



**ISEL**  
INSTITUTO SUPERIOR DE  
ENGENHARIA DE LISBOA

## Projeto Final de Curso

Licenciatura de Engenharia Informática, Redes e Telecomunicações  
Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

2021/22

Verão

---

# UX-IOT

## INTERFACE DE UTILIZADOR DE ELEMENTO IOT

---

Gustavo Campos (A47576) e Tiago Cebola (A47594)

Orientadores: Luís Osório e Carlos Gonçalves

26 DE JUNHO DE 2022



# Índice

---

Índice.....	i
Lista de Figuras .....	iii
Lista de Abreviaturas, Acrónimos e Siglas.....	v
Resumo .....	vii
Abstract .....	ix
1. Introdução .....	1
2. Estado da Arte .....	3
2.1. Linguagem de Modelação SysML .....	3
2.2. A Norma Componentes Web .....	4
2.3. Quadro de desenvolvimento Web - Vaadin .....	5
2.4. Fundamentos JSON .....	5
2.5. REST API.....	6
2.6. O tipo de arquivo SVG .....	6
2.7. A biblioteca Lit .....	7
2.8. A linguagem HTML .....	7
2.9. A linguagem CSS .....	7
2.10. A linguagem JavaScript .....	8
2.11. A linguagem TypeScript.....	8
2.12. Protocolo ONVIF.....	9
3. Análise do domínio do problema UX-IoT.....	11
3.1. Requisitos Funcionais e Não Funcionais .....	11
3.2. Casos de Uso .....	13
3.3. Modelo de Dados.....	14
3.3.1. Entidade PESSOA .....	15
3.3.2. Entidade Widget_IoT .....	15
3.4. Contextos de Aplicação .....	17

4.	Desenvolvimento do Sistema UX-IoT .....	19
4.1.	Autenticação.....	20
4.2.	Registo de Utilizadores .....	21
4.3.	Registo de Widget IoT .....	23
4.4.	Widgets-IoT .....	24
4.5.	Simulador de Câmara .....	26
4.6.	Operações sobre um Sinótico.....	26
5.	Conclusões .....	29
5.1.	Pesquisa e Trabalho Futuro.....	30
6.	Bibliografia .....	33
7.	Apêndices .....	37
7.1.	Apêndice 1 .....	37

## Lista de Figuras

---

Figura 1 – Arquitetura de contexto do sistema UX-IoT .....	2
Figura 2 – Comparação entre SysML e UML.....	4
Figura 3 – Requisitos Funcionais .....	12
Figura 4 – Requisitos não Funcionais.....	12
Figura 5 – Diagrama casos de uso .....	13
Figura 6 – Modelo EER simplificado.....	14
Figura 7 – Entidade PESSOA do modelo de dados .....	15
Figura 8 – Entidade Widget_IoT do modelo de dados.....	16
Figura 9 – Entidade SINÓTICO do modelo de dados.....	16
Figura 10 – Relação SINOP_WIDG do modelo de dados .....	17
Figura 11 – Modelo que representa as partes do sistema.....	18
Figura 12 – Painel de controlo do sistema UX-IoT .....	19
Figura 13 – Formulário de autenticação .....	20
Figura 14 – Fluxograma do processo de autenticação.....	21
Figura 15 – Formulário de registo de utilizadores .....	22
Figura 16 – Fluxograma do processo de registo de pessoas .....	23
Figura 17 – Formulário de registo de widgets IoT .....	24
Figura 18 – Componente Web (Câmara).....	25
Figura 19 – Menu de contexto do Componente Web .....	25
Figura 20 – Vista do Quadro Sinótico .....	27



## Lista de Abreviaturas, Acrónimos e Siglas

---

<b>API</b>	Application Programming Interface
<b>CGI</b>	Common Gateway Interface
<b>CRUD</b>	Create, Read, Update and Delete
<b>CSS3</b>	Cascading Style Sheets version 3
<b>DOM</b>	Document Object Model
<b>GUI</b>	Graphical User interface
<b>HTTP</b>	Hypertext Transfer Protocol
<b>HTTPS</b>	Hypertext Transfer Protocol Secure
<b>HTML5</b>	HyperText Markup Language version 5
<b>IP</b>	Internet Protocol
<b>JSON</b>	JavaScript Object Notation
<b>ONVIF</b>	Open Network Video Interface Forum
<b>REST</b>	Representational State Transfer
<b>RGB</b>	Red, Green and Blue
<b>RGBA</b>	Red, Green, Blue and Alpha
<b>RTSP</b>	Real Time Streaming Protocol
<b>SQL</b>	Structured Query Language
<b>SSE</b>	Server Side Event
<b>SVG</b>	Scalable Vector Graphics
<b>TCP</b>	Transmission Control Protocol
<b>TLS</b>	Transport Layer Security
<b>UDP</b>	User Datagram Protocol
<b>UI</b>	User Interface
<b>UML</b>	Unified Modeling Language

