

# IPBeja

INSTITUTO POLITÉCNICO  
DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

## Relatório de Projeto Integrado

Plataforma de Gestão de Laboratório de Investigação

*Vasco Gomes N<sup>º</sup> 19921  
Fernando Simões N<sup>º</sup> 19922*

Beja, 3 de Fevereiro de 2023



**INSTITUTO POLITÉCNICO DE BEJA**

**Escola Superior de Tecnologia e Gestão**

**Licenciatura em Engenharia Informática**

## **Relatório de Projeto Integrado**

**Plataforma de Gestão de Laboratório de Investigação**

Vasco Gomes Nº 19921

Fernando Simões Nº 19922

Orientado por :

Professor João Carlos Martins, IPBeja

Relatório de Projeto Integrado

2023



# Resumo

## *Relatório de Projeto Integrado*

### *Plataforma de Gestão de Laboratório de Investigação*

Este projeto Web full-stack vai utilizar a tecnologia Flask para back-end, React para front-end e SQLite para gestor de base de dados. A aplicação é construída usando Python para a lógica do lado do servidor e JavaScript (React) para as interfaces de utilizador do lado do cliente. O SQLite é usado como mecanismo de base de dados para armazenar e recuperar dados com eficiência. O aplicativo visa fornecer uma interface amigável e responsiva, ao mesmo tempo em que lida com grandes quantidades de dados. O uso do Flask e do React permite uma integração perfeita do back-end e do front-end, enquanto o SQLite fornece uma solução de base de dados leve e fácil de usar. O objetivo do projeto é criar um sistema robusto e escalável que possa atender às necessidades dos seus utilizadores.

**Palavras-chave:** *Python, Flask, React, Bootstrap, API, SQLite.*



# Abstract

## *Relatório de Projeto Integrado*

### *Plataforma de Gestão de Laboratório de Investigação*

This project is a full-stack web application that utilizes the Flask framework for the backend, React for the frontend, and SQLite for data management. The application is built using Python for the server-side logic and JavaScript for the client-side user interface. SQLite is used as the database engine to store and retrieve data efficiently. The application aims to provide a user-friendly and responsive interface while also handling large amounts of data. The use of Flask and React allows for a seamless integration of the backend and frontend while SQLite provides a lightweight and easy-to-use database solution. The goal of the project is to create a robust and scalable application that can meet the needs of its users.

**Keywords:** *Python, Flask, React, Bootstrap, API, SQLite.*



## *Agradecimentos*

Queríamos deixar os nossos agredecimentos aos professores que nos guiaram no projetos desta unidade curricular, em especial ao professor João Martins o orientador deste projeto, que nos colocou novos desafios a cada semana que passava de modo a conseguir tornar esta plataforma o modular possível. Um agradecimento ao professor Luís Bruno pelos conhecimentos transmitidos durante as unidades curriculares de Tecnologias Web e Ambientes Móveis, e Desenvolvimento de Aplicações Web, tanto de front-end(React), como de back-end. Gostaríamos de expressar ainda a nossa gratidão á professora Isabel Brito pelos conhecimentos transmitidos durante a unidades curriculares de Engenharia de Software e Bases de Dados 2, assim como ao professor Luís Garcia pelos conhecimentos transmitidos na unidade curricular de Interação Pessoa Computador e por fim ao professor Nuno Sidónio pelos exemplos dados de sistemas semelhantes ao nosso para pudermos realizar uma comparação.



# Índice

<b>Resumo</b>	i
<b>Abstract</b>	iii
<b>Agradecimentos</b>	v
<b>Índice</b>	vii
<b>Índice de Figuras</b>	xii
<b>Índice de Tabelas</b>	xiii
<b>1 Introdução</b>	1
<b>2 Análise do Projeto</b>	3
2.1 Quem vai utilizar o sistema? . . . . .	3
2.2 Que tarefas executam atualmente? . . . . .	3
2.3 Que tarefas são desejáveis? . . . . .	3
2.3.1 Gerir listagem de equipamento de laboratório . . . . .	3
2.3.2 Fazer requisição de equipamentos . . . . .	3
2.3.3 Fazer devolução de equipamentos . . . . .	3
2.4 Como se aprendem as tarefas? . . . . .	3
2.5 Onde são desempenhadas as tarefas? . . . . .	3
2.6 Quais as relações entre utilizadores e informação? . . . . .	4
2.7 Que outros instrumentos tem o utilizador? . . . . .	4
2.8 Como comunicam os utilizadores entre si? . . . . .	4
2.9 Qual é a frequência do desempenho das tarefas? . . . . .	4
2.10 Quais são as restrições de tempo? . . . . .	4
2.11 Que acontece se algo corre mal? . . . . .	4
2.12 Sistemas Semelhantes . . . . .	4
<b>3 Caracterização dos Atores do Sistema</b>	5
3.1 Objetivo . . . . .	5

## ÍNDICE

---

3.2 Persona . . . . .	5
3.2.1 Docente Responsável . . . . .	5
3.3 Diagrama de Casos de Uso . . . . .	6
3.4 Tabelas de Especificação . . . . .	6
3.4.1 Caso de uso - Gerir Materiais . . . . .	7
3.4.2 Caso de uso - Fazer requisição de equipamentos . . . . .	8
3.4.3 Caso de uso - Fazer devolução de equipamentos . . . . .	9
<b>4 Requisitos funcionais e não funcionais</b>	<b>11</b>
4.1 Requisitos Funcionais . . . . .	11
4.1.1 . . . . .	11
4.2 Requisitos não funcionais . . . . .	12
4.2.1 Segurança . . . . .	12
4.2.2 Desempenho . . . . .	12
4.2.3 Acessibilidade / Facilidade de Uso . . . . .	12
<b>5 Desenho de Interfaces Gráficas</b>	<b>13</b>
5.1 Wireframes . . . . .	13
5.1.1 Interface de Login . . . . .	13
5.1.2 Interface de Registar . . . . .	14
5.1.3 Interface Homepage . . . . .	15
5.1.4 Interface Dashboard . . . . .	16
5.1.5 Interface de Requisição . . . . .	17
5.1.6 Interface de Devolução . . . . .	18
5.1.7 Interface de Listagem de Materiais . . . . .	19
5.1.8 Interface de Atualização de Stocks . . . . .	20
5.1.9 Interface de Adicionar um Novo Material . . . . .	21
5.1.10 Interface para Criar Kit de Material . . . . .	22
5.1.11 Interface para Listar os Kits Materiais . . . . .	23
5.1.12 Interface para Adicionar Tipo de Material . . . . .	24
5.1.13 Interface para Criar Projeto . . . . .	25
5.1.14 Interface para Listar Projetos . . . . .	26
5.2 Storyboards . . . . .	27
5.2.1 Gerir listagem de material de laboratório . . . . .	27
5.2.2 Fazer requisição de materiais . . . . .	29
5.2.3 Fazer devolução de materiais . . . . .	31
<b>6 Desenho da Base de Dados</b>	<b>33</b>
6.1 Modelo Conceptual . . . . .	33
6.2 Diagrama Entidade-Relação . . . . .	33

6.2.1	.....	33
<b>7 Decisões de Implementação</b>		<b>35</b>
7.1 Back-end	.....	35
7.1.1 Flask	.....	35
7.2 Sistema de Gestão de Base de Dados	.....	36
7.2.1 SQLite	.....	36
7.2.2 SQLAlchemy	.....	36
7.3 Front-end	.....	37
7.3.1 React	.....	37
7.3.2 React Router	.....	38
7.3.3 Bootstrap	.....	38
<b>8 Dificuldades de Implementação</b>		<b>39</b>
<b>9 Testes com Utilizadores</b>		<b>45</b>
9.1 Participantes	.....	45
9.2 Moderador de testes	.....	45
9.3 Local dos testes	.....	45
9.4 Recolha de dados (Questionário)	.....	45
9.5 Questionários	.....	45
9.6 Resultados	.....	46
9.7 Análise dos dados	.....	47
<b>10 Melhorias Futuras</b>		<b>49</b>
<b>11 Conclusão</b>		<b>51</b>
<b>Bibliografia</b>		<b>53</b>



# Índice de Figuras

3.1	Persona - Diogo André( <i>in https://thispersondoesnotexist.com/</i> ) . . . . .	5
3.2	Diagrama de Casos de Uso . . . . .	6
5.1	Wireframe - Login . . . . .	14
5.2	Wireframe - Registrar . . . . .	15
5.3	Wireframe - HomePage . . . . .	16
5.4	Wireframe - Dashboard . . . . .	17
5.5	Wireframe - Requisição . . . . .	18
5.6	Wireframe - Devolução . . . . .	19
5.7	Wireframe - Lista de Materiais . . . . .	20
5.8	Wireframe - Atualizar Stocks . . . . .	21
5.9	Wireframe - Adicionar Material . . . . .	22
5.10	Wireframe - Adicionar Kit de Material . . . . .	23
5.11	Wireframe - Lista de Kits . . . . .	24
5.12	Wireframe - Adicionar Tipo de Material . . . . .	25
5.13	Wireframe - Adicionar Projeto . . . . .	26
5.14	Wireframe - Listar Projetos . . . . .	27
5.15	Caso de Uso - Gerir Material . . . . .	28
5.16	Caso de Uso - Requisitar Material . . . . .	30
5.17	Caso de Uso - Devolver Material . . . . .	31
6.1	Modelo Conceptual . . . . .	33
6.2	Diagrama Entidade-Relação . . . . .	34
8.1	Interface Requisição . . . . .	39
8.2	Código Requisição React . . . . .	40
8.3	Código Requisição React . . . . .	41
8.4	Código Requisição React . . . . .	42
8.5	Código Requisição Flask . . . . .	43
9.1	Tabela - Testes com Utilizadores . . . . .	47



# Índice de Tabelas

3.1	Caso de uso - Gerir Materiais . . . . .	7
3.2	Caso de uso - Fazer requisição de equipamentos . . . . .	8
3.3	Caso de uso - Fazer devolução de equipamentos . . . . .	9



# Capítulo 1

## Introdução

Este trabalho tem como objetivo realizar um sistema de gestão para o laboratório *SEPSI*, assim como cimentar-nos as lições necessárias já dadas noutras unidades curriculares de como se faz um projeto do início ao fim, passando por todas as fases do desenvolvimento do mesmo.

A motivação necessária para realizar este trabalho foi a necessidade de um sistema de gestão para o laboratório *SEPSI* para organização dos materiais existentes, assim como de colocar os nossos conhecimentos à prova, e como de servir de preparação para estágio.

Neste projeto vão ser usadas tecnologias como *React* para *front-end*, *Flask* e *SQLAlchemy* para *back-end*.



## Capítulo 2

# Análise do Projeto

Este capítulo responde ás 11 Quentões da analise de Tarefas.

### 2.1 Quem vai utilizar o sistema?

O sistema irá ser utilizado pelos professores e estudantes do Instituto Politécnico de Beja que estejam de alguma forma relacionados a projetos do laboratório *SEPSI*.

### 2.2 Que tarefas executam atualmente?

Atualmente não executa nenhuma tarefa, porque o sistema ainda não entrou na fase de implementação.

### 2.3 Que tarefas são desejáveis?

2.3.1 Gerir listagem de equipamento de laboratório

2.3.2 Fazer requisição de equipamentos

2.3.3 Fazer devolução de equipamentos

### 2.4 Como se aprendem as tarefas?

Este sistema não requer aprendizagens especiais para quem utiliza regularmente *websites*

### 2.5 Onde são desempenhadas as tarefas?

As tarefas deverão ser desempenhadas numa máquina presente no laboratório *SEPSI*, que possua um browser, assim como uma ligação á internet.

## 2.6 Quais as relações entre utilizadores e informação?

Para que o utilizador possa requisitar, e devolver os equipamentos e ferramentas do laboratório, este necessita de possuir uma conta de utilizador, e realizando a autenticação em qualquer tipo de dispositivo, este pode realizar as tarefas que deseja.

Para que seja possível ver a listagem de equipamentos e ferramentas do laboratório não é necessário qualquer tipo de *login*.

## 2.7 Que outros instrumentos tem o utilizador?

O utilizador apenas precisa de uma máquina com acesso á internet e com rato e teclado.

## 2.8 Como comunicam os utilizadores entre si?

Os utilizadores não comunicam entre si.

