, um utilizador não registado pretende efetuar um pedido de registo na aplicação para conseguir aceder às suas funcionalidades.

Ao iniciar a aplicação, sem sessão iniciada, o utilizador é direcionado para a página de login. Como não tem conta, o utilizador pressiona a opção para se registar na aplicação, como mostra a figura 22. É pedido ao utilizador para introduzir o seu email, nome, password e função no CESAE (figura 20).

Depois de preenchidos os dados, é verificado se o email introduzido já existe no sistema e o pedido de registo é introduzido na base de dados, ficando à espera de ser respondido.

02:02 61%

ment.vercel.app

Registar

Email*
Ex. email@example.com

Campo obrigatório

Password*
.....

Confirm Password*
.....

Nome*

Função
Formador

Registar

Já tem conta? Login.

Figura 20 - Página de registo versão mobile

4.1.2 UC2 – Autenticar na aplicação

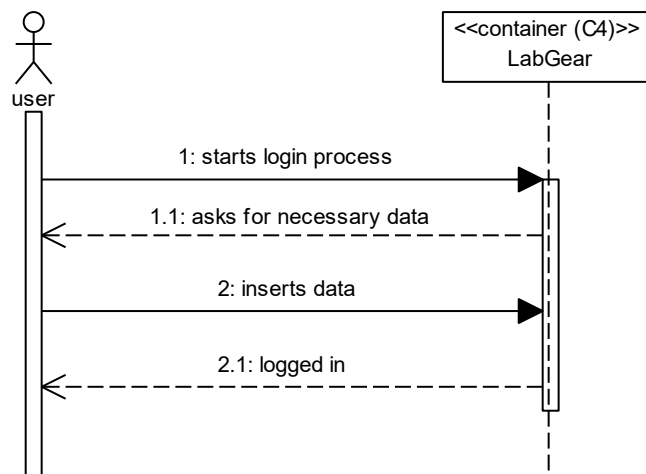


Figura 21 - SSD UC2

Neste caso de uso o utilizador pretende autenticar-se na aplicação através da combinação email e password.

Ao iniciar a aplicação sem sessão ativa, o utilizador é automaticamente direcionado para a página de login.

Figura 22 - Página de login

Depois de introduzir as suas credenciais, é validado se existe um utilizador com o email introduzido na base de dados e, caso exista, é verificada a password. Seguidamente é gerado um JWT (*Json Web Token*) com validade de 60 minutos em que é guardada a função do utilizador. O JWT e os detalhes do utilizador em questão são retornados ao *front-end* com a resposta de sucesso e o utilizador entra na aplicação, tendo acesso a um menu diferente dependendo da sua função.

4.1.3 UC3 – Responder a pedido de registo

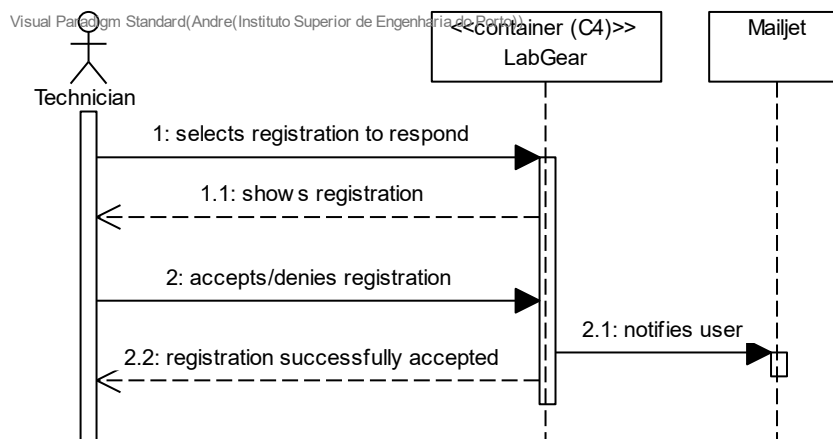


Figura 23 - SSD UC3

A funcionalidade de responder a pedidos de registo encontra-se disponível para o administrador e os técnicos. Expandindo a aba de “Utilizadores”, ao seleccionar “Gerir

Utilizadores”, é possível visualizar a lista de pedidos de registo pendentes e responder aos pedidos, como mostra a figura 24.

Dashboard

Home

Equipamentos

Requisições

Utilizadores

Adicionar Utilizador

Gerir Utilizadores

Hello André

Todos

Filtrar

UTILIZADORES PENDENTES

Email	Função	Nome	Notas	Ações
colaborator.test.cesae@gmail.com	colaborator	André		✓ ✕

UTILIZADORES ATIVOS

Email	Nome	Função	Notas	Ações
asilva200499@gmail.com	André	Administrador	Observations	
asilva2004999@gmail.com	name	Técnico	Observations	Desativar conta
asilva20049999@gmail.com	k	Colaborador		Desativar conta

UTILIZADORES INATIVOS

Figura 24 - Página gestão de utilizadores

Ao aceitar um pedido de registo, o utilizador fica a poder autenticar-se na aplicação. O utilizador recebe também um email a ser informado da sua aceitação na app.

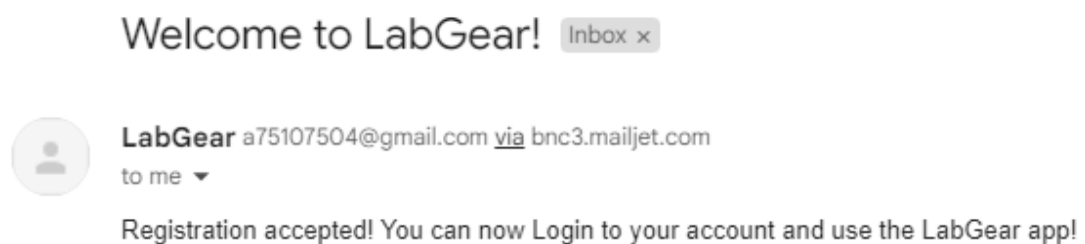


Figura 25 - Email de aceitação de registo

4.1.4 UC4 – Gerir utilizadores

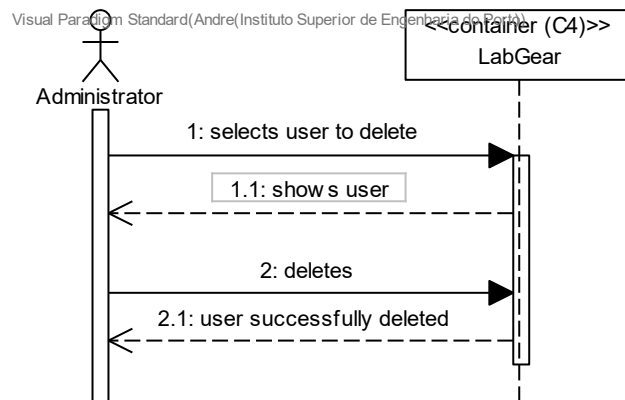


Figura 26 - SSD UC4 - delete user

Navegando à aba Utilizadores -> Adicionar Utilizador, o administrador consegue criar uma conta com função de técnico ou administrador (figura 27).

O administrador consegue ainda desativar as contas de utilizadores que não possuam a função de administrador como mostra a figura 24.

Os técnicos apenas possuem permissões para visualizar a lista de utilizadores e os seus detalhes, não conseguindo criar utilizadores ou desativar as suas contas. Ao listar os utilizadores, é possível filtrá-los através da sua função.

Figura 27 - Página adicionar utilizador

4.1.5 UC5 – Criar categoria de equipamento

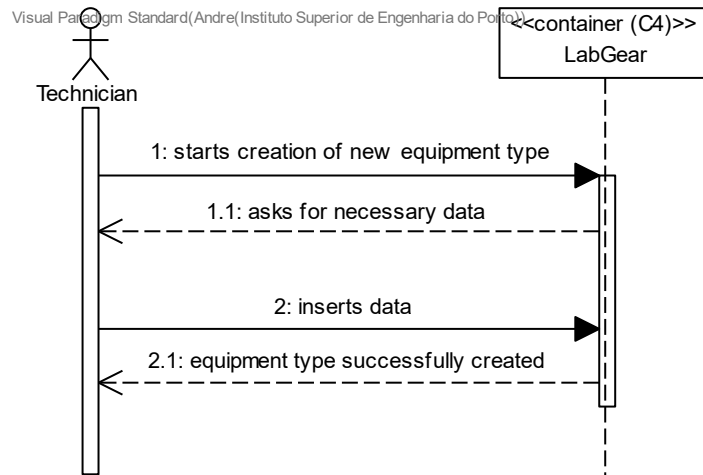


Figura 28 - SSD UC5

Neste caso de uso, o técnico pretende criar uma categoria de equipamento. Começa por dirigir-se à aba Equipamentos -> Adicionar Equipamento. Ao adicionar um equipamento é necessário selecionar a sua categoria. Caso a categoria pretendida ainda não exista no sistema, o técnico seleciona a opção “Nova categoria”, presente na figura 29.