<b>URI</b>	Uniform Resource Identifier
<b>URL</b>	Uniform Resource Locator
<b>XML</b>	Extensible Markup Language



## Resumo

---

A Internet das Coisas configura um novo paradigma que transforma o dia a dia dos seus utilizadores elevando o padrão de vida através da tecnologia. As coisas enquanto dispositivos, funcionam de forma interligada tirando proveito da rede internet a qual permite a troca de dados entre dispositivos. Os dados são representados como informações de monitorização e rastreamento e a comunicação é realizada através de sensores que, por sua vez, torna os dispositivos inteligentes ao ponto de permitir automatizar tarefas.

O Componente Web refere-se a uma tecnologia recente que permite o desenvolvimento de elementos personalizados podendo ser reutilizados em diversos documentos na Web. Esta tecnologia permite que o seu desenvolvimento seja isolado do resto da estrutura de um sistema, permitindo não só a sua reutilização como referido acima, como também a sua independência dos quadros de desenvolvimento, fazendo com que os *browsers* consigam interpretar estes elementos sem uma biblioteca específica.

Foi objetivo para este projeto desenvolver uma representação gráfica abstrata de um dispositivo da rede da Internet das Coisas a partir de Componentes Web. As representações gráficas tiveram ainda o intuito de serem inseridas como elementos de uma interface que permite aos utilizadores interagirem com os dispositivos e manipulá-los através da alteração dos seus atributos e especificações.



## Abstract

---

The Internet of Things configures a new paradigm that transforms the daily life of its users raising the standard of life through technology. Things, as devices, work in an interconnected way taking advantage of the Internet network which allows the exchange of data between devices. The data is represented as monitoring and tracking information and this communication is carried out through sensors which in turn makes the devices intelligent to the point of allowing tasks to be automated.

The Web Component refers to a recent technology that allows the development of customized elements that can be reused in several web documents. This technology allows its development to be isolated from the rest of a system's structure, allowing not only its reuse as mentioned above but also its independence from development frameworks, making browsers able to interpret these elements without a specific library.

This project aimed to develop an abstract graphical representation of an Internet of Things network device from Web Components. The graphical representations were also intended to be inserted as elements of an interface that allows users to interact with the devices and manipulate them by changing the value of the attributes.



# 1. Introdução

---

O relatório apresenta e discute o projeto final de curso denominado de Experiência de Utilizador – Internet das coisas, ou no inglês *User Experience – Internet of Things* ao qual atribuímos a sigla UX-IoT.

O projeto enquadra-se num contributo para melhorar, tal como o nome indica, a experiência de utilizadores, oferecendo uma interface funcional que responda em tempo real às ações realizadas sobre dispositivos da rede da Internet das Coisas, do inglês *Internet of Things* (IoT) [1, 2]. Atualmente, os sistemas que incorporam mecanismos de interação com dispositivos IoT encontram-se limitados, pese embora exista tecnologia disponível. Por outro lado, a utilização de interfaces para dispositivos IoT é complexa, pelo que se torna essencial simplificar o processo de interação, oferecendo ao utilizador mecanismos de utilização simples.

O projeto teve como principal objetivo a conceção de uma interface web para utilizadores de sistemas de gestão de quadros tecnológicos envolvendo equipamentos ciberfísicos [3]. Entendemos por equipamento ciberfísico, um sistema que possui uma parte que comunica com o mundo físico e outra parte computacional com interface para o mundo ciber. Unindo os conceitos de equipamento ciberfísico e IoT, concluímos que, apesar destes possuírem origens distintas, ambas se referem a integração com o mundo físico, como a capacidade computacional e a conectividade de rede. Assim sendo, a utilização de ambos os termos ao longo do relatório remetem para a mesma classe de objetos.