## 2.9 Qual é a frequência do desempenho das tarefas?

A frequência de desempenho das tarefas é definida de acordo com as necessidades dos utilizadores deste sistema uma vez que estes podem requisitar vários equipamentos e ferramentas durante um dia, assim como devolve-las ou não requisitar caso não necessitem das mesmas.

## 2.10 Quais são as restrições de tempo?

A aplicação não apresenta restrições de tempo impostas, os utilizadores para que estes realizem as suas tarefas.

## 2.11 Que acontece se algo corre mal?

Quando algo corre mal, é mostrada uma mensagem de erro para o erro específico que aconteceu, utilizando uma técnica de programação para tratamento de exceções, mais conhecida como *try catch*.

## 2.12 Sistemas Semelhantes

Os sistemas semelhantes apresentados no decorrer deste projeto foi o [myL23], uma vez que os objetivos eram bastante similares aos nossos, que são realizar a gestão de laboratórios. Vale a pena ainda referir que este sistema semelhante foi-nos indicado pelo professor de física Nuno Sidónio.

## Capítulo 3

# Caracterização dos Atores do Sistema

Neste capítulo iremos começar por criar personas, neste sistema existe apenas uma persona, o docente responsável.

### 3.1 Objetivo

O objetivo do Docente é conseguir gerir todo o material do laboratório *SEPSI*. O docente deve conseguir ver os stocks atuais, fazer requisições de materiais e também devoluções, por fim também deve conseguir atualizar as quantidades dos diversos materiais.

### 3.2 Persona

#### 3.2.1 Docente Responsável



**Figura 3.1:** Persona - Diogo André(*in* <https://thispersondoesnotexist.com/>)

Diogo André, anos, licenciados em Engenharia Informática e Engenharia eletrónica e com Mestrado em Engenharia Eletrónica e Telecomunicações na Universidade Nova, é actualmente docente no Instituto Politécnico de Beja, onde leciona as unidades curriculares

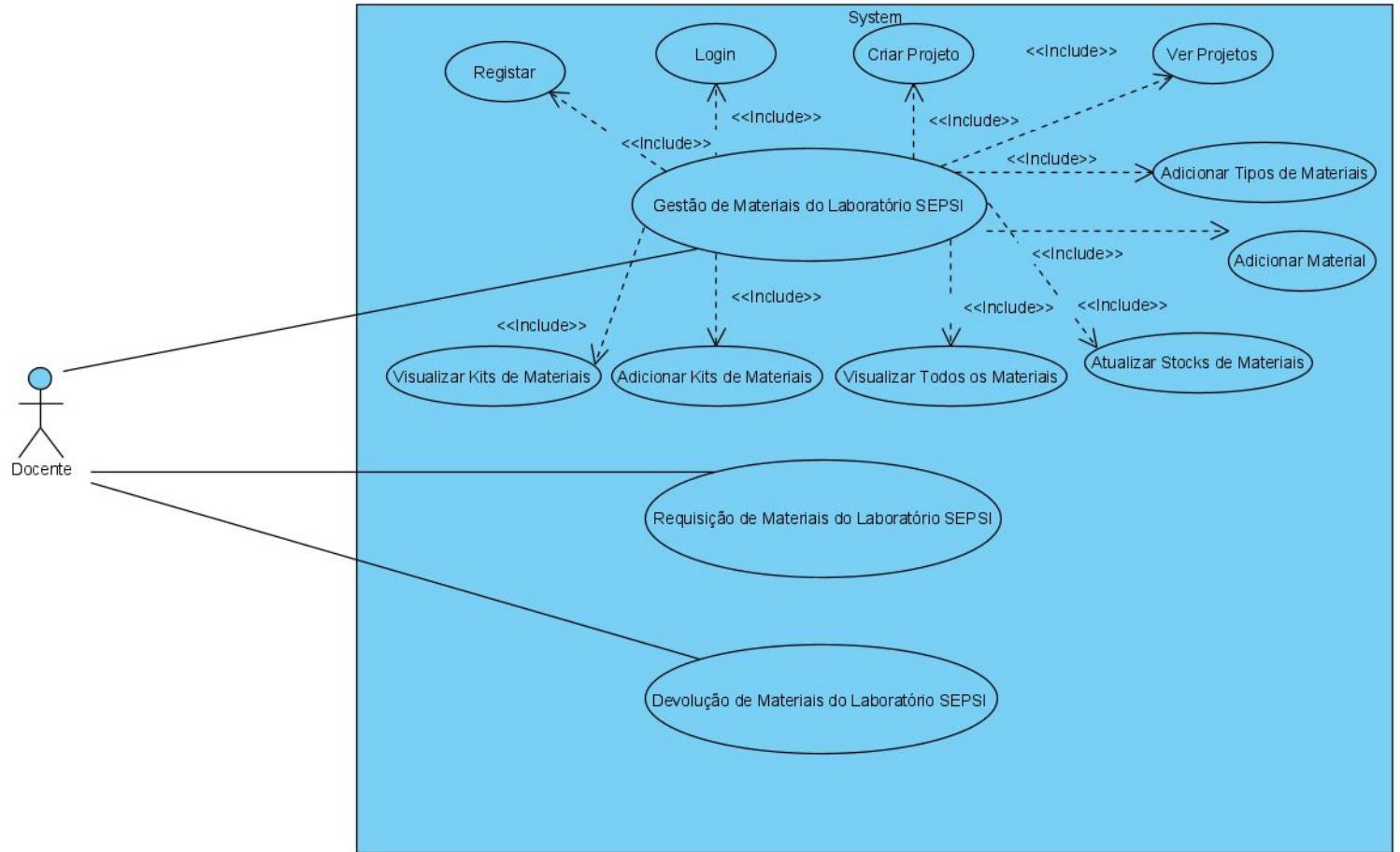
### 3. CARACTERIZAÇÃO DOS ATORES DO SISTEMA

---

de Linguagens de Programação e Estruturas de Dados e Algoritmos. É o principal responsável do laboratório *SEPSI* que se encontra também no mesmo local.

### 3.3 Diagrama de Casos de Uso

Nesta secção iremos apresentar o diagrama de casos de uso do sistema a ser implementado neste projeto, assim como as tabelas de especificação referentes ao mesmo, como podem ser vistos nas secções seguintes.



**Figura 3.2:** Diagrama de Casos de Uso

### 3.4 Tabelas de Especificação

### 3.4.1 Caso de uso - Gerir Materiais

Caso de Uso	Gerir Materiais		
Atores	Docente		
Data	06/11/2022		
Descrição	O docente pretende ver o material do laboratório, atualizar as suas quantidades, e adiconar um novo material no sistema.		
Pré-condições	O Docente tem de estar registado no sistema.		
Pós-condições	O Docente fica a saber o stock existente e adiciona um material novo no sistema.		
Fluxo de eventos	Nº	Entrada do Ator	Resposta do Sistema
	1	O Docente realiza o login no Sistema	
	2		O Sistema verifica os dados de Login e redireciona o docente para o dashboard
	3	O Docente vai a secção de atualizar Stocks e altera a quantidade do material desejado.	
	4		O Sistema confere os dados inseridos e altera a base de dados. O sistema mostra um alerta de que os dados foram alterados com sucesso.
	5	O Docente vai a secção de adicionar um novo material e insere os dados relativos ao mesmo.	
	6		O sistema insere o material novo na base de dados. O sistema mostra um alerta de que o novo material foi inserido com sucesso.

**Tabela 3.1:** Caso de uso - Gerir Materiais

### 3. CARACTERIZAÇÃO DOS ATORES DO SISTEMA

---

#### 3.4.2 Caso de uso - Fazer requisição de equipamentos

Caso de Uso	Fazer uma requisição de um material		
Atores	Docente		
Data	06/11/2022		
Descrição	O docente pretende fazer uma requisição de um determinado material do laboratório SEPSI.		
Pré-condições	O Docente tem de estar registado no sistema.		
Pós-condições	O Docente pode levar o material requisitado do laboratório SEPSI.		
Fluxo de eventos	Nº	Entrada do Ator	Resposta do Sistema
	1	O Docente realiza o registo no sistema.	
			O sistema cria na base de dados um novo utilizador e alerta o docente que a conta foi criada com sucesso. O sistema reencaminha o docente para a página de login.
	2	O Docente realiza o login no Sistema	
	3		O Sistema verifica os dados de Login e redireciona o docente para o dashboard
	4	O Docente vai à secção de requisição e preenche o formulário de acordo com os materiais que pretende requisitar, e carrega no botão requisitar.	
	5		O Sistema confere os dados inseridos e realiza uma nova requisição do docente, e de seguida mostra um alerta a indicar que a requisição foi feita com sucesso.

**Tabela 3.2:** Caso de uso - Fazer requisição de equipamentos

### 3.4.3 Caso de uso - Fazer devolução de equipamentos

Caso de Uso	Fazer uma devolução de um material		
Atores	Docente		
Data	06/11/2022		
Descrição	O docente pretende fazer uma devolução de um determinado material do laboratório SEPSI.		
Pré-condições	O Docente tem de estar registado no sistema.		
Pós-condições	O Docente repõe o material foi requisitado do laboratório SEPSI.		
Fluxo de eventos	Nº	Entrada do Ator	Resposta do Sistema
	1	O Docente realiza o login no Sistema	
	2		O Sistema verifica os dados de Login e redireciona o docente para o dashboard
	3	O Docente vai a secção de devolução de materiais e seleciona da tabela o material e caso não encontre o material pretendido pode procurar o mesmo através da barra de pesquisa. E quando o material for encontrado o docente carrega no botão devolver.	
	4		O Sistema confere os dados inseridos e realiza uma devolução dos materiais selecionados e de seguida mostra um alerta a confirmar a devolução.

**Tabela 3.3:** Caso de uso - Fazer devolução de equipamentos



## Capítulo 4

# Requisitos funcionais e não funcionais

Este capítulo exemplifica os requisitos funcionais e não funcionais da nossa aplicação.

### 4.1 Requisitos Funcionais

Os requisitos funcionais são representativos das funcionalidades do *software* requeridas pelo cliente.

#### 4.1.1

##### **Gerir listagem de equipamento de laboratório**

Este software permite que seja visualizada uma lista com todos os equipamentos e ferramentas constituintes do laboratório *SEPSI*, assim como detalhes como por exemplo se estes se encontram requisitados ou não, assim como as quantidades dos items acima referidos, para que seja possível visualizar alguma falta de *stock* sem ser necessário proceder a um login por parte dos utilizadores.

##### **Fazer requisição de equipamentos**

O *software* deverá permitir os utilizadores realizarem requisições dos diversos equipamentos disponíveis no laboratório *SEPSI*, ficando o nome da pessoa que fez o requisito associado ao levantamento, para saber quem e quando fez o levantamento de um determinado equipamento.

##### **Fazer devolução de equipamentos**

O requisito funcional de devolução de equipamentos e ferramentas ao Laboratório *SEPSI*, tem como objetivo repor as ferramentas e equipamentos no stock do laboratório, para

## **4. REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS**

---

que estes possam ser utilizados por outros utilizadores do mesmo laboratório desde que se encontrem devidamente registados na plataforma.

### **4.2 Requisitos não funcionais**

#### **4.2.1 Segurança**

Foi utilizada a biblioteca *bcrypt* para realizar a encriptação da password, e o que fica armazenado na base de dados é uma *hash* que corresponde à password, matéria essa que é lecionada nas aulas da unidade curricular de Segurança de Redes Informática.

#### **4.2.2 Desempenho**

Foi utilizada a plataforma *flask* para o desenvolvimento do *back-end* devido a ser um ferramenta leve que permite aos desenvolvedores utilizar apenas as bibliotecas necessárias ao desenvolvimento do projeto. Já para o *front-end* foi utilizado *React* uma vez que este utiliza várias técnicas para minimizar o custo das operações necessárias para realizar atualizações da UI, isto torna as aplicações *React* muito eficientes no que toca a interfaces gráficas porque está dividido em componentes, e estes são mais fáceis de realizar a manutenção, à semelhança daquilo que se passa na programação orientada por objetos em *MVC* e que torna os programas mais amigáveis a alterações por parte do programador.

#### **4.2.3 Acessibilidade / Facilidade de Uso**

As interfaces vão estar a seguir as recomendações da acessibilidade da agência modernização administrativa [acessibilidade.gov.pt](http://acessibilidade.gov.pt), informação esta que nos foi transmitida pelo professor das unidades curriculares de Tecnologias Web, e Desenvolvimento de Aplicações Web, assim como coordenador do curso de engenharia informática Luís Carlos Bruno.