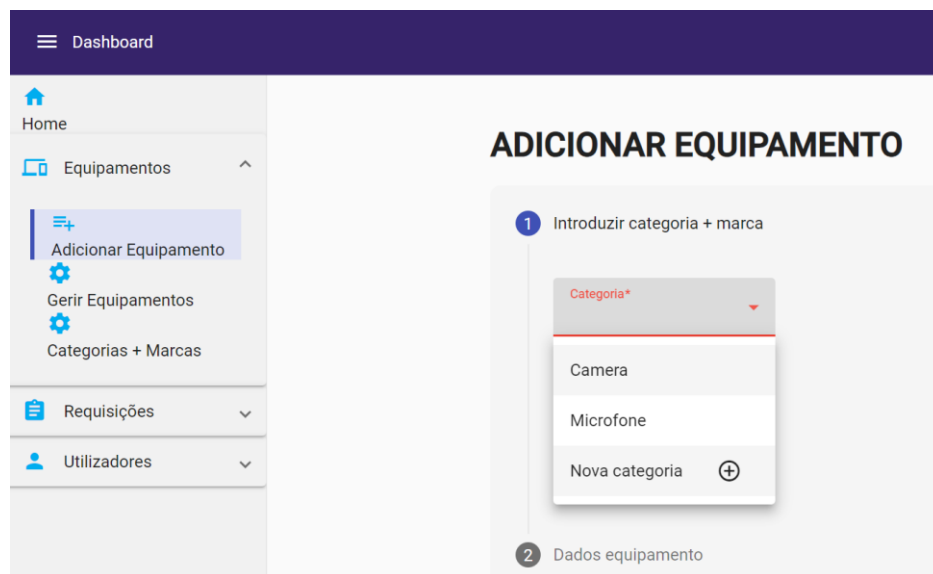


Figura 29 - Página adicionar equipamento, selecionar categoria

O técnico é, então, direcionado para a página de adicionar categoria onde é pedido para introduzir o nome da categoria e o stock mínimo para esse tipo de equipamento (figura 30).

manage-lab-equipment.vercel.app says

Categoria de equipamentos adicionada com sucesso!

OK

EQUIPAMENTO CATEGORIA

Descrição:

Stock mínimo:

'0' para não definir stock mínimo

[go back](#) [Adicionar](#)

Figura 30 - Página adicionar categoria de equipamento

4.1.6 UC6 - Adicionar equipamento

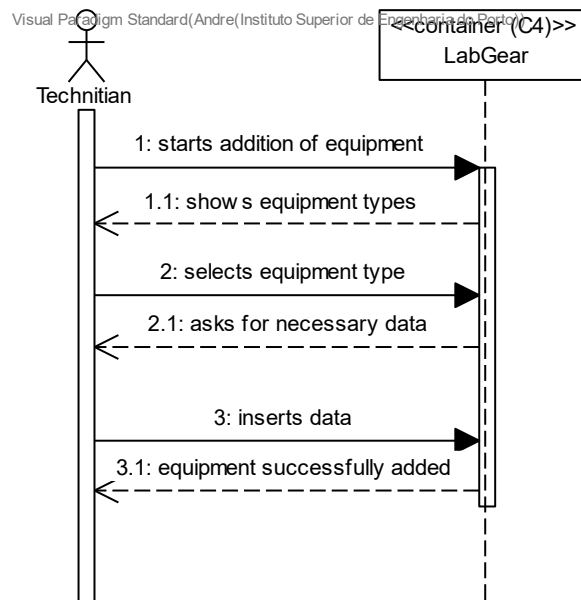


Figura 31 - SSD UC6

Para adicionar um equipamento no sistema, o técnico começa por seleccionar a categoria do equipamento e a marca do mesmo (figura 29). Caso a marca não se encontre registada na aplicação, é necessário escrevê-la.

Seguidamente são pedidos os dados referentes ao equipamento: descrição, modelo, referência, número de série e observações como se pode ver pela figura 32.

2 Dados equipamento

Descrição*

Modelo*

Referência*

Número de série

Comentário

Back Next

Figura 32 - Adicionar equipamento (dados equipamento)

Finalmente, é pedido ao técnico para confirmar (figura 33).

Figura 33 - Adicionar equipamento (confirmação)

É feito o pedido ao *back-end* para adicionar o equipamento, caso o modelo ou a marca introduzida não existam no sistema, são primeiro adicionados. Finalmente, é registado o equipamento na aplicação com os seus dados.

4.1.7 UC7 - Listar equipamentos existentes

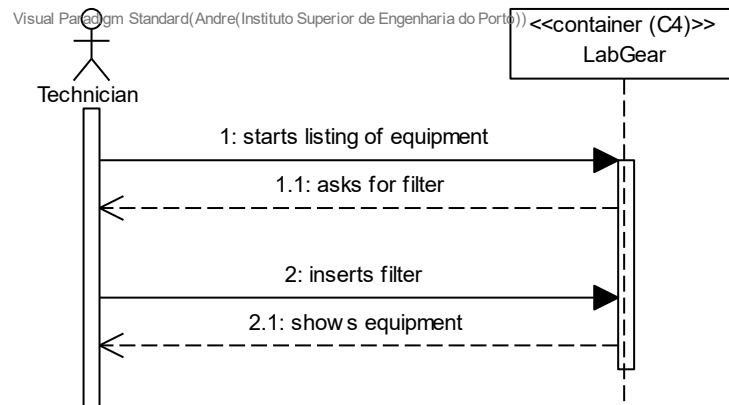


Figura 34 - SSD UC7

Através da aba Gerir Equipamentos, o técnico consegue ver a lista de todos os equipamentos existentes no sistema. Os equipamentos encontram-se divididos em equipamentos requisitáveis e equipamentos não requisitáveis. É possível filtrar a lista através da categoria e marca dos equipamentos (filtragem demonstrada na figura 35).

EQUIPAMENTOS REQUISITÁVEIS

Elgato

Camera

Filtrar

Camera Elgato Premium 1080p60

ID: 8afee03c
Marca: Elgato
Modelo: Premium 1080p60
Categoria: Camera
Disponível: Em stock

Bloquear

EQUIPAMENTOS NÃO REQUISITÁVEIS

Figura 35 - Página gerir equipamentos

4.1.8 UC8 – Editar equipamento

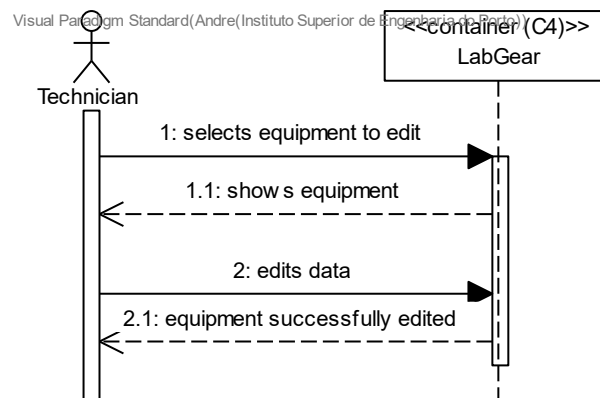



Figura 36 - SSD UC8

Ao selecionar um equipamento o técnico é redirecionado para uma página com informação desse equipamento. Seguidamente pressiona o botão de editar e torna-se possível alterar os campos editáveis (figura 37).

EQUIPAMENTO CAMERA ELGATO PREMIUM 1080P60 DETALHES



ID: 8afee03c

Marca: Elgato

Modelo: Premium 1080p60

Referência: 23asdas433a

Número de série:

Disponível: Em stock

Equipamento descrição:

Camera Elgato Premium 1080p60

Categoria

Camera

Comentário

Requisitável

save

Figura 37 - Página com informação de um equipamento

Quando o técnico submete as alterações, os dados inseridos são validados e, não ocorrendo erros, as alterações são persistidas na base de dados.