Foi ainda objetivo utilizar tecnologias mais recentes, de modo a modernizar não só o conteúdo gráfico, como também o código fonte com o intuito de estabelecer uma base para desenvolvimento futuro ou até a integração do projeto como contributo em novos produtos do mercado. A adoção de abordagens mais atualizadas, como por exemplo, os Componentes Web referidos anteriormente, permitem a sua reutilização noutros sistemas, onde poderão ter o papel de incrementar a eficácia do desenvolvimento, uma vez que não será necessário a criação e utilização de novos recursos na interação dos utilizadores com dispositivos IoT/Ciberfísicos.

A nossa abordagem considera componentes gráficas web na interação de

utilizadores com dispositivos IoT, dispostos no que designamos por quadro sinótico.

A Figura 1 ilustra a experiência de utilização, na qual considera uma pessoa que pretende interagir com um elemento gráfico inserido num quadro sinótico pertencente à interface e que representa virtualmente um equipamento ciberfísico que pode ser de diversos tipos, tais como câmaras ou silos, como demonstrado na figura.

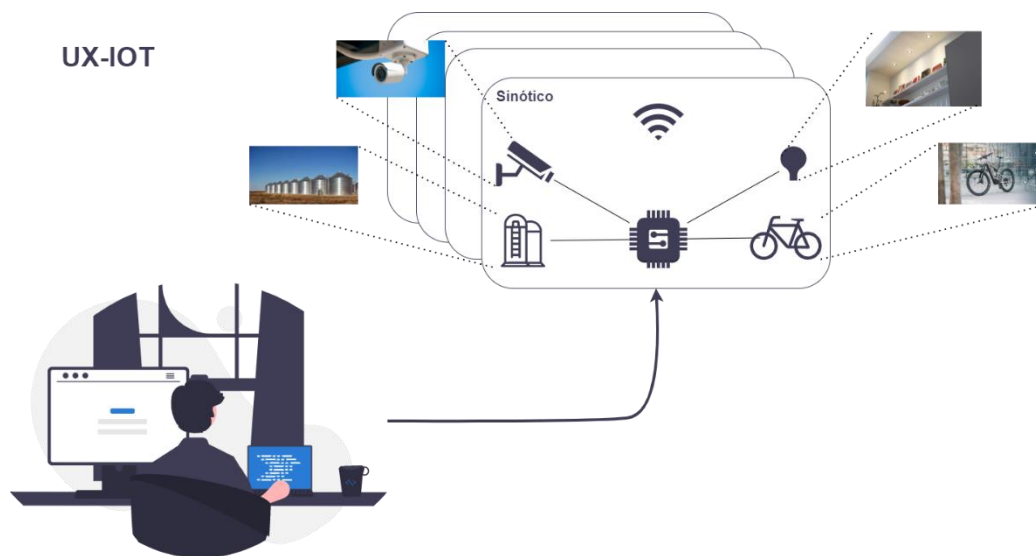


Figura 1 – Arquitetura de contexto do sistema UX-IoT

O restante documento foi organizado segundo a seguinte estrutura:

- O capítulo 2 aborda conceitos teóricos necessários para a estruturação e implementação do projeto, dando ao leitor um contexto e entendimento sobre as abordagens adotadas ao longo do projeto.
- O capítulo 3 apresenta uma análise do problema onde são apresentados e discutidos os requisitos funcionais e não funcionais, os tipos de diagramas utilizados para definir estruturas e a utilização de um modelo relacional para a criação dos dados.
- No último capítulo, é apresentado e discutido o demonstrador do projeto com as funcionalidades implementadas, assim como a definição de futuras implementações com melhorias a serem introduzidas no sistema.

## 2. Estado da Arte

---

Neste capítulo são apresentados os principais conceitos para o desenvolvimento do projeto, nomeadamente a linguagem *System Modeling Language* (SysML) [4], utilizada para conceber o modelo do sistema UX-IoT referida na secção 2.1.

A secção 2.2 aborda a norma para o desenvolvimento dos Componentes Web que apresenta as especificações necessárias para conceber componentes independentes de qualquer *framework* [5]. O quadro de desenvolvimento escolhido para o desenvolvimento do sistema foi o Vaadin, sendo que a secção 2.3, aborda e discute a razão pela sua escolha.