## Capítulo 5

# Desenho de Interfaces Gráficas

Este capítulo mostra o desenvolvimento das interfaces gráficas do *software* em desenvolvimento.

### 5.1 Wireframes

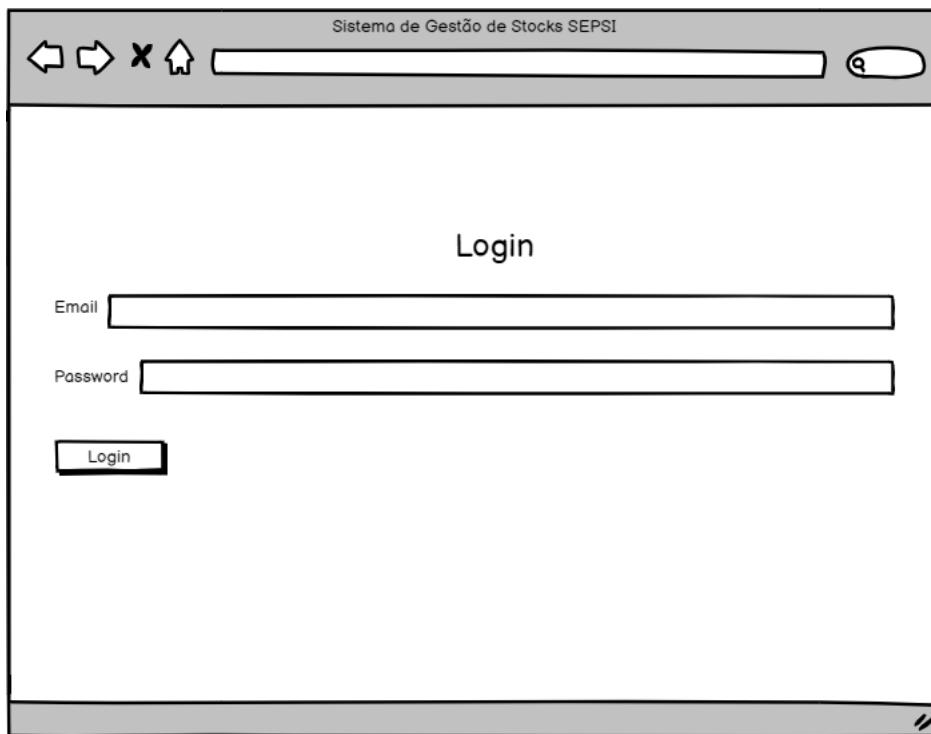
Primeiramente foi feito o desenho inicial das interfaces que o utilizador irá utilizar para interagir com a aplicação, utilizamos os princípios de usabilidade lecionados na unidade curricular de Tecnologias Web e Ambiente Móvel.

#### 5.1.1 Interface de Login

Como podemos verificar na figura abaixo, temos a interface que permite ao utilizador do sistema realizar o Login, assim como de registar-se no sistema.

## 5. DESENHO DE INTERFACES GRÁFICAS

---



**Figura 5.1:** Wireframe - Login

### 5.1.2 Interface de Registrar

Esta interface permite os utilizadores criar uma conta para ter acesso ás funcionalidades da aplicação.

The wireframe shows a registration form titled "Registrar Novo Utilizador". It includes fields for Name, Email, Telephone, Password, and Repeat Password, each with a corresponding input box. A "Registrar Utilizador" button is at the bottom.

Sistema de Gestão de Stocks SEPSI

Registrar Novo Utilizador

Nome

Email

Telefone

Password

Repetir Password

Registrar Utilizador

**Figura 5.2:** Wireframe - Registrar

### 5.1.3 Interface Homepage

Esta é a página inicial do sistema de gestão de stocks, como podemos verificar na figura seguinte. Esta permite visualizar os equipamentos e ferramentas que existem em stock, assim como a quantidade que foi requisitada e por quem. Por último esta página pode ainda redirecionar para a interface de registar e a interface de login através dos botões.

## 5. DESENHO DE INTERFACES GRÁFICAS

---

The wireframe shows a web browser window titled "Sistema de Gestão de Stocks SEPSI". The interface includes a top navigation bar with icons for back, forward, stop, and search. Below the title, there is a main heading "Sistema de Gestão de Stocks SEPSI" and a descriptive text: "Aqui podem ser visualizadas as Ferramentas e os Materiais que existem no laboratório, pode pesquisar os materiais através da barra de pesquisa abaixo por qualquer um dos campos da tabela". There are two input fields: "Pesquisa" and "Nome Material". A table displays stock information with columns: Nome, Quantidade, Observações, and Data de Aquisição. The table rows show the following data:

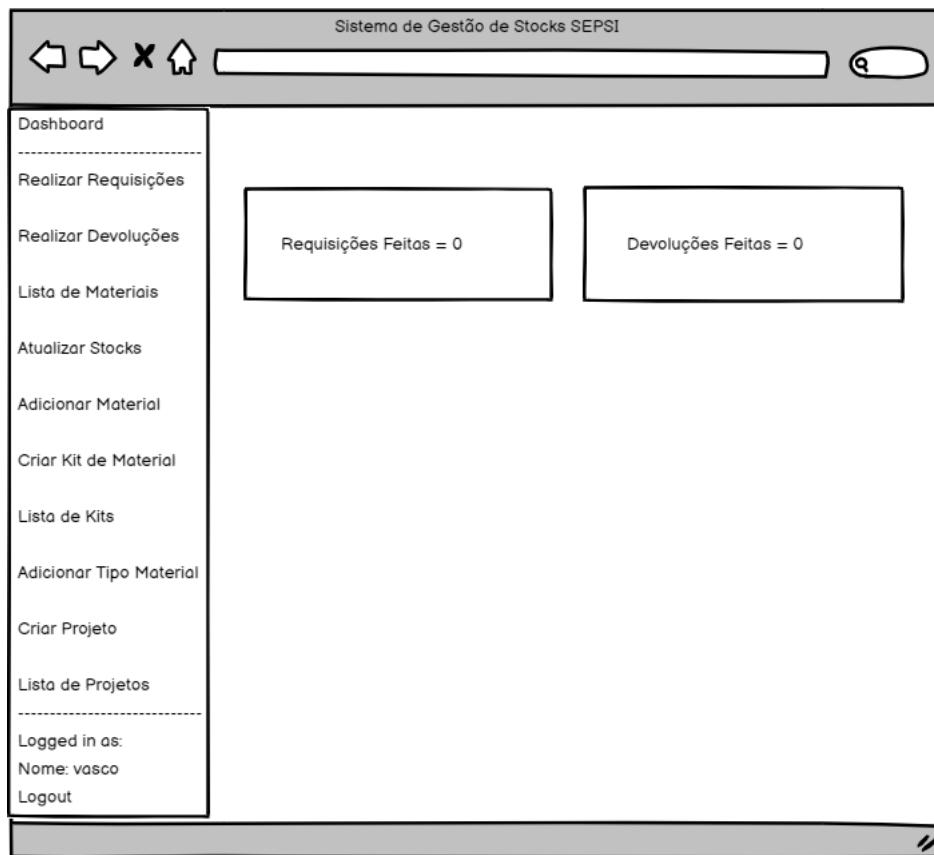
Nome	Quantidade	Observações	Data de Aquisição
Giacomo Guilizzoni Founder & CEO	40	Peldi	<input type="radio"/>
Marco Botton Tuttofare	38		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	41	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef	:)	Val	<input checked="" type="checkbox"/>

At the bottom left is a "Login" button, and at the bottom right is a "Registrar Utilizador" button.

**Figura 5.3:** Wireframe - HomePage

### 5.1.4 Interface Dashboard

É nesta página que os utilizadores podem selecionar a tarefa que pretendem realizar, através do menu presente no lado esquerdo da página. Apresenta também um breve estatística dos equipamentos que se encontram requisitados.



**Figura 5.4:** Wireframe - Dashboard

### 5.1.5 Interface de Requisição

Nesta Interface o utilizador consegue fazer o registo de uma requisição, bastando apenas que este preencha o formulário com os dados do equipamento e da pessoa que vai fazer a requisição.

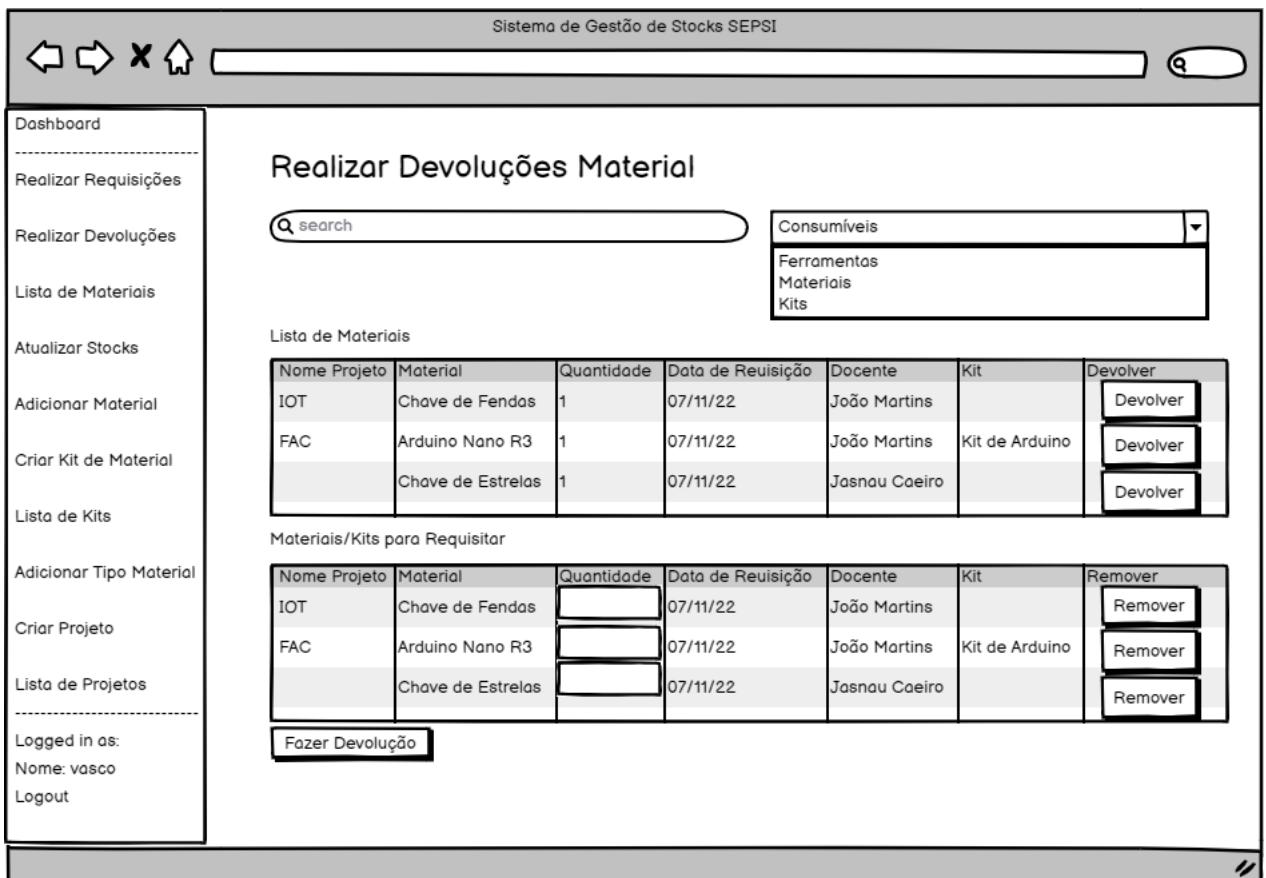
## 5. DESENHO DE INTERFACES GRÁFICAS

The wireframe shows a user interface for material requests. At the top, there's a header bar with icons for back, forward, close, and home, followed by a search bar and a magnifying glass icon. On the left, a sidebar menu lists various administrative functions: Dashboard, Realizar Requisições, Realizar Devoluções, Lista de Materiais, Atualizar Stocks, Adicionar Material, Criar Kit de Material, Lista de Kits, Adicionar Tipo Material, Criar Projeto, and Lista de Projetos. Below this, it shows 'Logged in as: Nome: vasco' and 'Logout'. The main content area has a title 'Requisições de Materiais/Kits' and a subtitle instructing users to fill out a form and click 'Fazer Requisição'. It includes fields for 'Nome Docente ou Aluno' (José Silva), 'Projeto' (radio buttons for 'Usar em Projeto' or 'Não Usar em Projeto'), 'Nome Projeto' (text input), and a search bar with dropdown filters for 'Consumíveis', 'Ferramentas', 'Materiais', and 'Kits'. There are two tables: 'Lista de Materiais' (Materials List) showing 'Cabos de Rede' (45) and 'Fichas RJ-45' (200), and 'Materiais/Kits para Requisitar' (Materials/Kits to Request) showing 'Fichas RJ-45' (4). A date input field shows '13/11/22' with a calendar icon. A large 'Fazer Requisição' button is at the bottom.