4.1.9 UC9 - Eliminar equipamento

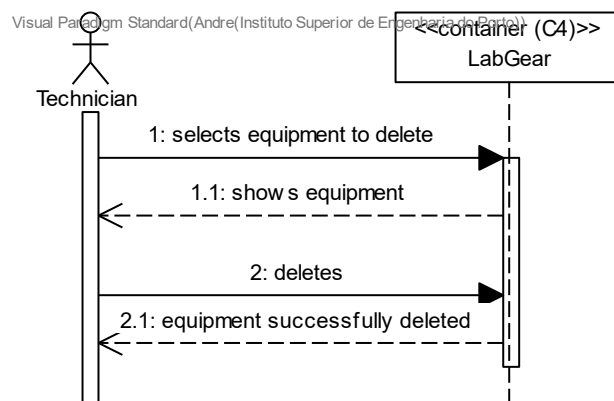


Figura 38 - SSD UC9

Neste caso de uso, o técnico pretende eliminar um equipamento do sistema. Apenas é possível eliminar equipamentos que não se encontrem ativos, ou seja, não requisitáveis. De notar que equipamentos com requisições ativas ou concluídas não podem ser eliminados, lançando uma exceção proveniente do *back-end*. Ao pressionar o botão de remover, o equipamento é eliminado da base de dados (figura 39).

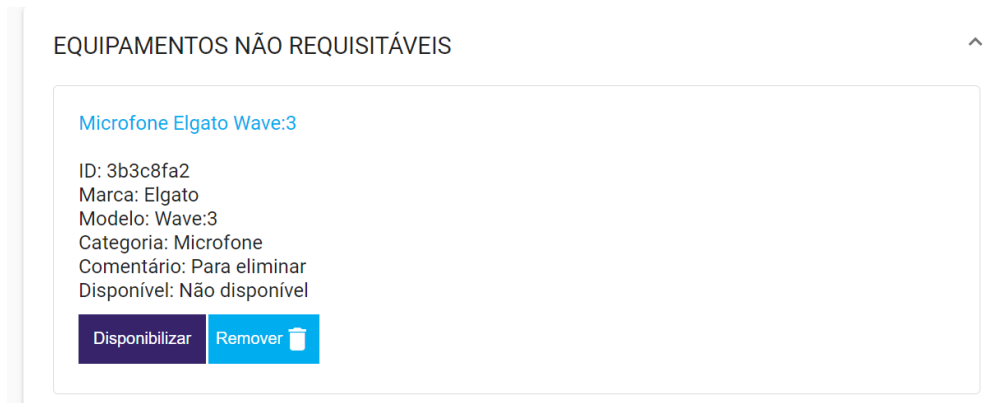


Figura 39 - Remover equipamento

4.1.10 UC10 - Efetuar requisição de equipamentos

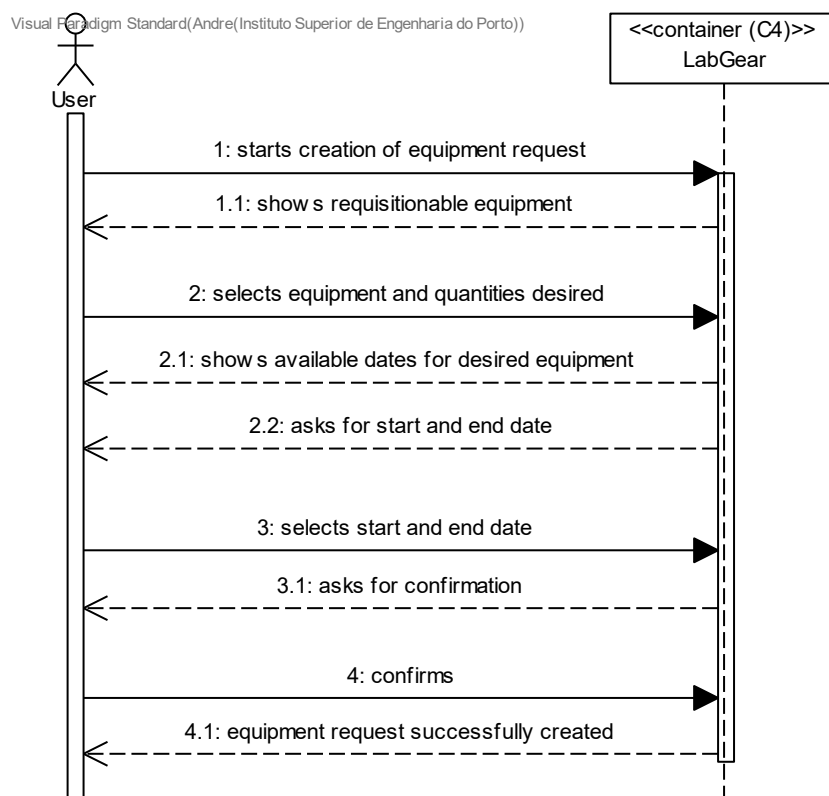


Figura 40 - SSD UC10

O caso de uso de efetuar requisição de equipamentos trata-se da funcionalidade principal da aplicação. O utilizador começa por dirigir-se à aba Requisições -> Criar requisição. É lhe mostrado uma lista dos equipamentos requisitáveis à qual podem ser aplicados filtros de marca e/ou categoria de equipamento (figura 41). O utilizador seleciona os equipamentos que pretende requisitar e avança para a escolha de datas.

CRIAR REQUISIÇÃO

1 Selecionar equipamentos

2 Escolher datas

3 Dados requisição

4 Confirmar

Marca ▼

Categoria ▼

FILTRAR

Microfone Elgato Wave:3

ID: 223bfbe5
Marca: Elgato
Modelo: Wave:3
Categoria: Microfone

ADICIONAR

Camera Elgato Premium 1080p60

ID: 8afee03c
Marca: Elgato
Modelo: Premium 1080p60
Categoria: Camera

ADICIONAR

Equipamentos Selecionados

Microfone Elgato 150px High res : 3f0236a8

REMOVER

NEXT

Figura 41 - Página criar requisição (escolher equipamentos)

Como descrito do diagrama da figura 16, ao avançar para a escolha de datas, a lista de equipamentos é enviada ao *back-end*. No *back-end* é feita uma busca à base de dados por requisições ativas (estado aceite, levantado ou atrasado) que contenham algum dos equipamentos e são retornadas as datas em que os equipamentos se encontrem ocupados. É apresentado ao utilizador um calendário para selecionar a data de início e data de fim pretendidas para a requisição com as datas em que os equipamentos se encontram indisponíveis bloqueadas (figura 42).

Figura 42 - Página criar requisição (escolha de datas)

Ao avançar, é pedido ao utilizador dados relativos à requisição como a designação do curso, a turma a que pertence e o nome da UFCD e por fim é pedida a confirmação. No seguimento da confirmação é criado um pedido de requisição no sistema ficando no estado “Pendente”, à espera de ser aceite ou recusado.

4.1.11 UC11 - Listar requisições de um utilizador

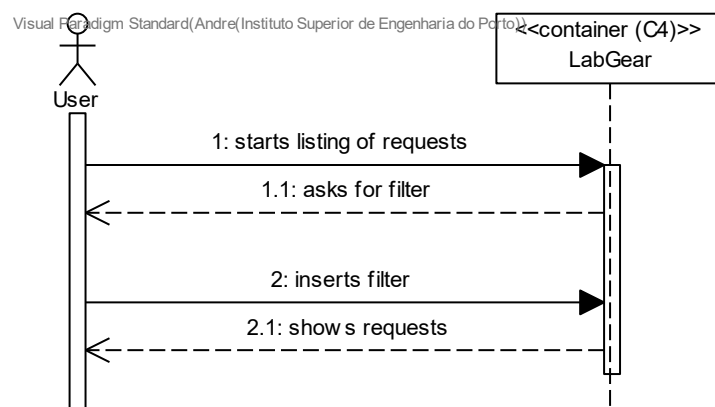


Figura 43 - SSD UC11

Neste caso de uso um utilizador pretende visualizar as suas próprias requisições. Ao seleccionar a aba “Minhas Requisições”, é feito um pedido ao *back-end* que responde com a lista de requisições do utilizador com sessão iniciada agrupadas pelo estado da requisição, como mostra a figura 44.