A secção 2.4 apresenta os fundamentos do modelo JSON que pode ser utilizado na troca de dados entre o cliente e o servidor. Foi planeado na nossa concepção de que os Componentes Web iriam utilizar imagens do tipo SVG na sua composição sendo este tipo de ficheiro apresentado na secção 2.6. Outra das nossas decisões para o desenvolvimento dos Componentes Web foi a utilização da biblioteca Lit, apresentada na secção 2.7.

As secções 2.8, 2.9, 2.10 e 2.11 abordam as linguagens utilizadas ao longo do desenvolvimento do sistema, respetivamente, HTML, CSS, JavaScript e TypeScript. Por fim, a secção 2.12 apresenta o protocolo ONVIF, utilizado em comunicação com algumas câmaras de vídeo.

### 2.1. Linguagem de Modelação SysML

O SysML é uma linguagem de modelação gráfica estabelecido pelo *standart* ISO/IEC 19514:2017 [6] de uso geral para conceber, especificar, analisar, projetar e validar sistemas complexos que podem incluir *hardware*, *software*, dados, processos e/ou pessoas. A linguagem SysML propõe unificar as diversas linguagens de modelação utilizadas, para facilitar a comunicação dentro de uma equipa composta por engenheiros mecânicos, eletrotécnicos, informáticos e de outras especialidades.

Esta linguagem é baseada num subconjunto mínimo da *Unified Modeling Language* (UML) [7], como é apresentado na Figura 2 a partir do diagrama de Venn. As suas diferenças passam pela remoção de diagramas originalmente concebidos para modelação de aspetos do desenvolvimento de software, introduzindo o conceito de Bloco, aplicado a qualquer domínio e sistema. Uma das melhorias da linguagem SysML em relação ao UML é o suporte para representar os requisitos e relacioná-los com os modelos do sistema, os testes e o projeto real.

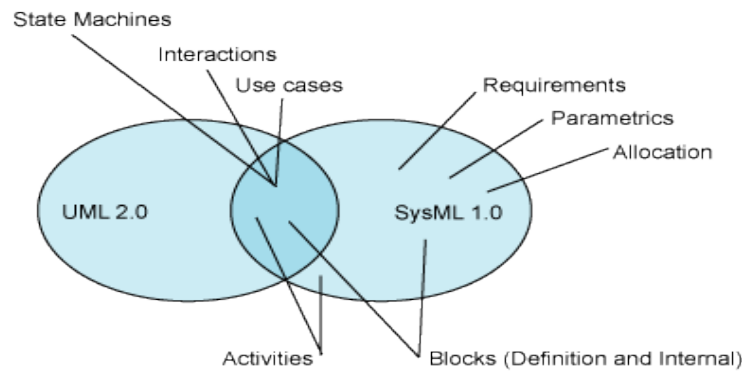


Figura 2 – Comparação entre SysML e UML

## 2.2. A Norma Componentes Web

A especificação aberta Componentes Web [8, 9] consiste num conjunto de cinco conceitos principais: `Template`, `Shadow DOM`, `Custom Element`, `HTML Import` e `Decorator`, que no seu conjunto visam o desenvolvimento de componentes, tornando o código mais simples, organizado e modular.

- **Template:** Representa uma sub-árvore no modelo de objetos num *web browser*, Document Object Model (DOM) [10] que por sua vez permanece estática e é incluída na árvore principal, quando acionada pelo JavaScript [11].
- **Shadow Dom:** Estabelece o isolamento de elementos do quadro DOM em trechos independentes evitando que estilos CSS [12] e comportamentos JavaScript dos elementos alterem as características quando em diferentes domínios DOM.
- **Custom Element:** Permite o desenvolvimento de novos elementos ou *tags* de forma a incluir e conectar *tags* nativas, de forma personalizada e reutilizável. Estes elementos personalizados podem herdar as características e propriedades das *tags* nativas.
- **HTML Import:** Possibilita importar e reutilizar documentos HTML auxiliando no encapsulamento dos componentes em arquivos externos.
- **Decorator:** Permite aplicar o `Template` com base em seletores CSS e JavaScript para criar mudanças não só visuais, como também no seu comportamento.



## 2.3. Quadro de desenvolvimento Web - Vaadin

O quadro de desenvolvimento Vaadin [13] estabelece um quadro de código aberto para desenvolvimento de aplicações Web que atua do lado do servidor. Isto é, a maior parte da sua lógica computacional é executada no servidor, mas também possui Componentes Web pré-criados para serem utilizados no lado do cliente de forma a assegurar uma experiência rica e interativa, com suporte ao kit de ferramentas Web do Google, no inglês, *Google Web Toolkit* (GWT) [14]. Uma das vantagens do Vaadin é a possibilidade de utilizar Java como a linguagem de programação para criar conteúdo Web ou então Typescript com a nova vertente de desenvolvimento promovida pela empresa Vaadin denominada Hilla.