Figura 5.5: Wireframe - Requisição

### 5.1.6 Interface de Devolução

A interface de devolução de ferramentas e materiais, como o próprio nome indica tem como objetivo o utilizador realizar uma pesquisa pelo seu nome e carregar no botão de devolver para que o *stock* do material devolvido seja acrescentado no sistema.

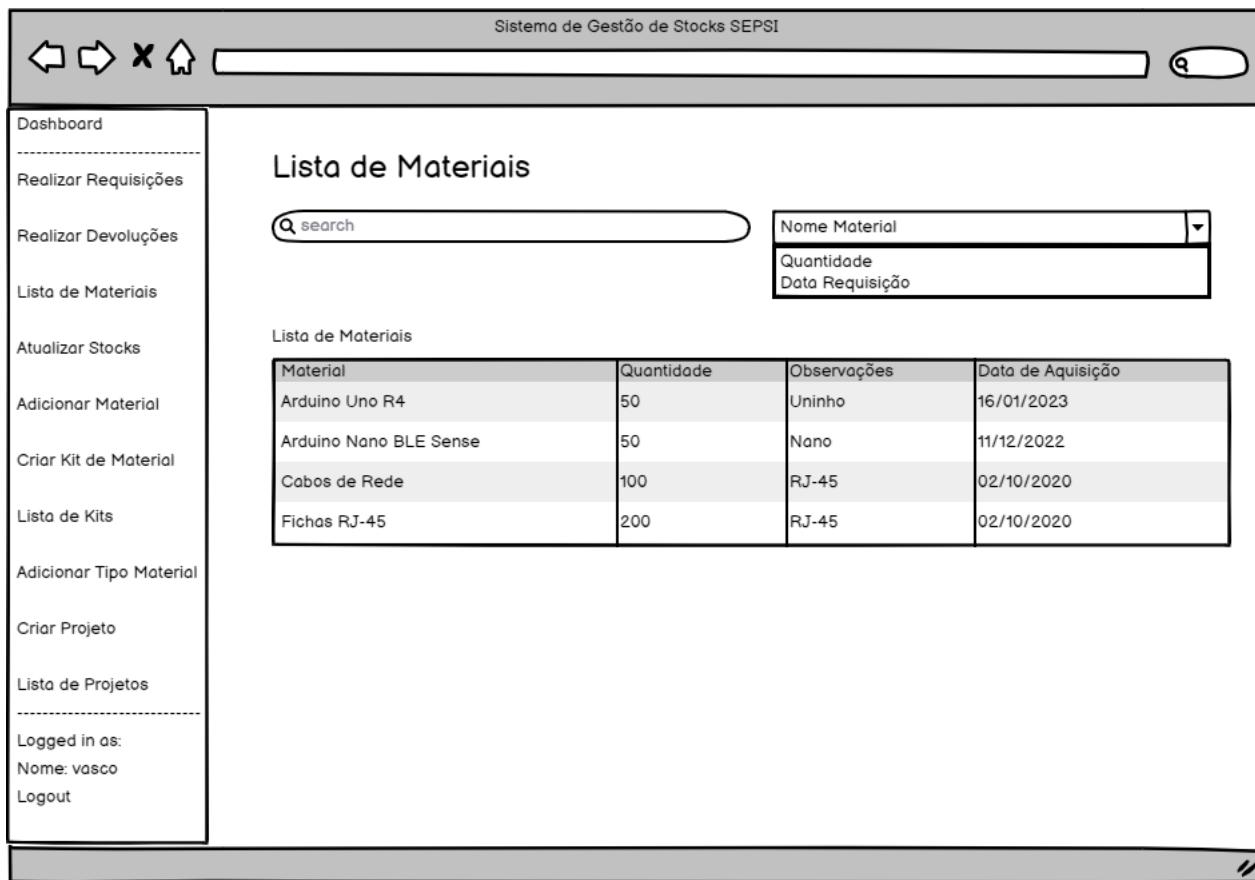


**Figura 5.6:** Wireframe - Devolução

### 5.1.7 Interface de Listagem de Materiais

Por fim é nesta interface que os utilizadores conseguem ter uma visão geral de todos os materiais, e as respetivas quantidades, que o laboratório tem, incluindo se estes se encontram requisitados ou não.

## 5. DESENHO DE INTERFACES GRÁFICAS



**Figura 5.7:** Wireframe - Lista de Materiais

### 5.1.8 Interface de Atualização de Stocks

A interface de atualização de *stocks* permite que o utilizador atualize a quantidade de materiais presentes no laboratório.

Sistema de Gestão de Stocks SEPSI

Dashboard

Realizar Requisições

Realizar Devoluções

Lista de Materiais

**Atualizar Stocks**

search

Material	Observação	Data Aquisição	Quantidade	Nova Quantidade	Atualizar
Arduino Uno R4	Uninho	16/01/2023	50	<input type="text"/>	Atualizar
Arduino Nano BLE Sense	Nano	11/12/2022	50	<input type="text"/>	Atualizar
Cabos de Rede	RJ-45	02/10/2020	100	<input type="text"/>	Atualizar
Fichas RJ-45	RJ-45	02/10/2022	200	<input type="text"/>	Atualizar

Lista de Materiais

Nome: vasco

Logout

**Figura 5.8:** Wireframe - Atualizar Stocks

### 5.1.9 Interface de Adicionar um Novo Material

Para que seja possível adicionar novo material é necessário preencher um breve formulário com alguns campos como o nome do material, a quantidade e o tipo de material, e de seguida carregar em Adicionar Novo Material, como pode ser visualizado na figura seguinte.

## 5. DESENHO DE INTERFACES GRÁFICAS

The wireframe shows a user interface for adding new material. At the top, there's a header bar with icons for back, forward, close, and search. On the left, a sidebar lists various management functions: Dashboard, Realizar Requisições, Realizar Devoluções, Lista de Materiais, Atualizar Stocks, Adicionar Material, Criar Kit de Material, Lista de Kits, Adicionar Tipo Material, Criar Projeto, and Lista de Projetos. Below this, a message area says "Logged in as: Nome: vasco Logout". The main content area is titled "Adicionar Novo Material". It contains several input fields: "Nome do Material" (Name of Material), "Quantidade" (Quantity), "Observações" (Observations), "Tipo de Material" (Type of Material) with options like Consumíveis, Ferramentas, and Equipamentos, "Associar a Projeto" (Associate to Project) with options like Não Associar and Projeto de Teste, and a date field "Data de Aquisição" (Purchase Date) with a calendar icon. At the bottom is a large "Adicionar Material" button.

**Figura 5.9:** Wireframe - Adicionar Material

### 5.1.10 Interface para Criar Kit de Material

Também deve ser possível criar kits de materiais. Para que seja possível adicionar novo material é necessário preencher um breve formulário com alguns campos como o nome do material, a quantidade e o tipo de material, e de seguida carregar em Adicionar Novo Material, como pode ser visualizado na figura seguinte.

Sistema de Gestão de Stocks SEPSI

Dashboard

Realizar Requisições

Realizar Devoluções

Lista de Materiais

Atualizar Stocks

Adicionar Material

Criar Kit de Material

Lista de Kits

Adicionar Tipo Material

Criar Projeto

Lista de Projetos

Logged in as:  
Nome: vasco  
Logout

### Adicionar Novo Kit de Material

Nome Material Kit

search

**Lista de Materiais**

Material	Quantidade Total	Adicionar
Arduino Uno R4	70	<input type="button" value="Adicionar"/>
Cabos de Rede	100	<input type="button" value="Adicionar"/>
Fichas RJ-45	200	<input type="button" value="Adicionar"/>

**Materiais no Kit**

Material	Quantidade no Kit	Remover
Arduino Uno R4	<input type="text"/>	<input type="button" value="Remover"/>
Cabos de Rede	<input type="text"/>	<input type="button" value="Remover"/>
Fichas RJ-45	<input type="text"/>	<input type="button" value="Remover"/>

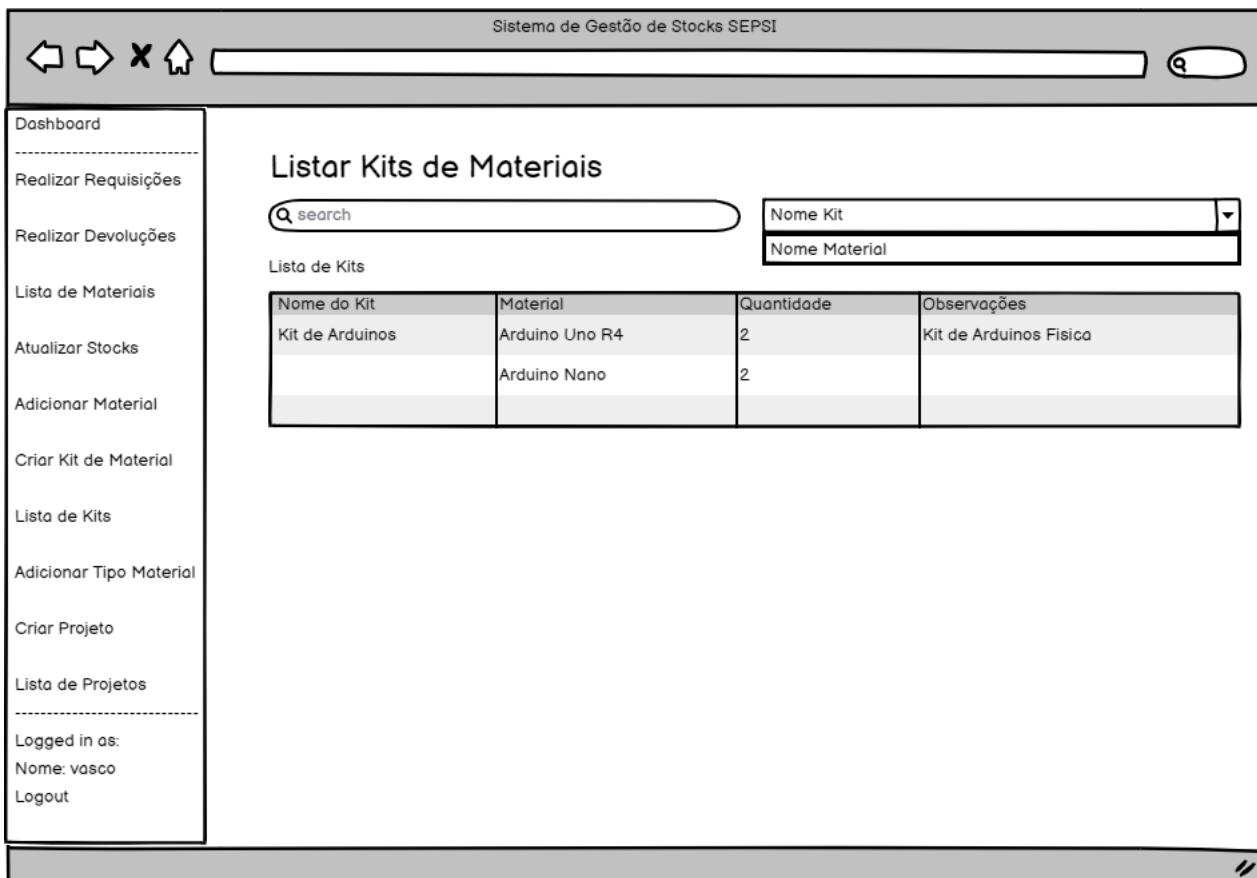
Observações Material Kit

Figura 5.10: Wireframe - Adicionar Kit de Material

#### 5.1.11 Interface para Listar os Kits Materiais

Também deve ser possível criar *kits* de materiais, de modo a conseguir colocar vários materiais num *kit* para que seja possível realizar uma requisição *kit* a *kit* com os materiais já predefinidos, para uma turma ou um projeto.