Dashboard

Home

Equipamentos

Requisições

Criar Requisição

Minhas Requisições

Gerir Requisições

Utilizadores

REQUISICOES PENDENTES

ID	Estado	Requisitante	Técnico	Data início	Data fim	Comentário	Ações
793d317d	Cancelado	asilva200499@gmail.com		28/9/2023	28/9/2023	Requisition cancelled due to overlapping equipment with another requisition. Create a new requisition with different dates or equipment or Update this requisition.	
f70ef214	Pendente	asilva200499@gmail.com		8/9/2023	9/9/2023		

REQUISICOES ATIVAS

ID	Estado	Requisitante	Técnico	Data início	Data fim	Comentário
121d4d12	Aceite	asilva200499@gmail.com	asilva200499@gmail.com	19/9/2023	25/9/2023	
456a0815	Aceite	asilva200499@gmail.com	asilva200499@gmail.com	28/9/2023	28/9/2023	

REQUISICOES CONCLUIDAS

Figura 44 - Página que mostra as requisições do utilizador com sessão iniciada

4.1.12 UC12 - Editar requisição

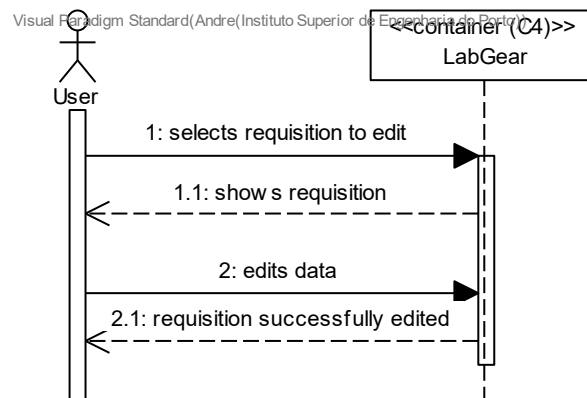


Figura 45 - SSD UC12

Neste caso de uso, um técnico, ou o utilizador que efetuou a requisição, conseguem editar uma requisição, sendo que só é possível alterar a lista de equipamentos e as datas de início e de fim caso a requisição se encontre no estado pendente ou cancelado. A edição da lista de equipamentos e das datas seguem a mesma metodologia que na criação da requisição (figuras 41 e 42).

4.1.13 UC13 – Responder a pedido de requisição

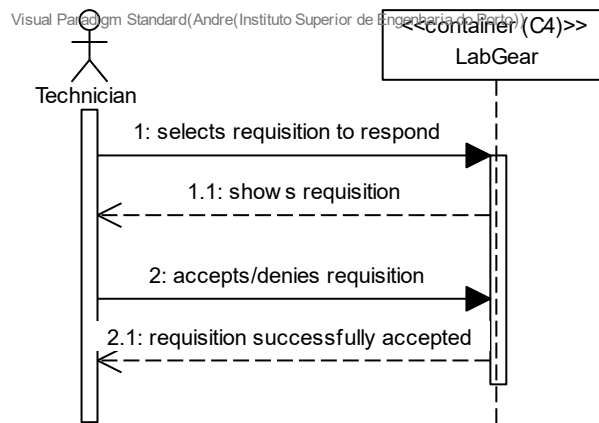


Figura 46 - SSD UC13

Para responder a um pedido de requisição, o técnico navega até à aba “Gerir Requisições”. São listadas as requisições pendentes com as opções de aceitar ou recusar (figura 47). Caso o técnico pretenda ver os detalhes da requisição basta clicar nela e é redirecionado para a página com toda a informação sobre a requisição.

ID	Estado	Requisitante	Técnico	Data início	Data fim	Comentário	Ações
30d2b2e9	Pendente	asilva200499@gmail.com		28/9/2023	28/9/2023		✓ ✕
320208dc	Pendente	asilva200499@gmail.com		28/9/2023	28/9/2023		✓ ✕
f70ef214	Pendente	asilva200499@gmail.com		8/9/2023	9/9/2023		✓ ✕

ID	Estado	Requisitante	Técnico	Data início	Data fim	Comentário	Ações
121d4d12	Aceite	asilva200499@gmail.com	asilva200499@gmail.com	19/9/2023	25/9/2023		Levantada

Figura 47 - Página gerir requisições

Caso existam duas requisições pendentes que utilizam um equipamento em comum e com sobreposição de datas, não é possível aceitar ambas as requisições. Nesse cenário, caso o técnico aceite uma das requisições, a segunda passa automaticamente para o estado cancelado (figura 48). No estado cancelado, uma requisição pode ser editada pelo técnico ou pelo utilizador que efetuou a requisição para efetuar um novo pedido com equipamentos e datas válidas.

REQUISIÇÕES CANCELADAS

ID	Estado	Requisitante	Técnico	Data início	Data fim	Comentário	Ações
320208dc	Cancelado	asilva200499@gmail.com		28/9/2023	28/9/2023	Requisition cancelled due to overlapping equipment with another requisition. Create a new requisition with different dates or equipment or Update this requisition.	

Figura 48 - Página gerir requisições (requisições canceladas)

4.1.14 UC15 - Listar requisições existentes

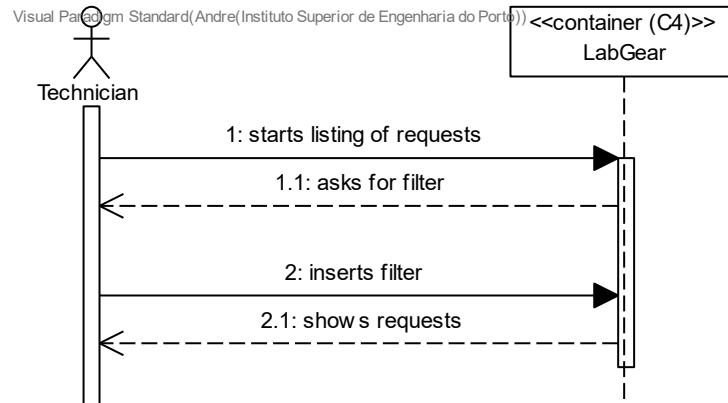


Figura 49 - SSD UC15

O técnico pode listar requisições existentes navegando até à aba “Gerir Requisições”. As requisições encontram-se agrupadas pelo seu estado e é possível filtrar as requisições pelo utilizador requisitante, como mostra a figura 47, ou por equipamento, em que são retornadas da base de dados todas as requisições que continham o equipamento em questão.

4.1.15 UC16 – Notificar utilizador de mudanças no estado da requisição

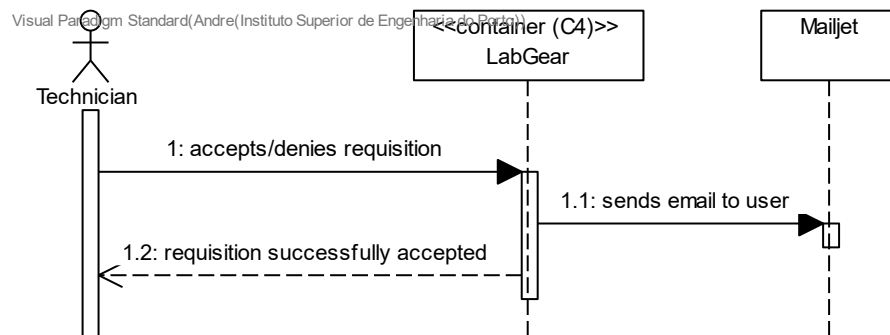


Figura 50 - SSD UC16

Quando ocorrem mudanças no estado de uma requisição, o utilizador é avisado por email dessa ocorrência (figura 51).



Figura 51 - Email requisição cancelada

4.1.16 UC17/UC18 – Definir stock mínimo para categoria de equipamento e notificar em caso de stock reduzido

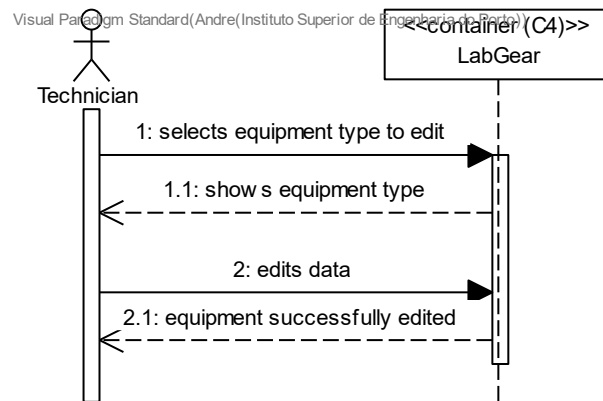


Figura 52 - SSD UC17

Neste caso de uso um técnico ou administrador definido no sistema vai receber alertas de stock reduzido. Para definir o stock mínimo de uma categoria de equipamento o técnico navega até à aba “Categorias + Marcas” e seleciona a categoria de equipamento pretendida para editar o stock mínimo para essa categoria (figura 53). Também é possível observar a quantidade de stock disponível.