O quadro de desenvolvimento Web Vaadin incorpora a programação orientada a eventos e *widgets*, o que possibilita um modelo de programação similar à programação utilizada para a área de trabalho de interface gráfica de utilizador, ou no inglês, *Graphical User Interface* (GUI) *desktop* [15], que contrasta com o desenvolvimento Web normalmente realizado com base em HTML [16] e JavaScript.

Os Componentes Web que fazem parte da biblioteca disponibilizada pelo quando Vaadin podem ser personalizados visualmente a partir de arquivos CSS. Durante o desenvolvimento, é possível adicionar, se necessário, validações nas ações executadas pelos utilizados em um dado componente.

A escolha do quadro de desenvolvimento Web Vaadin engloba os aspetos referidos acima contudo é de se notar que existem outros quadros de desenvolvimento, nomeadamente o React [17], o Angular [18] e o Vue [19], que possibilitam reutilizar os Componentes Web criados visto que a principal característica destes componentes é serem independentes de qualquer quadro de desenvolvimento Web.

## 2.4. Fundamentos JSON

A especificação *JavaScript Object Notation* (JSON) [20] estabelece um formato baseado em texto padrão, ou seja, de livre acesso e implementação, que é independente de taxas de uso e sem critérios de favorecimento ou discriminação por parte dos seus implementadores. O seu objetivo é representar dados estruturados com base na sintaxe de um objeto na linguagem JavaScript, assumindo a inclusão de dados dos tipos: *string*, *number* e *boolean* na construção de hierarquias. É utilizado para transmitir dados do

servidor para o cliente ou vice-versa, para que possam ser exibidos numa página web ou manipulados para validações por parte do servidor.

## 2.5. REST API

A *Application Programmer Interface* (API) [21] em português, Interface de Programação de Aplicações, representa um conjunto de rotinas, padrões estabelecidos e documentados para que uma determinada aplicação saiba como utilizar os seus métodos. O facto de não se conhecer a lógica computacional utilizada no seu desenvolvimento, permite aos programadores usufruir da API mesmo quando esta passa por um processo de alteração desde que se mantenha a mesma estrutura com que foi construída partindo de bons princípios de programação na sua formulação. Os seus dados são obtidos via *HyperText Transfer Protocol* (HTTP) [22], sendo que as API do tipo *Representational State Transfer* (REST) [23], definem um conjunto de restrições utilizadas para que as trocas de mensagens sigam as diretrizes definidas na arquitetura:

- **Cliente-Servidor:** Modelo que considera um componente designado por cliente a aceder a funcionalidades de outro componente designado por servidor, distribuídos e interligados por uma rede de comunicação
- **Sem Estado/Stateless:** As requisições são feitas de forma independente, ou seja, cada uma executa apenas uma determinada ação;
- **Interface Uniforme:** Agrupa outros quatro conceitos em que determina que os recursos devem ser identificados, a manipulação dos recursos deve ser por meio de representação com mensagens auto-descritivas e a utilização de links para efetuar a navegação pelo sistema.

## 2.6. O tipo de arquivo SVG

A especificação *Scalable Vector Graphics* (SVG) refere-se a um tipo de arquivo que descreve gráficos vetoriais escaláveis e retrata um tipo de imagem que utiliza como formato de texto a linguagem *Extensible Markup Language* (XML) [24].

Este tipo de arquivo pode ser dimensionado para tamanhos diferentes sem perder qualidade, ou seja, o formato é independente da resolução, permitindo reduzir, aumentar e

moldar as imagens sem limites, sempre com a mesma qualidade, além de possuir um baixo custo de armazenamento por ficheiro.

## 2.7. A biblioteca Lit

A biblioteca Lit [25] tem como propósito o desenvolvimento de Componentes Web de forma rápida, simples e leve, o que possibilita renderizar apenas certas partes da interface do utilizador sem interferir no sistema como um todo. O Lit fornece uma sintaxe simples do tipo declarativa com suporte tanto a JavaScript como TypeScript, sendo por padrão todos os elementos desenvolvidos a partir desta biblioteca um Componente Web.