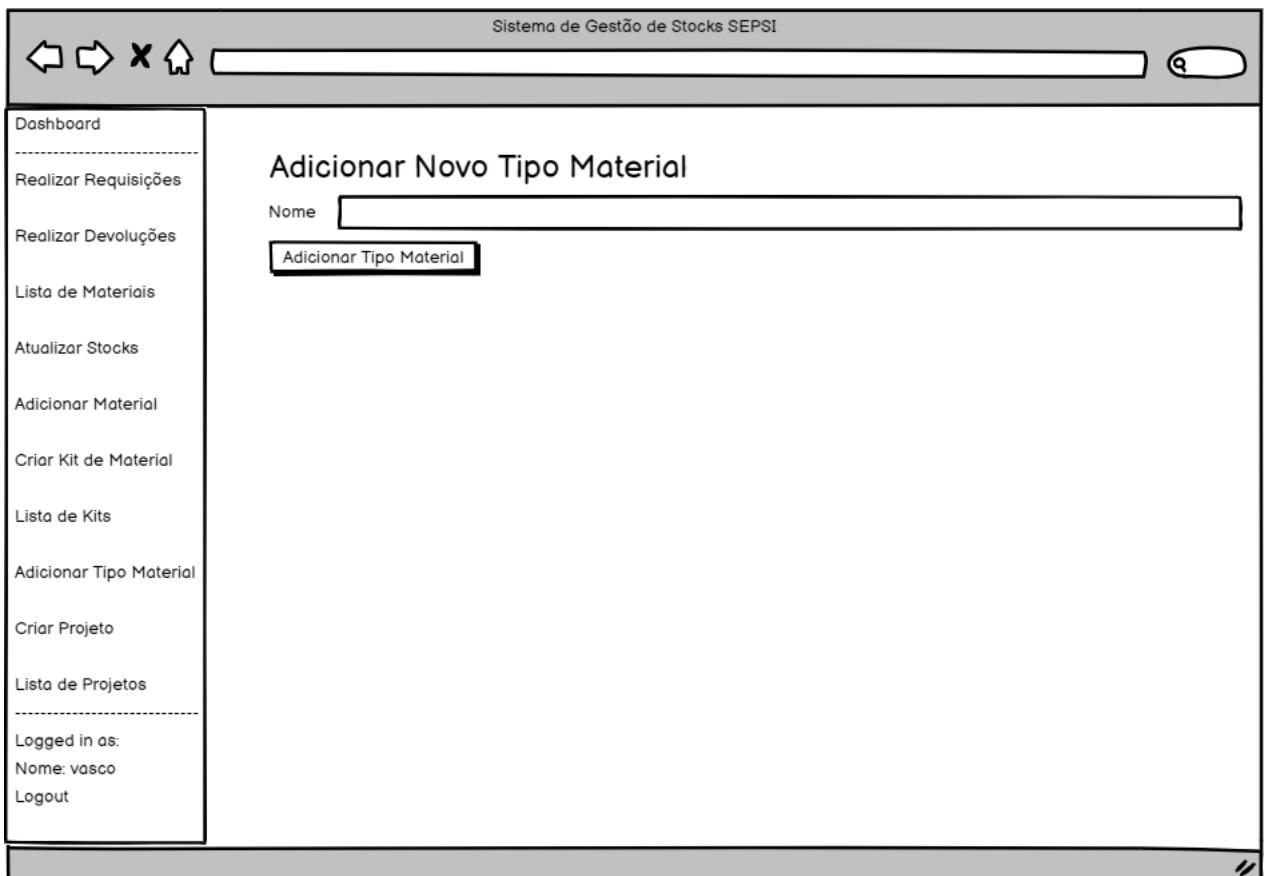
## 5. DESENHO DE INTERFACES GRÁFICAS



**Figura 5.11:** Wireframe - Lista de Kits

### 5.1.12 Interface para Adicionar Tipo de Material

Uma vez que vão existir diversas categorias de materiais num laboratório, estes necessitam de ter um Tipo de Material, como por exemplo Consumíveis, Ferramentas, Materiais.



**Figura 5.12:** Wireframe - Adicionar Tipo de Material

### 5.1.13 Interface para Criar Projeto

Esta interface permite criar projetos em que os materiais foram adequiridos ou que estes estão a ser usados.

## 5. DESENHO DE INTERFACES GRÁFICAS

The wireframe shows a user interface for adding a new project. At the top, there's a header bar with icons for back, forward, close, and search. Below it is a sidebar with a list of navigation options: Dashboard, Realizar Requisições, Realizar Devoluções, Lista de Materiais, Atualizar Stocks, Adicionar Material, Criar Kit de Material, Lista de Kits, Adicionar Tipo Material, Criar Projeto, and Lista de Projetos. A 'Logged in as:' section shows the user 'vasco'. The main content area is titled 'Adicionar Novo Projeto' and contains fields for 'Nome' (Name), 'Observações' (Observations), 'Data Inicio' (Start Date), and 'Data Fim' (End Date). There's also a button labeled 'Adicionar Novo Projeto'.

**Figura 5.13:** Wireframe - Adicionar Projeto

### 5.1.14 Interface para Listar Projetos

Uma vez termos os projetos criados, deve ainda ser possível visualizar a lista de projetos existentes.

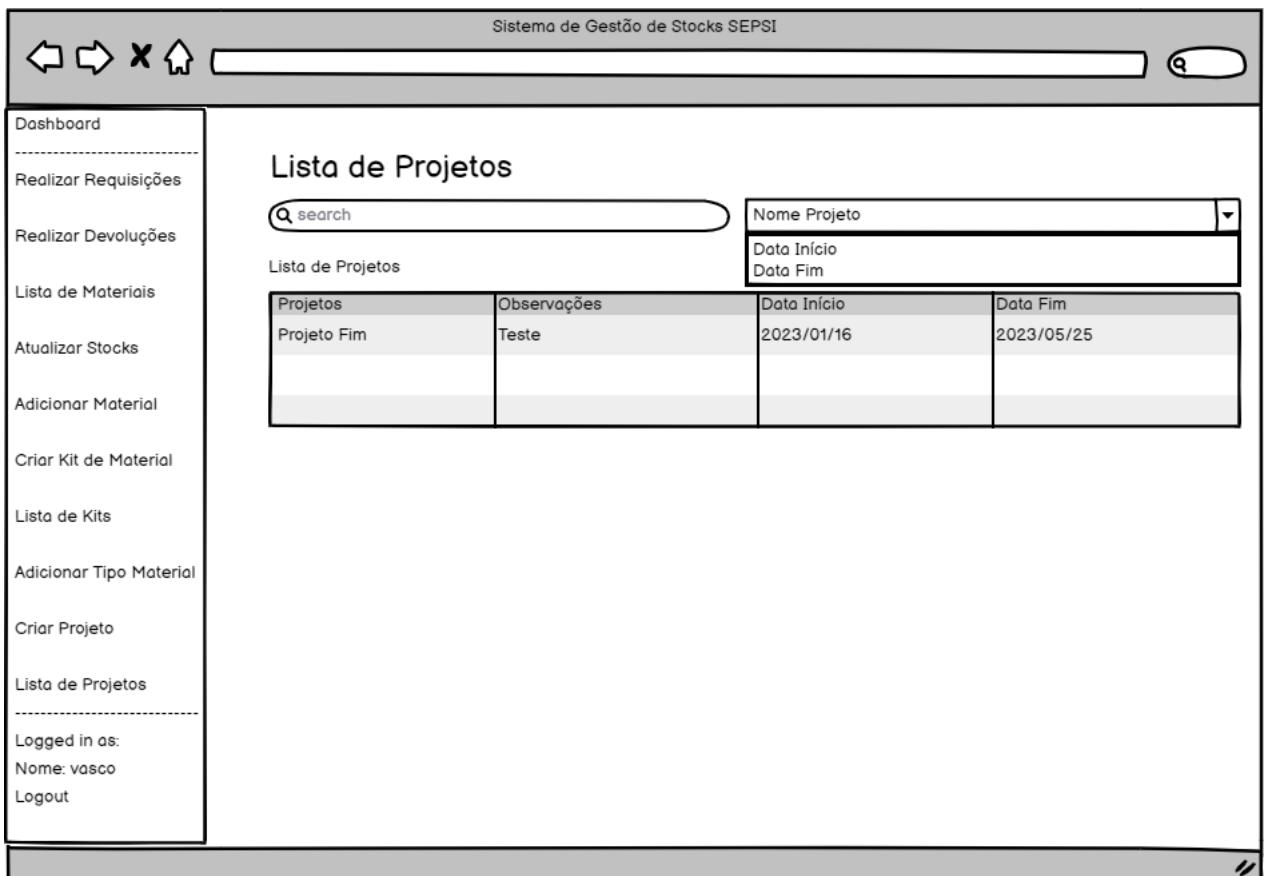


Figura 5.14: Wireframe - Listar Projetos

## 5.2 Storyboards

Após termos visto os wireframes, seguimos para os *storyboards*, que nos vão dar uma perspetiva de como as várias ações do sistema devem ser efetuadas.

### 5.2.1 Gerir listagem de material de laboratório

Este é o *storyboard* do caso de uso 1 que ensina como é realizada a tarefa de gestão dos materiais do laboratório.

## 5. DESENHO DE INTERFACES GRÁFICAS

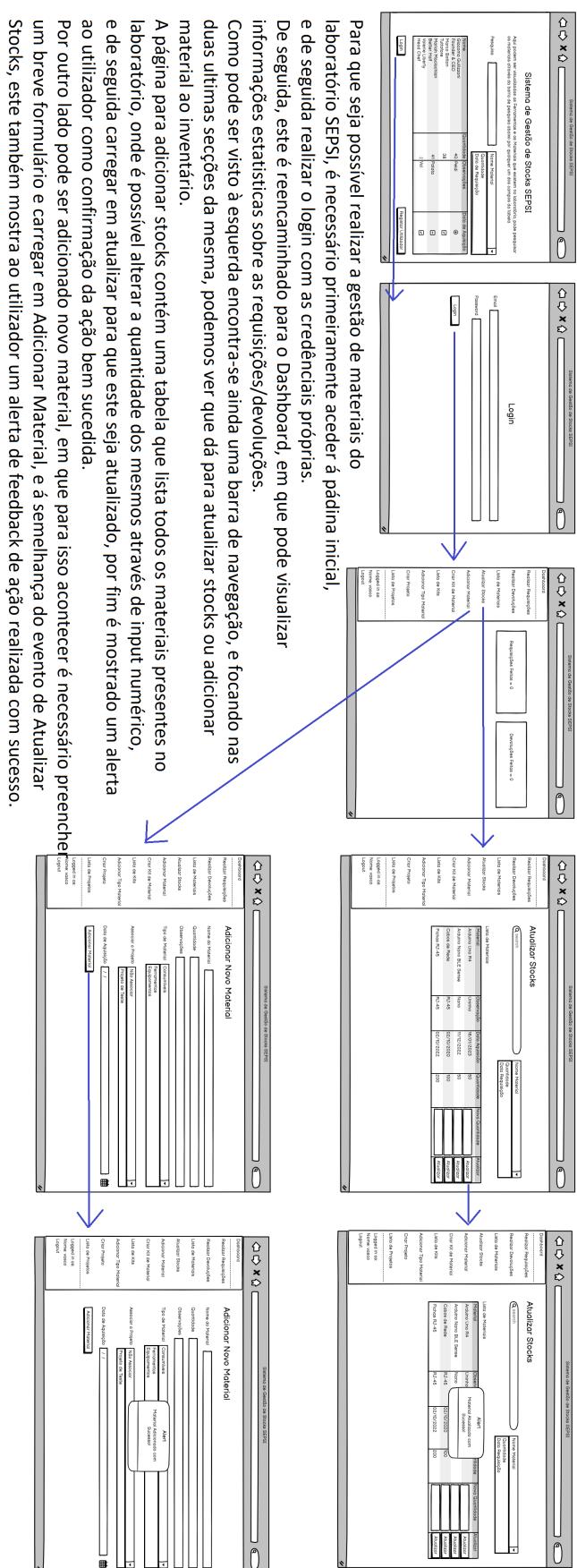
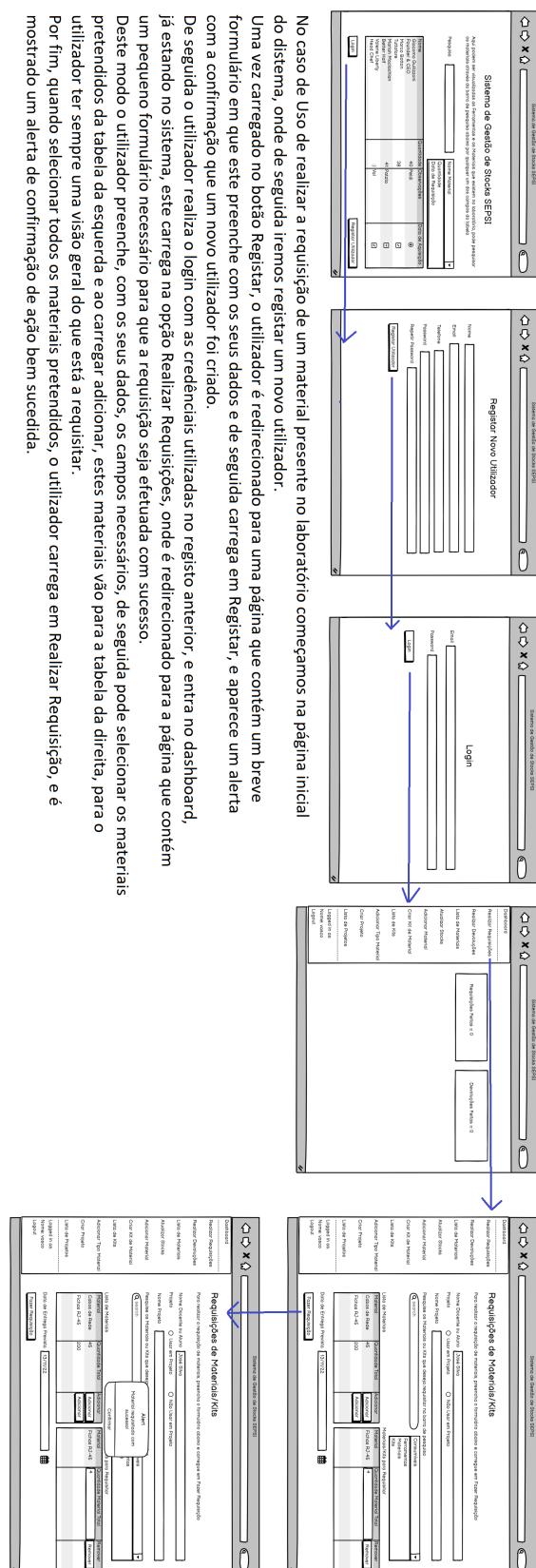