Figura 53 - Página detalhes de categoria de equipamento

O stock disponível é atualizado automaticamente e, quando uma requisição é levantada, os equipamentos deixam de estar em stock. Caso o stock disponível de uma categoria de equipamento seja inferior ao stock mínimo dessa categoria, vai ser lançado um alerta para avisar desse facto como mostra a figura 54.

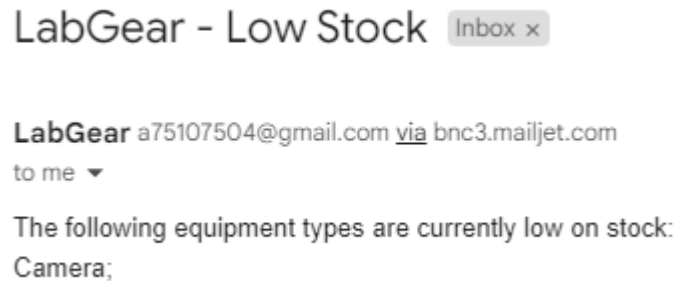


Figura 54 - Email de alerta de stock reduzido

4.1.17 UC19 – Alterar estado da requisição

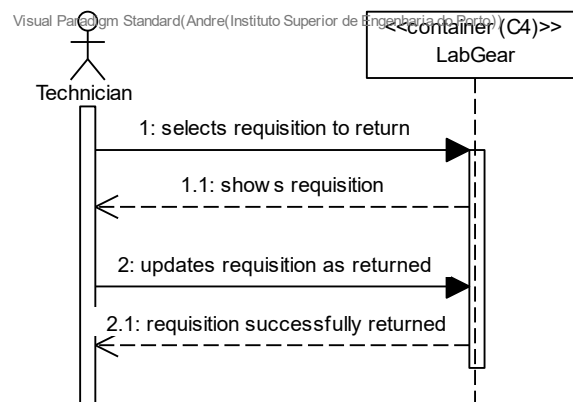


Figura 55 - SSD UC19

Como é demonstrado na figura 47, na página de gerir requisições, um técnico consegue alterar o estado das requisições ativas, podendo alterar o estado da requisição para levantado e, posteriormente, devolvido.

4.1.18 UC20 – Notificar atrasos na devolução da requisição

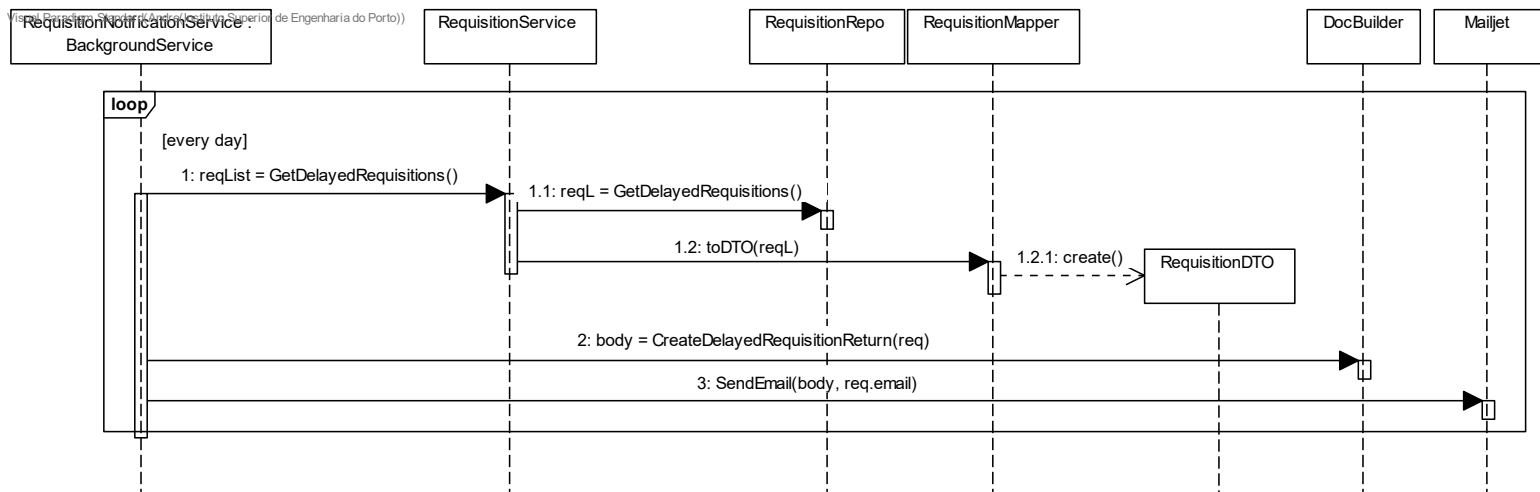


Figura 56 - Vista lógica nível 3 UC20

Neste caso de uso, caso a devolução de uma requisição se encontre atrasada, isto é, a data de fim da requisição já passou, o estado da requisição é alterado para atrasado e o utilizador requisitante e o técnico responsável são notificados por email do atraso.

Esta funcionalidade é conseguida através de um serviço executado diariamente no *back-end* que chama um método para verificar se existem requisições atrasadas (figura 57).

```

public async Task<List<RequisitionDto>> UpdateRequisitionDelayedAsync()
{
    List<Requisition> pickedUpRequisitions = await _repo.GetRequisitionByStatusAsync("picked_up");

    List<Requisition> toDelayedRequisitions = pickedUpRequisitions.Where(r => r.EndDate > DateTime.Now).ToList();

    foreach (Requisition requisition in toDelayedRequisitions)
    {
        requisition.DelayedRequisition();
        await this._emailSender.SendEmailRequisitionDelayed(requisition.Requisitioner.Value, requisition.Id.AsString().Substring(0, 8));
        await this._emailSender.SendEmailRequisitionDelayed(requisition.TechnicianAdmin.Value, requisition.Id.AsString().Substring(0, 8));
    }

    await this._unitOfWork.CommitAsync();

    return toDelayedRequisitions.Select(r => RequisitionMapper.ToDto(r)).ToList();
}
  
```

Figura 57 - Método para verificar atrasos nas requisições

Este método vai buscar à base de dados todas as requisições levantadas e compara a data de fim de cada uma dessas requisições com a data atual. Caso a devolução esteja atrasada, a requisição passa para o estado atrasado são enviadas notificações por email para alertar do atraso.

4.1.19 Bibliotecas e APIs relevantes

4.1.19.1 Angular Material

A biblioteca mais usada no *front-end* para criar os elementos e organizar os *templates* da UI foi a Angular Material, desde o menu de navegação ao calendário para selecionar datas, os componentes do Angular Material foram usados em grande parte na criação da UI.

4.1.19.2 MailJet

Para enviar as notificações foi usada a plataforma MailJet através de chamadas à sua API (figura 58).

```
public virtual async Task<bool> SendEmailRegistrationAcceptedAsync(string recipientEmail, string recipientFirstName)
{
    var client = new MailjetClient(_smtpSettings.UserName, _smtpSettings.Password);

    JArray jArray = EmailRegistrationAccept(recipientEmail, recipientFirstName, _smtpSettings);

    MailjetRequest request = new MailjetRequest
    {
        Resource = Send.Resource,
    }
    .Property(Send.Messages, jArray);

    MailjetResponse response = await client.PostAsync(request);
    if (response.IsSuccessStatusCode)
    {
        Console.WriteLine(string.Format("Total: {0}, Count: {1}\n", response.GetTotal(), response.GetCount()));
        Console.WriteLine(response.GetData());
        return true;
    }
    else
    {
        Console.WriteLine(string.Format("StatusCode: {0}\n", response.StatusCode));
        Console.WriteLine(string.Format("ErrorInfo: {0}\n", response.GetErrorInfo()));
        Console.WriteLine(response.GetData());
        Console.WriteLine(string.Format("ErrorMessage: {0}\n", response.GetErrorMessage()));
        return false;
    }
}
```

Figura 58 - Método para enviar emails usando MailJet

É criado um cliente MailJet com a chave da API e a chave secreta. De seguida é gerado o email como um JArray que contem o conteúdo do email, o destinatário e outros detalhes. É criado um pedido para enviar o email e é chamada a API do MailJet com o pedido criado.