## 2.8. A linguagem HTML

A *HyperText Markup Language* (HTML) é uma linguagem de Marcação de Hipertexto, sendo um dos principais elementos utilizados no desenvolvimento web, pois permite aos programadores definir a forma como será exibida os elementos no *browser*, seja através de texto, hiperligações ou arquivos multimédia. A versão mais recente é a 5ª e traz consigo novos recursos como o suporte de áudio e vídeo de alto nível, a inserção de imagens do tipo *Scalable Vector Graphics* (SVG) [26], Canvas [27] e novos elementos de controle para formulário como datas, horas, email, entre outros.

Outra característica introduzida também foi o suporte para a análise sintática de forma padronizada, sendo que as introduções destes novos recursos visam tornar mais fácil a inclusão e a manipulação dos conteúdos sem ter de recorrer a *plugins*, bem como enriquecer o conteúdo semântico das páginas Web.

## 2.9. A linguagem CSS

A *Cascading Style Sheet* (CSS) é uma linguagem do tipo, folha de estilo em cascata e é utilizado para estilizar elementos HTML e retrata a componente visual/estética de um site. Esta linguagem permite que a função de um dado elemento seja isolada da sua parte estética sendo possível reaproveitar o mesmo estilo em vários elementos. Além disso a sua utilização advém de seletores utilizados nos elementos HTML nomeadamente: a classe, o *id*, a *tag* e o nome.

A nova versão é a 3ª que introduz conceitos como a criação de módulos que permite a divisão em grupos de recursos, a criação de animações, interações, responsividade, o uso de um sistema de cores baseado em RGBA (Red, Green, Blue, Alpha) [28] e cores gradiente, tal como também o suporte a mídia/multimédia, tudo isto através de uma sintaxe simples com um desenvolvimento interativo e incremental, tornando-se flexível e de fácil manutenção.

## 2.10. A linguagem JavaScript

O JavaScript é uma linguagem de programação orientada a protótipos e permite a criação de sistemas que se executam inteiramente no *browser*. É atualmente a principal linguagem para programação utilizada para implementações no lado do cliente.

O Javascript possui um modelo híbrido sendo este o mais utilizado no desenvolvimento de objetos em JavaScript e consiste na combinação de dois padrões: o `Constructor` e o `Prototype`.

O padrão `Constructor` define as propriedades de uma instância, enquanto que o padrão `Prototype` define métodos e as propriedades partilhadas entre instâncias, estas por sua vez são objetos que contêm todas as propriedades e métodos que devem estar disponíveis. Estes objetos são literalmente protótipos para os objetos a serem criados assim que o seu construtor seja invocado, ou seja, novos objetos são criados pela cópia do protótipo. Portanto, este modelo apresenta as seguintes vantagens:

- Todas as propriedades e métodos de um protótipo são criados apenas uma vez e partilhados pelas suas instâncias/objetos, o que se traduz num ganho de eficiência em termos de consumo de memória.
- Permite o encapsulamento do código privado, expondo apenas as propriedades e métodos que se pretendam públicos.
- Possibilita a fácil extensão do código sem afetar a implementação existente.

## 2.11. A linguagem TypeScript

O TypeScript é uma linguagem de programação baseada em JavaScript, com a proposta de oferecer novas funcionalidades e ferramentas. Advindo do contexto da

evolução de uma simples linguagem do lado do cliente que era o Javascript, em que eram utilizadas poucas linhas de código para manusear alguns aspetos visuais ao ponto de ganhar destaque e hoje ser o cerne de muitos sistemas. Como tal e de forma a suportar o paradigmas de programação como a orientação a objetos, foi visto a necessidade de um ajuste na linguagem tais como a tipagem e a criação de interfaces.

O Typescript, portanto, aborda os seguintes conceitos:

- Encapsulamento: Forma de estruturação de código para que determinados blocos tenham acesso a pontos específicos para o ambiente externo, ou seja, há visibilidade e acessibilidade dos elementos internos de uma classe.
- Herança: Classes filhas podem herdar, ou não, os comportamentos e características de uma classe pai, sem que seja necessário redefinir todas as funções novamente utilizando a sintaxe *extends*.
- Abstração: Capacidade de destacar apenas algumas características de elementos do mundo real, agrupadas em classes que representam partes de um elemento e os seus atributos para a solução de um determinado problema. As classes abstratas, ou seja, que não possuem representação no mundo real, mas que realizam determinadas funções necessárias ao sistemas são chamadas de interfaces.
- Polimorfismo: Permite que sejam utilizados objetos dentro da programação conforme a situação, ou seja, em contraste com a herança uma classe pai pode utilizar atributos de qualquer das suas classes filhas.