Figura 5.15: Caso de Uso - Gerir Material

### 5.2.2 Fazer requisição de materiais

De seguida, temos o *storyboard* do caso de uso 2 que ensina como é realizada a tarefa de requisição de materiais.e

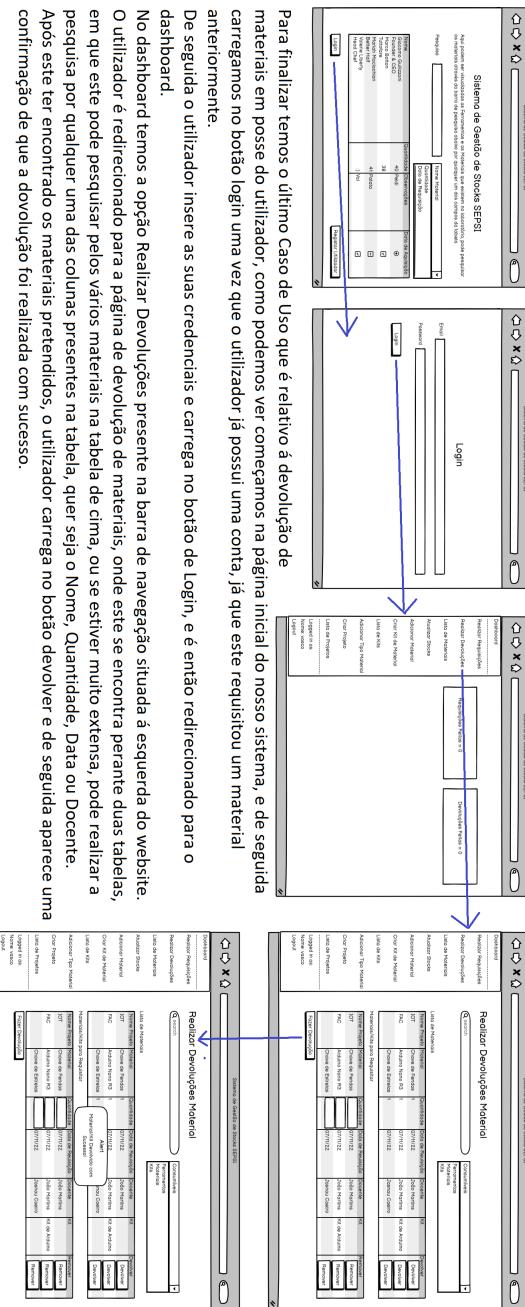
## 5. DESENHO DE INTERFACES GRÁFICAS



**Figura 5.16:** Caso de Uso - Requisitar Material

### 5.2.3 Fazer devolução de materiais

Por último, mas não menos importante, temos o *storyboard* do caso de uso 3 que ensina como é realizada a tarefa de devolução de materiais.



**Figura 5.17:** Caso de Uso - Devolver Material

Para finalizar temos o último Caso de Uso que é relativo á devolução de materiais em posse do utilizador, como podemos ver começamos na página inicial do nosso sistema, e de seguida carregamos no botão login uma vez que o utilizador já possui uma conta, já que este requisitou um material anteriormente.

De seguida o utilizador insere as suas credenciais e carrega no botão de Login, e é então redirecionado para o dashboard.

No dashboard temos a opção Realizar Devoluções presente na barra de navegação situada á esquerda do website. O utilizador é redirecionado para a página de devolução de materiais, onde esta se encontra perante duas tabelas, em que este pode pesquisar pelos vários materiais na tabela de cima, ou se estiverem muito extensas, pode realizar a pesquisa por qualquer uma das colunas presentes na tabela, quer seja o Nome, Quantidade, Data ou Docente.

Após este ter encontrado os materiais pretendidos, o utilizador carrega no botão devolver e de seguida aparece uma confirmação de que a devolução foi realizada com sucesso.

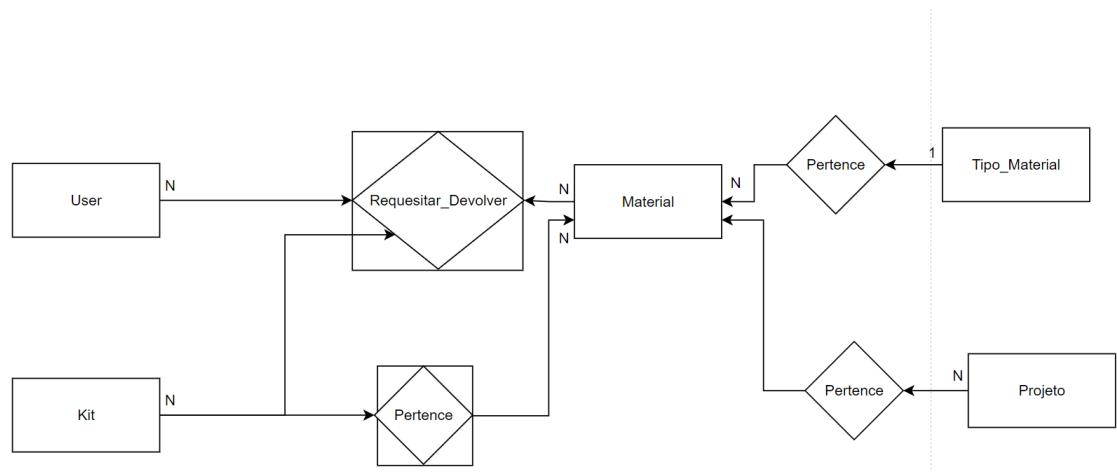


# Capítulo 6

## Desenho da Base de Dados

Este capítulo demonstra o processo de desenhado da base de dados do projeto.

### 6.1 Modelo Conceptual



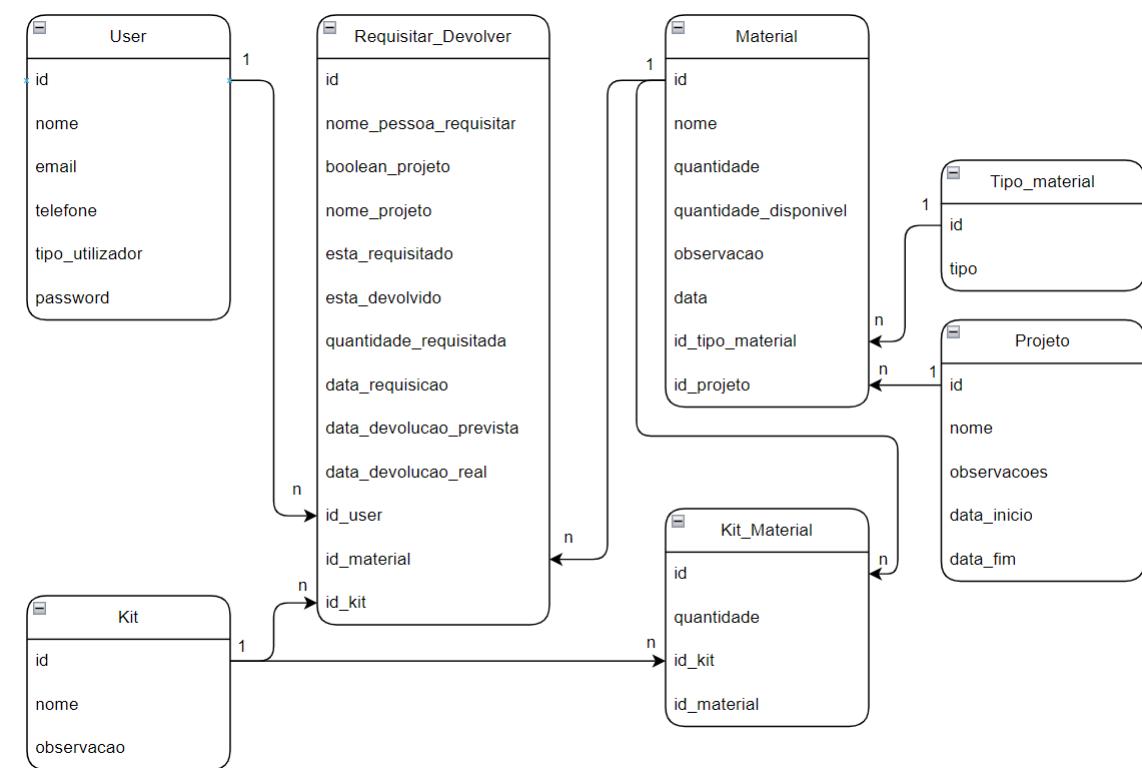
**Figura 6.1:** Modelo Conceptual

### 6.2 Diagrama Entidade-Relação

#### 6.2.1

## 6. DESENHO DA BASE DE DADOS

---



**Figura 6.2:** Diagrama Entidade-Relação

# Capítulo 7

## Decisões de Implementação

Este capítulo demonstra as principais decisões tomadas durante a implementação do projeto.

### 7.1 Back-end

#### 7.1.1 Flask

*Python Flask* é um *micro framework web* escrito em *Python*. É classificado como um *microframework* porque não requer ferramentas ou bibliotecas específicas. Ele não possui camada de abstração de base de dados, validação de formulário ou qualquer outro componente em que bibliotecas de terceiros pré-existentes forneçam funções comuns. O *Flask* é uma estrutura leve, fácil de aprender e usar, tornando-o uma escolha popular para a criação de aplicativos da Web pequenos e simples. Ele também permite fácil integração com outras bibliotecas e estruturas. *Flask* também é popular para construir *APIs RESTful*. [Ron23c]

#### Flask - Bcrypt

*Flask-Bcrypt* é uma extensão do *Flask* que fornece ferramentas de *hash bcrypt* para aplicações *Flask*. *Bcrypt* é um algoritmo de hash de senha considerado muito seguro e desenhado para ser lento, o que torna difícil para os invasores usar força bruta para quebrar senhas com *hash*. O *Flask-Bcrypt* facilita o uso do *bcrypt* em aplicações *Flask*, fornecendo uma interface simples para *hash* e verificação de senhas. Ele também inclui várias opções para configurar o algoritmo de *hash*, como o número de ciclos usados para *hash*. O uso do *Flask-Bcrypt* pode ajudar a manter as senhas, dos seus utilizadores, seguras e protegidas, fazendo *hash* antes de armazená-la na base de dados. [Ron23a]

### Flask - Session

*Flask-Session* é uma extensão do *Flask* que fornece suporte de sessão para aplicações *Flask*. Uma sessão é uma maneira de armazenar dados no lado do servidor para um utilizador específico. Isto é útil para armazenar informações como preferências do utilizador, estado de *login* e outros dados que precisam persistir em várias solicitações. *Flask-Session* fornece uma interface simples para configurar e obter dados de sessão. Também inclui várias opções para configurar a sessão, como o tempo de expiração da sessão.