4.2 Testes

Para assegurar o bom funcionamento e prevenção de erros na *web app*, foram criados testes para cada funcionalidade da aplicação. Foram criados testes unitários e testes de integração tanto no *front-end* como no *back-end*, tendo sido também realizados testes à API através do Postman. Para testar o *front-end* foram usadas as bibliotecas Jasmine e a biblioteca de testes do Angular, enquanto no *back-end* foi utilizado o xUnit.net.

4.2.1 Testes unitários

Com o objetivo de verificar o comportamento de componentes e métodos da aplicação de forma isolada foram criados diversos testes unitários. Ao realizar testes unitários é possível testar os componentes sem qualquer dependência. [44]

Na figura 59 está presente um exemplo de um teste unitário no *front-end*. Trata-se de um teste ao método de criar uma requisição da classe *RequisitionService*. Primeiramente é criado um DTO com a requisição que se pretende criar e uma requisição *dummy* que vai ser retornada ao criar a requisição. É usado um *mock* para definir o retorno do pedido que seria enviado ao *back-end*. É chamada o método *addRequisition* e verificado se o retorno é o esperado.

```
describe('#addRequisition', () => {
  it('should return an Observable<RequisitionDto>', () => {

    const requisition =
      {
        course: 'course', class_str: 'class', ufcd: 'ufcd', comment: 'comment', startYear: 2024, startMonth: 1, startDay: 22, endYear: 2024,
        endMonth: 2, endDay: 1, requisitioner: 'email@gmail.com', equipmentList: []
      } as RequisitionDtoCreate;

    const dummyRequisition =
      {
        id: 'id', course: 'course', class_str: 'class', ufcd: 'ufcd', comment: 'comment', startYear: 2024, startMonth: 1, startDay: 22, endYear: 2024,
        endMonth: 2, endDay: 1, status: "pending", requisitioner: 'email@gmail.com', technicianAdmin: 'tech@gmail.com', equipmentList: []
      } as RequisitionDto;

    service.addRequisition(requisition).subscribe(requisition => {
      expect(requisition).toEqual(dummyRequisition);
    });

    // The following `expectOne()` will match the request's URL.
    // If no requests or multiple requests matched that URL
    // `expectOne()` would throw.
    const request = httpMock.expectOne(URL);
    expect(request.request.method).toBe('POST');

    // Respond with mock data, causing Observable to resolve.
    // Subscribe callback asserts that correct data was returned.
    request.flush(dummyRequisition);
  });
});
```

Figura 59 - Teste unitário front-end

Na figura 60 está presente um exemplo de um teste unitário no *back-end*. Este é um teste ao método *AddAsync* da classe *RequisitionService* em que é esperado ser lançado uma exceção ao tentar criar uma requisição. Inicialmente são criadas várias entidades e DTOs para definir os retornos dos métodos que são chamados nesta função do serviço. De seguida esses retornos são definidos usando múltiplos *mocks*. A porção de código destacada representa o *mock* ao método para verificar se os equipamentos da requisição se encontram disponíveis no período pretendido. Os equipamentos encontram-se disponíveis se não for retornada nenhuma requisição, no entanto, o *mock* retorna uma lista de requisições com um elemento. É chamado o método a testar e verificado se é lançada a exceção esperada.

```

[Fact]
public async Task AddAsync_UnavailableEquipment()
{
    // Arrange
    List<Equipment> equipmentList = new List<Equipment>();
    var equipment = new Equipment("ref", "desc", "serial", "comment", new EquipmentModel("model", new EquipmentBrand("brand")), new EquipmentType("type", 0));
    string equipmentId = equipment.Id.Value;
    equipmentList.Add(equipment);
    List<string> stringList = new List<string>();
    stringList.Add(equipmentId);

    Guid id = Guid.NewGuid();
    var dto = new RequisitionDtoCreate(id, "java", "2B", "JLAC", "comment", 2024, 1, 15, 2024, 1, 20, "email@gmail.com", "", stringList);

    User requisitioner = new User("email@gmail.com", "Password1", "name", "999999999", "obs", 2000, 10, 18, "technician");

    List<Requisition> requisitionList = new List<Requisition>();
    Requisition requisition = new Requisition(
        "Course", "Class", "Ufcd",
        2024, 1, 1,
        2024, 6, 30,
        new UserEmail("test@example.com")
    );
    requisitionList.Add(requisition);

    AppConfig config = new AppConfig("string", "asilva200499@gmail.com", 0);

    _equipmentRepo.Setup(x => x.GetEquipmentListAsync(It.IsAny<List<EquipmentId>>()))
        .ReturnsAsync(equipmentList);

    _userRepo.Setup(x => x.GetByIdAsync(It.IsAny<UserEmail>())).ReturnsAsync(requisitioner);

    _repo.Setup(x => x.GetRequisitionBetweenDatesWithEquipmentAsync(It.IsAny<DateTime>(), It.IsAny<DateTime>(), It.IsAny<List<EquipmentId>>()))
        .Returns(Task.FromResult(requisitionList));

    _appConfigRepo.Setup(x => x.GetLast()).ReturnsAsync(config);

    _emailSender.Setup(x => x.SendEmailNewRequisition(It.IsAny<string>(), It.IsAny<string>())).ReturnsAsync(true);

    // Act
    var exception = await Assert.ThrowsAsync<BusinessRuleValidationException>(() =>
        _service.AddAsync(dto));

    // Assert
    Assert.Contains("Equipment not available in this period", exception.Message);
}

```

Figura 60 - Teste unitário back-end

Na tabela 3 encontram-se presentes alguns dos testes unitários para a funcionalidade de criar uma requisição.

Tabela 3 - Testes unitários para a criação de uma requisição

Teste	Descrição
Constructor_WithValidArgs: domain	Teste de sucesso na criação de uma requisição ao nível do domínio
Constructor_WithEmptyStrings: domain	Teste de sucesso na criação de uma requisição com <i>strings</i> vazias nos argumentos <i>nullable</i>
Constructor_WithInvalidStringLength: domain	Teste de insucesso na criação de uma requisição ao ter argumentos com <i>strings</i> demasiado extensas, lança exceção

Constructor_WithInvalidDates1: domain	Teste de insucesso na criação de uma requisição ao possuir uma data de início de requisição posterior à data de fim, lança exceção
Constructor_WithInvalidDates2: domain	Teste de insucesso na criação de uma requisição ao possuir uma data de início de requisição posterior à data atual, lança exceção
Constructor_WithInvalidDates3: domain	Teste de insucesso na criação de uma requisição ao possuir uma duração superior a um ano, lança exceção
Constructor_WithInvalidDates4: domain	Teste de insucesso na criação de uma requisição ao possuir uma data de início a faltar mais de 6 meses da data atual, lança exceção
AddAsync_AddsRequisition: service	Teste de sucesso na criação de uma requisição, retorna DTO com os dados da requisição criada
AddAsync_NoEquipment: service	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição com equipamentos que não existem no sistema, lança exceção
AddAsync_UserDoesNotExist: service	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição em que o email do requisitante não foi encontrado no sistema, lança exceção
AddAsync_UnavailableEquipment: service	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição com equipamentos indisponíveis para o intervalo de datas selecionado, lança exceção

4.2.2 Testes de integração

Para testar as funcionalidades garantindo que diferentes componentes interagem da forma pretendida foram realizados testes de integração.