Uma das suas maior vantagem face ao Javascript é a deteção de erros durante a sua implementação, permitindo construir sistemas mais seguros e de fácil manutenção melhorando consequentemente a produtividade.

## 2.12. Protocolo ONVIF

O protocolo Open Network Vídeo Interface (ONVIF) [29] estabelece as normas para a comunicação com câmaras de vídeo de modo a oferecer soluções e se tornar um *standard*, dada a divergência dos fabricantes na utilização de diferentes protocolos formatos e especificações. Estas especificações definem um protocolo comum para troca de informações entre dispositivos de captura de vídeo baseados em IP em que disponibilizam operações de diversos serviços, como transmissão de áudio/vídeo em tempo real, análise de vídeo, controlo do áudio e reprodução de gravações.



### 3. Análise do domínio do problema UX-IoT

---

De forma a conseguir traçar o plano de ação utilizado no desenvolvido do sistema, foram definidos os requisitos funcionais e não funcionais, apresentados na secção 3.1, os casos de uso do sistema, discutidos na secção 3.2, o modelo de dados concebido, o qual é apresentado na secção 3.3 e também o contexto de aplicações do sistema, apresentado na secção 3.4. Todos estes requisitos são apresentados através de diagramas, com o objetivo de oferecer ao utilizador uma perceção do sistema.

#### 3.1. Requisitos Funcionais e Não Funcionais

Devido à necessidade de separar as funcionalidades do sistema das suas características, considerámos as divisões dos requisitos em funcionais (RF) [30] e não funcionais (RNF) [31].

Os requisitos funcionais são descrições de funcionalidades do sistema onde não são especificados detalhes de âmbito tecnológico. Já os requisitos não funcionais, como o nome sugere, não estão diretamente relacionados com funcionalidades implementadas pelo sistema para os seus utilizadores, mas sim com as características expressas na forma de uma qualidade ou restrição, que ajuda a definir requisitos de desempenho, segurança e confiabilidade.

Na Figura 3, é possível observar as funcionalidades pretendidas para o sistema, sendo elas a possibilidade de apresentar uma interface ao utilizador, RF-1.1, o suporte para o registo de uma `PESSOA` no sistema, RF-1.2, e a sua posterior autenticação, RF-1.3.

A funcionalidade referenciada como RF-1.4, propõe-se a estabelecer um quadro sinótico para visualizar um conjunto de elementos gráficos que representam dispositivos IoT (*widgets*). Pretende-se ainda disponibilizar um menu de contexto que permita aceder às características dos dispositivos IoT, RF-1.5.

Por fim é ainda pretendido oferecer o suporte à criação ou exclusão de um novo elemento no quadro sinótico, tal como a gestão dos dispositivos IoT, RF-1.6.