Por defeito, o *Flask-Session* armazena os dados da sessão na memória do servidor. No entanto, é possível configurá-lo para armazenar os dados da sessão em outros locais, como numa base de dados ou *cache*. *Flask-Session* também oferece suporte a várias formas seguras de armazenar dados de sessão, como *cookie* assinado, criptografar os dados da sessão no lado do cliente. [Ron23b]

## 7.2 Sistema de Gestão de Base de Dados

### 7.2.1 SQLite

Para gestor de base de dados foi escolhido o *SQLite*, uma biblioteca em linguagem C que implementa um motor de base de dados *SQL* bastante rápido e relativamente pequeno em termos de tamanho.

O *SQLite* armazena todos os dados em um único ficheiro no disco, facilitando o gerenciamento e o *backup* da base de dados. O *SQLite* também é leve e rápido, tornando-o adequado para aplicativos de pequeno e médio porte. Também é multiplataforma, o que significa que pode ser usado em vários sistemas operacionais, incluindo *Windows*, *macOS* e *Linux*.

O *SQLite* suporta a maioria dos comandos *SQL* padrão, como *SELECT*, *INSERT*, *UPDATE*, *DELETE* e *CREATE TABLE*. Também oferece suporte a transações, que permitem que várias operações sejam executadas como uma única unidade atômica.

O *SQLite* é amplamente usado em muitas aplicações, como aplicações móveis, navegadores de *Web* e aplicativos de *desktop*. Também é popular entre os desenvolvedores, pois o código é *open-source* e possui uma grande comunidade de utilizadores que contribuem para seu desenvolvimento. [Hip23]

### 7.2.2 SQLAlchemy

Este *toolkit* é um mapeador relacional de objetos que facilita a comunicação entre a base de dados e aplicações *Python*, ela permite traduzir classes em *Python* para tabelas de base de dados relacionais e também converte funções *Python* para instruções *SQL*.

*SQLAlchemy* é uma biblioteca *Python* que fornece um conjunto de *APIs* de mapeamento objeto-relacional e de mapeamento de dados de alto nível para trabalhar com base

de dados relacionais. Ele fornece uma interface comum para interagir com diferentes tipos de base de dados, como *PostgreSQL*, *MySQL*, *SQLite* etc...

O *SQLAlchemy* permite que os desenvolvedores trabalhem com base de dados usando objetos *Python*, em vez de escrever consultas *SQL* tradicionais. Ele fornece um sistema de mapeamento objeto-relacional (*ORM*) poderoso e flexível que permite definir a estrutura de suas tabelas de base de dados como classes *Python* e, em seguida, executar operações *CRUD* (criar, ler, atualizar, excluir) nessas classes. Isso significa que você pode usar as construções internas do *Python*, como classes e objetos, para interagir com base de dados, tornando mais fácil escrever, entender e manter código.

O *SQLAlchemy* também fornece uma *API* de mapeamento de dados de baixo nível que permite trabalhar diretamente com a base de dados usando *SQL*, em vez de *ORM*. Isto é útil para casos de uso mais avançados, como quando você precisa trabalhar com procedimentos armazenados ou executar consultas complexas que não podem ser expressas por meio de *ORM*.

O *SQLAlchemy* também possui suporte integrado para *pooling* de conexões, o que é útil para gerir e reutilizar conexões de base de dados em um ambiente *multithread* e de alto desempenho. [Bay23]

### **SQLAlchemy - Marshmallow**

*SQLAlchemy-Marshmallow* é uma biblioteca que fornece uma maneira simples de integrar a biblioteca *Marshmallow*, que é uma biblioteca popular para validação e serialização de dados, com *SQLAlchemy*.

*Marshmallow* fornece uma maneira simples de definir e validar estruturas de dados complexas, como as usadas em *APIs REST*. Permite que definir um esquema para os dados, que descreve os campos, tipos e regras de validação para esses dados e, em seguida, use esse esquema para validar e serializar os dados.

*SQLAlchemy-Marshmallow* fornece uma maneira simples de integrar *Marshmallow* com *SQLAlchemy* criando esquemas de *Marshmallow* a partir de modelos *SQLAlchemy* e vice-versa. Isso permite que seja possível usar os recursos de validação e serialização do *Marshmallow* com seus modelos *SQLAlchemy*, para que se possa facilmente validar e serializar dados antes de guardá-los na base de dados e também desserializar e validar os dados quando vierem do lado do cliente/utilizador. [som23]

## **7.3 Front-end**

### **7.3.1 React**

Foi escolhido o React para desenvolver o *front-end* da aplicação, devido a nossa experiência com a tecnologia nas unidades curriculares de *TWAM* e *DAW*. *React* funciona com

## **7. DECISÕES DE IMPLEMENTAÇÃO**

---

componentes o que permite construir interfaces complexas apartir de vários componentes separados. [Mc23b]

### **7.3.2 React Router**

Esta libraria para React disponibiliza a programação de rotas no lado do cliente. Isto torna as aplicações mais rápidas uma vez que com apenas um clique em um link, é possivel navegar para outro *URL* e fazer pedidos "fetch"para mostrar essa informação na nova página. [Mc23b]

### **7.3.3 Bootstrap**

Para construir as interfaces gráficas de forma eficiente foi utilizado o *Bootstrap* como *framework*, uma vez que facilita imenso a custumização dos vários elementos gráficos e também ajuda com que toda a parte gráfica da aplicação fique responsiva e se adapte a vários tamnhos de ecrãs. [Mc23a]

# Capítulo 8

## Dificuldades de Implementação

Neste capítulo, iremos mostrar as nossas dificuldades de implementação ao nível do *front-end React* e ao nível do *back-end Flask*.

Existe uma coisa em que eu e o meu colega concordamos, que são as requisições, e as devoluções seriam mais fáceis se não tivessemos os *kits* à mistura, mas isto também não pode ser tudo simples se não isto não tinha piada nenhuma. Dito isto, vamos mostrar como foi implementada uma dessas funcionalidades.

The screenshot shows the 'Requisição de Material' (Material Request) page of the SEPSI system. At the top, there's a navigation bar with the SEPSI logo and the text 'Sistema de Gestão de Laboratório SEPSI'. On the left, a sidebar lists various operations: Requisições, Devoluções, Lista de Materiais, Atualizar Stocks, Adicionar Material, Criar Kit de Material, Lista de Kits, Adicionar Tipo Material, Criar Projeto, and Lista de Projetos. Below this, it shows 'Logged in as: Nome: teste' and a 'Logout' button. The main content area has a heading 'Requisição de Material' and instructions: 'Para realizar uma requisição efetue uma pesquisa pelo tipo de material que deseja requisitar e pesquise pelo nome do mesmo na caixa de pesquisa. Preencha todos os campos'. It includes fields for 'Nome Docente ou Aluno' (with a placeholder 'Nome'), 'Projeto' (radio buttons for 'Usar em Projeto' and 'Não Usar em Projeto'), 'Pesquisa:' (text input), and a dropdown for 'Consumíveis'. There are two tables: 'Lista de Materiais' (Material, Quantidade Total, Adicionar) and 'Materiais/Kits para Requisitar' (Material, Quantidade Material Total, Adicionar). A date input field 'Data Entrega Prevista' (dd/mm/aaaa) is also present. At the bottom is a blue 'Fazer Requisição' (Make Request) button.

**Figura 8.1:** Interface Requisição

Como pode ser visualizado na figura anterior, temos a interface das requisições que contém os campos necessários para que seja possível efetuar uma requisição, e no meio disso tudo temos uma tabela e mais acima uma barra de pesquisa com a caixa de seleção, e

## 8. DIFICULDADES DE IMPLEMENTAÇÃO

---

é aí que entra a dificuldade, porque se fosse só pesquisar pelos *kits* de materiais, estávamos nós bem, mas a realizar a pesquisa pelos *kits* deu umas dores de cabeça, vamos ver o código no *React* e de seguida passamos para o *Flask*.

```
// search bar
useEffect( effect: () => {
    //checks if search bar has data
    if (searchInput.length > 0) {
        //gets data from the API
        fetch(
            input: `http://localhost:5000/showmaterialsbynamebytype?search=`
                +
                searchInput +
                "&search_type=" +
                typeSearch
        ) Promise<Response>
            .then((res : Response ) => res.json()) Promise<any>
            .then((data) => {
                //sets searchResultList with data that came from the API
                setSearchResultList(data.list_kit_mateirals);
            });
    } else if (searchInput.length <= 0) {
        setSearchResultList( value: []);
    }
}, [deps: [searchInput, typeSearch]]);
```

**Figura 8.2:** Código Requisição React

O código mostrado em cima é bastante simples e não tem nada que saber, agora vamos ao processo de realizar a requisição dos *kits*.

---

```

const makeKitsRequisition = async (e) => {
    //boolean variable to permit the requisition to happen
    let permit = true;
    //loops through requisicaoKitsList
    requisicaoKitsList.map((kitList) => {
        //checks if typeof quantidade is undefined therefore being empty and possibly
        if (typeof kitList.quantidade === "undefined") {
            permit = false;
        }
    });
    //checks if a wrong quantity is input like above the value or a negative number
    if (wrongQuantity === true || permit === false) {
        // show error message
        console.log("quantidades incorretas, n fez commit na db");
        //TODO SHOW ERROR MSG
        //changes wrongQuantity state to false and to true the wrongQuantityFinal
        setWrongQuantity( value: false);
        setWrongQuantityFinal( value: true);
        //unsets the timeout to 3 seconds and changes wrongQuantityFinal state to false
        setTimeout( handler: () => {
            setWrongQuantityFinal( value: false);
        }, timeout: 3000);
    }
    //checks if name input has data or if the requisicaoKitsList is empty
    else if (nome.length <= 0 || requisicaoKitsList.length === 0) {
        //if it puts wrongName state to true and activates the alert for 3 seconds
        setWrongName( value: true);
        //deactivates the alert after 3 seconds
        setTimeout( handler: () => {
    
```

**Figura 8.3:** Código Requisição React

Na figura anterior, podemos verificar a existência de um loop na lista de requisição de *kits*, com o objetivo de verificar se a quantidade está vazia ou não ativando assim um alerta. De seguida não basta apenas verificar se a quantidade está vazia, temos de verificar também se a quantidade está correta, se não estiver é mostrado um alerta de que a quantidade se encontra incorreta, e desaparece após 3 segundos, uma vez que está definido pela função *setTimeout()*.

## 8. DIFICULDADES DE IMPLEMENTAÇÃO

---

```
//deactivates the alert after 3 seconds
setTimeout( handler: () => {
    setWrongName( value: false);
}, timeout: 3000);
} else {
    let project = associatedProject;
    try {
        //Gets the default project name
        if (associatedProject === 1) {
            project = listOfProjects[0].nome;
        }
        //sends data to API
        await httpClient.post( url: "//localhost:5000/makekitsrequest", data: {
            nome,
            project,
            requisicaoKitsList,
            data_entrega_prevista,
        });
        //sets alert to change state after 3 seconds
        setTimeout( handler: () => {
            setAlert( value: (prevState : boolean) => !prevState);
        }, timeout: 3000);
        //Sets Variables to their initial state
        setState();
        //Changes the state of the alert
        setAlert( value: (prevState : boolean ) => !prevState);
    } catch (e) {
        if (e.response.status === 401) {
            alert("Invalid Type Info");
        }
    }
}
```

**Figura 8.4:** Código Requisição React

Já na última figura, podemos observar que caso o nome, e a lista de requisições dos *kits* não esteja vazia, é feita a requisição, enviando os dados necessários para a rota específica da *API* e seguidamente é mostrado um alerta de que a requisição foi bem sucedida.