Na figura 61 encontra-se um exemplo de um teste de integração no *back-end* entre o *controller* e o *service* das requisições para criar uma requisição. Inicialmente são criadas várias entidades e DTOs para definir os retornos dos métodos que são chamados nesta interação entre o *RequisitionController* e o *RequisitionService*. De seguida esses retornos são definidos usando

múltiplos *mocks*. Finalmente, é chamado o método do *controller* a testar e verificado se o retorno é o esperado.

```
[Fact]
public async Task TestCreateRequisition()
{
    List<Equipment> equipmentList = new List<Equipment>();
    var equipment = new Equipment("ref", "desc", "serial", "comment", new EquipmentModel("model", new EquipmentBrand("brand")),
        new EquipmentType("type", 0));
    string equipmentId = equipment.Id.Value;
    equipmentList.Add(equipment);
    List<string> stringList = new List<string>();
    stringList.Add(equipmentId);

    Guid id = Guid.NewGuid();
    var dto = new RequisitionDtoCreate(id, "java", "2B", "JLAC", "comment", 2024, 1, 15, 2024, 1, 20, "email@gmail.com", "", stringList);

    User requisitioner = new User("email@gmail.com", "Password1", "name", "999999999", "obs", 2000, 10, 18, "technician");

    List<Requisition> requisitionList = new List<Requisition>();

    AppConfig config = new AppConfig("string", "asilva200499@gmail.com", 0);

    _equipmentRepo.Setup(x => x.GetEquipmentListAsync(It.IsAny<List<EquipmentId>>()))
        .ReturnsAsync(equipmentList);

    _userRepo.Setup(x => x.GetByIdAsync(It.IsAny<UserEmail>())).ReturnsAsync(requisitioner);

    _repo.Setup(x => x.GetRequisitionBetweenDatesWithEquipmentAsync(It.IsAny<DateTime>(), It.IsAny<DateTime>(), It.IsAny<List<EquipmentId>>()))
        .Returns(Task.FromResult(requisitionList));

    _appConfigRepo.Setup(x => x.GetLast()).ReturnsAsync(config);

    _emailSender.Setup(x => x.SendEmailNewRequisition(It.IsAny<string>(), It.IsAny<string>())).ReturnsAsync(true);

    var result = await _controller.Create(dto);

    Assert.NotNull(result);
    Assert.Equal(dto.Course, result.Result.Value.Course);
    Assert.Equal(dto.Class_str, result.Result.Value.Class_str);
    Assert.Equal(dto.Ufcd, result.Result.Value.Ufcd);
    Assert.Equal(dto.StartYear, result.Result.Value.StartYear);
    Assert.Equal(dto.StartMonth, result.Result.Value.StartMonth);
    Assert.Equal(dto.StartDay, result.Result.Value.StartDay);
    Assert.Equal(dto.EndYear, result.Result.Value.EndYear);
    Assert.Equal(dto.EndMonth, result.Result.Value.EndMonth);
    Assert.Equal(dto.EndDay, result.Result.Value.EndDay);
    Assert.Equal(dto.Morada, result.Result.Value.Morada);
    Assert.Equal(dto.Requisitioner, result.Result.Value.Requisitioner);
    Assert.Equal(dto.EquipmentList, result.Result.Value.EquipmentList);
}
```

Figura 61 - Teste de integração back-end

Na tabela 4 encontram-se presentes alguns dos testes de integração para a funcionalidade de criar uma requisição.

Tabela 4 - Testes de integração para criação de uma requisição

Teste	Descrição
TestCreateRequisition	Teste de sucesso na criação de uma requisição, retorna DTO com os dados da requisição criada + sucesso 201
TestCreateRequisition_NoEquipment	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição

	com equipamentos que não existem no sistema, retorna exceção + erro 400
TestCreateRequisition_UserDoesNotExist	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição em que o email do requisitante não foi existe no sistema, retorna exceção + erro 400
TestCreateRequisition_UnavailableEquipment	Teste de insucesso na criação de uma requisição ao tentar criar uma requisição com equipamentos indisponíveis para o intervalo de datas selecionado, retorna exceção + erro 400

4.2.3 Testes à API

Foram ainda realizados testes à API com a utilização do Postman para testar as funcionalidades implementadas. A figura 62 mostra um exemplo de um ficheiro com testes para criar uma requisição.

```

POST POST test user
POST POST test technician
POST POST test eq brand
POST POST test eq type
POST POST test equipment
GET GET by Id(expected error 500)
POST POST requisition
POST POST requisition w/ error
GET GET Req by Id
GET GET all Req
GET GET Req by eq
PUT PUT Req
GET GET Req by Id after update
DEL DELETE requisition
DEL DELETE soft eq test
DEL DELETE hard eq test
DEL DELETE test eq model
DEL DELETE test eq brand
DEL DELETE test eq type
DEL DELETE soft test user
DEL DELETE hard test user
DEL DELETE soft test user 2
DEL DELETE hard test user 2
GET GET Req by Id (expected 500)

```

Figura 62 - Testes à API para criar requisição

Primeiro são adicionados à base de dados entidades de teste para ser possível realizar uma requisição. De seguida é criada a requisição e verificada se persistiu na base de dados através de um pedido GET. De seguida os dados da requisição são editados e é verificado se as alterações persistiram como mostra a figura 63. Finalmente, as entidades de teste são eliminadas é a verificado que a requisição previamente criada já não existe.



```

1  pm.test("Status code is 200", function () {
2    |    pm.response.to.have.status(200);
3  });
4
5  const responseJson = pm.response.json();
6
7  pm.test("course check", function () {
8    |    pm.expect(responseJson.course).to.eql("course1");
9  });
10
11 pm.test("class_str check", function () {
12 |    pm.expect(responseJson.class_str).to.eql("class1");
13 });
14
15 pm.test("ufcd check", function () {
16 |    pm.expect(responseJson.ufcd).to.eql("ufcd1");
17 });
18
19 pm.test("comment check", function () {
20 |    pm.expect(responseJson.comment).to.eql("comment1");
21 });
  
```

Figura 63 - Testes no Postman para editar requisição

4.3 Avaliação da solução

Para realizar a avaliação da solução foram realizadas reuniões recorrentes com o Supervisor da empresa com o objetivo de verificar o desenvolvimento da aplicação e das suas funcionalidades. Em todas reuniões eram efetuados testes de aceitação com a demonstração das várias funcionalidades estabelecidas nos requisitos iniciais. O Supervisor, que era o cliente da aplicação, fornecia *feedback* que servia para a realização de correções e ajustes nas funcionalidades.

A solução desenvolvida responde a todos os requisitos funcionais requeridos com sucesso e respeita todos os requisitos não funcionais, tendo obtido uma resposta positiva do Supervisor da empresa e colaboradores que a pretendem usar.

5 Conclusões

Este capítulo consiste num resumo do trabalho desenvolvido através do enquadramento dos objetivos concretizados com os previstos, prossegue-se com a identificação das limitações do trabalho realizado e possíveis direções de desenvolvimento futuro e, finalmente, uma apreciação final sobre o desenvolvimento do projeto.

5.1 Objetivos concretizados

Na secção 1.2.1 do capítulo de introdução foram identificadas as principais funcionalidades a ser implementadas na aplicação, sendo elas:

- Registo e login de utilizadores;
- Adição e gestão de equipamentos;
- Criação e gestão de requisições de equipamentos;
- Atualização automática dos stocks de equipamento disponível;
- Envio de notificações por parte do sistema.

Todas as funcionalidades propostas foram implementadas através da realização dos casos de uso identificados na secção 3.2.1 do capítulo 3:

- Os casos de uso 1, 2 e 3 focam-se no registo e login de utilizadores;
- Os casos de uso 5, 6, 7, 8 e 9 dizem respeito à adição e gestão de equipamentos;
- Os casos de uso 10, 11, 12, 13, 14, 15 e 19 possibilitam a criação e gestão de requisições;
- A atualização dos stocks de equipamento disponível é feita durante o fluxo dos casos de uso 6, 8, 9 e 19.
- Os casos de uso 3, 10, 16, 18 e 20 tratam de enviar notificações por email

Para além das funcionalidades propostas inicialmente, foram ainda implementadas funcionalidades consideradas importantes para a aplicação como a gestão de utilizadores e criação de contas de técnico por parte do administrador no caso de uso 4 e definição de stock mínimo e alertas de stock reduzido para cada categoria de equipamento.

5.2 Limitações e trabalho futuro

A aplicação desenvolvida responde de maneira capaz a todos os requisitos propostos inicialmente. Existe, no entanto, sempre a possibilidade de expandir o projeto e adicionar novas funcionalidades com uso prático para os utilizadores da aplicação. Algumas adições interessantes para trazer mais valor à aplicação podiam ser:

- Sugestão automática de equipamentos da mesma categoria caso um equipamento estivesse indisponível na data pretendida;
- Adicionar imagem aos equipamentos;
- Colocar um aviso nos equipamentos que pertencem a categorias com stock disponível reduzido;
- Importar listas de equipamentos de ficheiros;
- Ordenar listas de equipamentos e requisições pelo campo pretendido.

5.3 Apreciação final

Finalizado o projeto, o balanço feito é bastante positivo. Tratou-se de um projeto extremamente desafiante no qual foi tudo criado de raiz. Foi necessário um grande esforço constante durante a realização do projeto e autonomia para conseguir realizar todas as funcionalidades pretendidas. A aplicação desenvolvida trata-se de uma solução funcional que responde ao problema inicial e funcionalidades requeridas encontrando-se pronta a ser utilizada pelos elementos do CESAE.

A nível pessoal evoluí profundamente os meus conhecimentos ao longo do estágio e sinto-me bastante mais confortável com o uso das tecnologias utilizadas neste projeto e desenvolvimento de aplicações em geral.

6 Referências

- [1] “Importance of Information Systems in an Organization.” <https://smallbusiness.chron.com/importance-information-systems-organization-69529.html> (accessed Sep. 10, 2023).
- [2] F. Lestari, U. Ulfah, F. R. Aprianis, and S. Suherman, “Inventory Management Information System in Blood Transfusion Unit,” *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, vol. 2019-December, pp. 268–272, Jan. 2018, doi: 10.1109/IEEM.2018.8607557.
- [3] “Unleash the Power of Inventory Management for Your DTC Brand.” <https://dotcomdist.com/benefits-of-effective-inventory-management/> (accessed Sep. 08, 2023).
- [4] “Requisition Management | Creating a Requisition Management System Online.” <https://kissflow.com/procurement/purchase-requisition/requisition-management-automation/> (accessed Sep. 07, 2023).
- [5] “Requisition management software -Automate manual purchasing process and control cost - ProcureDesk.” <https://www.procuredesk.com/requisition-management-software/> (accessed Sep. 08, 2023).
- [6] “What Is the Definition of Order Processing Systems?” <https://smallbusiness.chron.com/definition-order-processing-systems-3197.html> (accessed Sep. 10, 2023).
- [7] R. Bichachi, “What Is a Purchase Requisition? | NetSuite.” <https://www.netsuite.com/portal/resource/articles/accounting/purchase-requisition.shtml> (accessed Sep. 07, 2023).
- [8] “The Importance Of Purchase Requisitions In Procurement.” <https://www.prokuria.com/post/importance-purchase-requisitions-procurement#viewer-1glfe> (accessed Sep. 08, 2023).
- [9] “6 Reasons to use Purchase Requisitions - Tradogram.” <https://www.tradogram.com/blog/6-reasons-to-use-purchase-requisitions> (accessed Sep. 08, 2023).

- [10] “What is a Purchase Requisition and Why It Is Important for Your Business? - Procurify.” <https://www.procurify.com/blog/purchase-requisition-important-business/> (accessed Sep. 08, 2023).
- [11] “Purchase Requisitions: Best Practices and Challenges.” <https://cashflowinventory.com/blog/purchase-requisition/#section-6> (accessed Sep. 08, 2023).
- [12] “What Is Purchase Requisition and How It Impacts Procurement.” <https://oboloo.com/blog/what-is-purchase-requisition-and-how-it-impacts-procurement/> (accessed Sep. 08, 2023).
- [13] “Procure-to-Pay Software. Everything in One Procure to Pay Platform | Procurify.” <https://www.procurify.com/product/#request> (accessed Sep. 08, 2023).
- [14] “Procure to Pay Software Features | Procurify Platform.” <https://www.procurify.com/product/features/> (accessed Sep. 08, 2023).
- [15] “Purchase Requisition Software System | Tradogram.” <https://www.tradogram.com/software/tools-and-feature/operations-features/purchase-requisition-system> (accessed Sep. 08, 2023).
- [16] “Tradogram Procurement Software Features.” <https://www.tradogram.com/software/tools-and-features> (accessed Sep. 08, 2023).
- [17] “Purchase Requisition Software | Asset Infinity.” <https://www.assetinfinity.com/features/purchase-requisition-software> (accessed Sep. 08, 2023).
- [18] “Node.js vs .NET Core: What to Choose in 2022 | Intelvision.” <https://intelvision.pro/blog/node-js-vs-net-core-what-to-choose-in-2022/> (accessed Jun. 21, 2023).
- [19] “What is ASP.NET Core? | .NET.” <https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core> (accessed Jun. 21, 2023).
- [20] “Overview of ASP.NET Core | Microsoft Learn.” <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0> (accessed Jun. 21, 2023).

- [21] “What is Angular?: Architecture, Features, and Advantages [2022 Edition].” <https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular> (accessed Jun. 21, 2023).
- [22] “Angular - What is Angular?” <https://angular.io/guide/what-is-angular> (accessed Jun. 21, 2023).
- [23] “React vs Angular - Which is the best framework for app development.” <https://appinventiv.com/blog/react-vs-angular/> (accessed Jun. 21, 2023).
- [24] “8 Best Front End Frameworks For Web Development (2023) - InterviewBit.” <https://www.interviewbit.com/blog/best-front-end-frameworks/> (accessed Jun. 21, 2023).
- [25] “What Is a Database | Oracle.” <https://www.oracle.com/database/what-is-database/> (accessed Jun. 20, 2023).
- [26] “Relational vs Non-Relational Databases | insightsoftware.” <https://insightsoftware.com/blog/whats-the-difference-relational-vs-non-relational-databases/> (accessed Jun. 20, 2023).
- [27] “Relational vs. Non-Relational Databases: Features and Benefits.” <https://www.couchbase.com/blog/relational-vs-non-relational-database/> (accessed Jun. 20, 2023).
- [28] “SQL Server: Advantages, Best Practices, and Top Monitoring Tools.” <https://www.tek-tools.com/database/sql-server-best-practices-and-tools#content> (accessed Jun. 20, 2023).
- [29] E. Shkodra, E. Jajaga, and M. Shala, “Development and Performance Analysis of RESTful APIs in Core and Node.js using MongoDB Database”, doi: 10.5220/0010621200003058.
- [30] “Node.js - O que é, como funciona e quais as vantagens | OPUS.” <https://www.opus-software.com.br/index.html%3Fp=7232.html> (accessed Jun. 21, 2023).
- [31] “Angular vs React: which framework to pick?” <https://www.imaginarycloud.com/blog/angular-vs-react/#React> (accessed Jun. 21, 2023).
- [32] “Stack Overflow Developer Survey 2022.” <https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe> (accessed Jun. 21, 2023).

- [33] “Companies That Use React.” <https://career karma.com/blog/companies-that-use-react/> (accessed Jun. 21, 2023).
- [34] “What is FURPS+? – Business Analyst Training in Hyderabad – COEPD.” <https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/> (accessed Jul. 04, 2023).
- [35] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)*. 2001. Accessed: Jul. 05, 2023. [Online]. Available: <http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0130925691>
- [36] “What do Requirements Stand for? – School of Information Systems.” <https://sis.binus.ac.id/2018/02/12/what-do-requirements-stand-for/> (accessed Jul. 04, 2023).
- [37] “The C4 model for visualising software architecture.” <https://c4model.com/> (accessed Jul. 07, 2023).
- [38] “Diagrams with C4 Model – Bits ‘n Bytes.” <https://mattjhayes.com/2020/05/10/diagrams-with-c4-model/> (accessed Jul. 07, 2023).
- [39] “The C4 Model for Software Architecture.” <https://www.infoq.com/articles/C4-architecture-model/> (accessed Jul. 07, 2023).
- [40] “4+1 Architectural view model in Software | by Pasan Devin Jayawardene | Javarevisited | Medium.” <https://medium.com/javarevisited/4-1-architectural-view-model-in-software-ec407bf27258> (accessed Jul. 05, 2023).
- [41] “nunopsilva / bulletproof-nodejs+ddd / wiki / Views — Bitbucket.” <https://bitbucket.org/nunopsilva/bulletproof-nodejs-ddd/wiki/Views.md> (accessed Jul. 07, 2023).
- [42] “Kruchten’s 4 + 1 views of Software Design | by Puneet Sapra | The Mighty Programmer | Medium.” <https://medium.com/the-mighty-programmer/kruchtens-views-of-software-design-e9088398c592> (accessed Jul. 07, 2023).
- [43] “Onion Architecture: Definition, Principles & Benefits | CodeGuru.” <https://www.codeguru.com/csharp/understanding-onion-architecture/> (accessed Jul. 06, 2023).

- [44] “Integration Testing Data Access in ASP.NET Core - DZone.”
<https://dzone.com/articles/integration-testing-data-access-in-aspnet-core> (accessed
Sep. 09, 2023).