Por fim podemos ir para o *Flask*, como podemos observar na figura referente ao código do *Flask*, é verificado se a data de entrega prevista é zero, caso seja, é colocada a data do dia, caso contrário é usada a função *strptime* que cria um objeto data a partir de uma *string*.

Seguidamente temos 2 *loops*, em que o 1º percorre todos os *kits*, e por sua vez o 2º *loop* percorre todos os materiais dentro do *kit*, e verifica se estão todos preenchidos e de seguida adiciona o *kit* à base de dados.

---

```

# makes a kit request
@app.route("/makekitsrequest", methods=["POST"])
def make_kits_request():

    data_entrega = ""
    if request.json["data_entrega_prevista"] == 0:
        data_entrega = date.today()
    else:
        data_entrega=datetime.strptime(request.json["data_entrega_prevista"], '%Y-%m-%d')

    # iterates over the kits
    for kit in request.json["requisicaoKitsList"]:
        list_mats_in_kit = Kit_Material.query.filter_by(id_kit=kit["id"]).all()
        kit_material_schema = Kit_MaterialSchema(many=True)
        result = kit_material_schema.dump(list_mats_in_kit)

        # iterates over the materials in each kit
        for mat in result:
            print("material =>, mat)
            new_material_kit_request = Requisitar_Devolver(
                nome_pessoa_requisitar=request.json["nome"],
                boolean_projeto=False if request.json["project"] == "" else True,
                nome_projeto=request.json["project"],
                esta_requisitado=True,
                esta_devolvido=False,
                # material qty multiplied by kit qty
                quantidade_requisitada=(
                    mat["quantidade"] * int(kit["quantidade"])),
                data_requisicao=datetime.now(),
                data_devolucao_prevista=data_entrega,
                data_devolucao_real=None,
                id_user=session.get("user_id"),
                id_material=mat["id_material"],
                id_kit=mat["id_kit"]
            )
            db.session.add(new_material_kit_request)
            db.session.commit()

            # removes kit quantity from database
            # Remove quantity from database
            mat_id = mat["id_material"]
            mat_qty = mat["quantidade"] * int(kit["quantidade"])
            new_mat_qty = Material.query.filter_by(id=mat_id).first()
            new_mat_qty.quantidade = new_mat_qty.quantidade - int(mat_qty)
            new_mat_qty.quantidade_disponivel = new_mat_qty.quantidade_disponivel - int(mat_qty)
            db.session.commit()

    return jsonify({
        "": ""
    })

```

**Figura 8.5:** Código Requisição Flask



# Capítulo 9

## Testes com Utilizadores

Este capítulo demonstra as estatísticas resultantes dos testes de utilizadores.

### 9.1 Participantes

Entrevistámos 5 participantes para realizar os testes.

### 9.2 Moderador de testes

O moderador dos testes foi o aluno Vasco Gomes.

### 9.3 Local dos testes

Os testes foram efetuados na sala G8 da ESTIG.

### 9.4 Recolha de dados (Questionário)

Os dados que foram recolhidos são o tempo de execução de cada tarefa assim como a opinião de cada participante dos testes acerca da maneira como são realizadas as tarefas e a sua facilidade de execução. Tudo isto irá ser documentado através da gravação de vídeos com a autorização dos participantes.

### 9.5 Questionários

O questionário foi segmentado em várias partes, sendo estas:

1. Realizar o registo no sistema de gestão de *stocks SEPSI*
2. Realizar o login no sistema com os dados do registo

## 9. TESTES COM UTILIZADORES

---

3. Classificar em termos de facilidade a tarefa de criação de um novo projeto.
4. Classificar em termos de facilidade a tarefa de visualizar de todos os projetos.
5. Classificar em termos de facilidade a tarefa de criação de um novo tipo de material.
6. Classificar em termos de facilidade a tarefa de adicionar um novo material, tanto por preenchimento do formulário ou a partir de um ficheiro *CSV*.
7. Classificar em termos de facilidade a tarefa de atualizar os stocks dos materiais.
8. Classificar em termos de facilidade a tarefa de visualizar todos os materiais.
9. Classificar em termos de facilidade a tarefa de criação de kits de materiais.
10. Classificar em termos de facilidade a tarefa de visualizar todos os *kits* existentes.
11. Classificar em termos de facilidade a tarefa de realizar uma requisição.
12. Classificar em termos de facilidade a tarefa de realizar uma devolução.

De uma forma resumida podemos afirmar que os testes com utilizadores foram um sucesso, dado que todos os utilizadores conseguiram efetuar todas as tarefas com sucesso e estes consideraram o sistema simples e fácil de usar, embora não estivessem habituados ao mesmo.

Por fim foi ainda colocada a questão a todos os participantes que realizaram o teste se acharam o sistema simples e fácil de usar, ao que obtivemos respostas positivas.

Por fim abaixo encontram-se todos os links dos vídeos.

[https://youtu.be/uVXYw\\_Dnk2c](https://youtu.be/uVXYw_Dnk2c) - Teste de Utilização 1  
<https://youtu.be/iJ225kCTiFY> - Teste de Utilização 2  
<https://youtu.be/3Rdg4q-bL9g> - Teste de Utilização 3  
<https://youtu.be/RabipCXhrjg> - Teste de Utilização 4  
<https://youtu.be/g8PNknGOSMI> - Teste de Utilização 5

## 9.6 Resultados

Os resultados podem ser observados através da tabela seguinte.

Utilizador	1	2	3	4	5	Min	Média	Máximo	Desvio Padrão
Tempo Tarefa 1	00:41	00:43	00:26	00:39	00:26	00:26	00:35	00:43	0,005789351
Nº de Clicks Tarefa 1	7	7	7	7	7	7	7	7	0
Tempo Tarefa 2	00:08	00:20	00:10	00:19	00:13	00:08	00:14	00:20	0,003707319
Nº de Clicks Tarefa 2	4	4	4	4	4	4	4	4	0
Tempo Tarefa 3	00:39	00:37	00:32	00:45	00:34	00:32	00:37	00:45	0,003492993
Nº de Clicks Tarefa 3	7	9	7	6	6	6	7	9	1,224744871
Tempo Tarefa 4	00:03	00:08	00:04	00:04	00:01	00:01	00:04	00:08	0,001770493
Nº de Clicks Tarefa 4	1	1	1	1	1	1	1	1	0
Tempo Tarefa 5	00:37	00:56	00:27	00:33	00:29	00:27	00:36	00:56	0,008062736
Nº de Clicks Tarefa 5	8	9	7	8	8	7	8	9	0,707106781
Tempo Tarefa 6	02:16	02:06	01:25	00:51	01:36	00:51	01:38	02:16	0,023567157
Nº de Clicks Tarefa 6	31	25	20	6	23	6	21	31	9,300537619
Tempo Tarefa 7	00:14	00:33	00:16	00:11	00:28	00:11	00:20	00:33	0,006635488
Nº de Clicks Tarefa 7	3	4	3	3	7	3	4	7	1,732050808
Tempo Tarefa 8	00:15	00:13	00:03	00:05	00:02	00:02	00:07	00:15	0,004155076
Nº de Clicks Tarefa 8	2	2	1	1	1	1	1,4	2	0,547722558
Tempo Tarefa 9	01:02	00:53	00:30	00:35	00:34	00:30	00:42	01:02	0,00966501
Nº de Clicks Tarefa 9	13	9	7	8	18	7	11	18	4,527692569
Tempo Tarefa 10	00:08	00:05	00:04	00:02	00:03	00:02	00:04	00:08	0,001598731
Nº de Clicks Tarefa 10	1	1	1	1	1	1	1	1	0
Tempo Tarefa 11	01:02	01:30	00:33	00:49	00:40	00:33	00:54	01:30	0,015601062
Nº de Clicks Tarefa 11	11	18	9	9	11	9	11,6	18	3,714835124
Tempo Tarefa 12	00:39	00:44	00:38	00:27	00:29	00:27	00:35	00:44	0,00497389
Nº de Clicks Tarefa 12	11	7	11	8	5	5	8,4	11	2,607680962

Figura 9.1: Tabela - Testes com Utilizadores

## 9.7 Análise dos dados

Por fim conforme podemos visualizar na tabela da secção acima, e dos testes com os utilizadores, podemos retirar algumas conclusões acerca das tarefas desempenhadas pelos participantes que realizaram os testes, sendo estas a facilidade de realizar a maior parte das tarefas.

Notou-se alguma demora na habituação ao uso conjugado da barra de pesquisa e da caixa de seleção de tipos de materiais, mas após as tarefas estarem executadas, os participantes do teste afirmaram que por ser um design novo e fora do normal não estariam tão habituados, embora depois de se fazer a tarefa mais 2 ou 3 vezes já achariam normal.



## Capítulo 10

# Melhorias Futuras

Neste capítulo, vamos apresentar as melhorias que podem vir a ser feitas, mas não foram, devido a limitações de tempo por estarmos no final de semestre e termos outras unidades curriculares a realizarem avaliações e projetos, projetos esses que são semelhantes a este que estamos a apresentar no caso da unidade curricular de Desenvolvimento de Aplicações Web.

Dito isto as melhorias que podem vir a ser feitas serão:

1. **Paginação nas Tabelas** -> Uma vez que a quantidade de materiais se torne muito extensa é necessário efetuar um scroll até ao fim da página mesmo depois de aplicados filtros, devendo por isso mesmo existir uma paginação.
2. **Separação de Projetos de Investigação e Comuns** -> O professor João Martins achou pertinente, e nós concordamos o facto de existirem uma separação entre estes 2 tipos de projetos para que não se misturem.
3. **Expansão do Sistema de Gestão do Laboratório** -> Com muita pena nossa, e devido á falta de tempo, uma vez que teríamos que reformular toda a base de dados numa altura bastante avançado do projeto, não conseguimos efetuar esta adição de uma possível expansão para diversos laboratórios utilizarem este mesmo sistema, embora nos pareça bastante bem pensado, uma vez que esta abordagem já havia sido lecionada pela professora Isabel Brito na unidade curricular de Engenharia de Software.



# Capítulo 11

## Conclusão

Para concluir podemos afirmar que foi bastante gratificante realizar este projeto, uma vez que podemos dar uso às diversas matérias lecionadas nas diferentes unidades curriculares do nosso curso, assim como foram cumpridos todos os objetivos pretendidos, sendo estes ter um sistema de gestão de materiais do laboratório *SEPSI* com as funcionalidades pedidas pelo professor João Martins, sendo estas:

1. Requisição de Materiais/Kit
2. Devolução de Materiais/kit
3. Gestão de Stocks do Laboratório



# Bibliografia

- [Bay23] Michael Bayer. *SQLAlchemy documentation*. <https://www.sqlalchemy.org/>. Consultado em 2023/01/16. 2023 (citado na página 37).
- [Hip23] D. Richard Hipp. *Flask - SQLite documentation*. <https://www.sqlite.org/index.html>. Consultado em 2023/01/16. 2023 (citado na página 36).
- [Mc23a] Meta e community. *Bootstrap documentation*. <https://getbootstrap.com/>. Consultado em 2023/01/16. 2023 (citado na página 38).
- [Mc23b] Meta e community. *React Router documentation*. <https://reactrouter.com/en/main>. Consultado em 2023/01/16. 2023 (citado na página 38).
- [myL23] myLIMS. *myLIMS*. <https://mylims.net/lims/>. Consultado em 2023/01/24. 2023 (citado na página 4).
- [Ron23a] Armin Ronacher. *Flask - Bcrypt documentation*. <https://flask-bcrypt.readthedocs.io/en/1.0.1/>. Consultado em 2023/01/16. 2023 (citado na página 35).
- [Ron23b] Armin Ronacher. *Flask - Session documentation*. <https://flask-session.readthedocs.io/en/latest/>. Consultado em 2023/01/16. 2023 (citado na página 36).
- [Ron23c] Armin Ronacher. *Flask documentation*. <https://flask.palletsprojects.com/en/2.2.x/>. Consultado em 2023/01/16. 2023 (citado na página 35).
- [som23] someone. *SQLAlchemy - Marshmallow documentation*. <https://marshmallow-sqlalchemy.readthedocs.io/en/latest/>. Consultado em 2023/01/16. 2023 (citado na página 37).

Documento elaborado com base no *template for final reports and dissertations (Instituto Politécnico de Beja)*, disponível em <https://www.overleaf.com/project/5d936b9ea273390001434a37>, Version 0.9, 2021/12/01, Autor: João Paulo Barros, joao.barros@ipbeja.pt