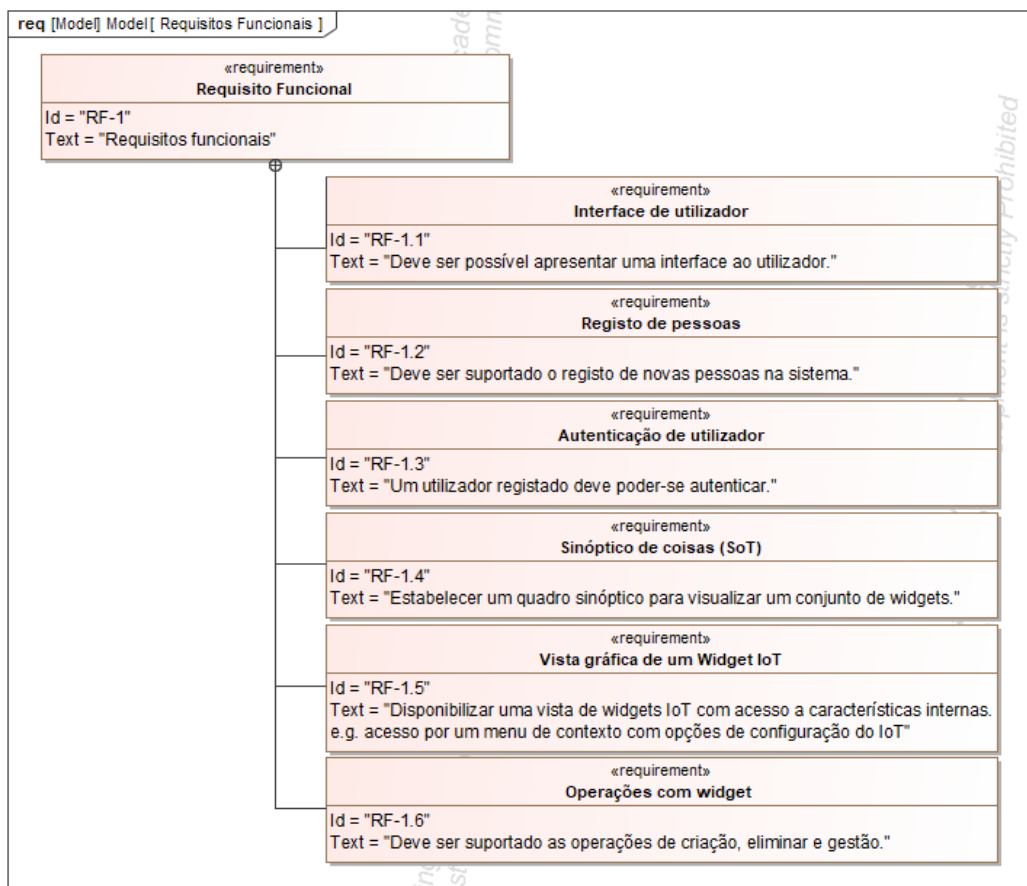


Figura 3 – Requisitos Funcionais

Na Figura 4 encontram-se definidos os requisitos não funcionais do sistema, que passam por garantir não só a persistência dos dados, RNF-1.1, como também a possibilidade de utilização do sistema em qualquer dispositivo, RNF-1.2.

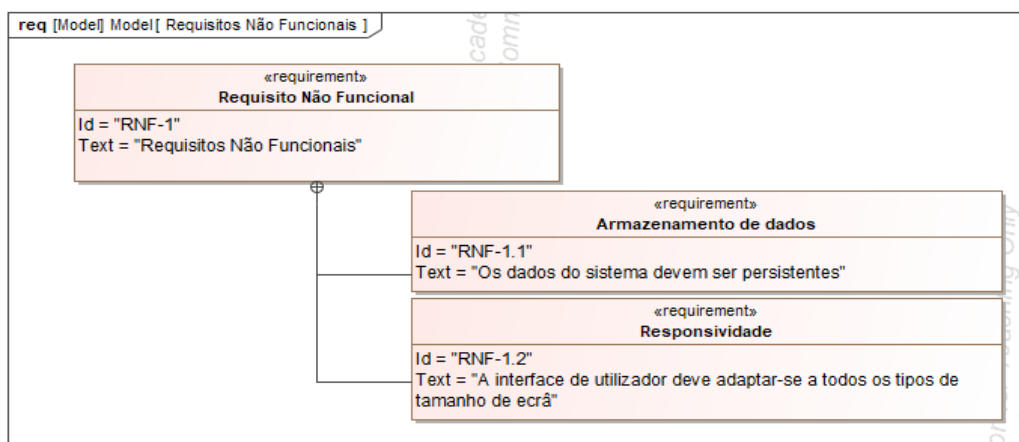


Figura 4 – Requisitos não Funcionais



### 3.2. Casos de Uso

Os diagramas de casos de uso [32] são utilizados para visualizar a forma como as pessoas interagem com o sistema e ajuda na representação de uma visão de alto nível das relações, de forma a facilitar o entendimento sem entrar em detalhes técnicos.

Conforme representado na Figura 5, foi adotada uma visão na qual uma *Pessoa* é o alvo principal do sistema podendo ser representada em dois papéis distintos: *Utilizador* ou *Administrador*. É possível também observar na Figura 5 que o papel de uma *Pessoa* condiciona a diferentes usos das funcionalidades do sistema.

As distribuições dos casos de uso referidos funcionam de uma forma hierárquica estando o *Administrador* no topo dessa hierarquia, uma vez que este tem acesso a todos os elementos, destacando a possibilidade de registar uma *Pessoa* e a manipulação dos *widgets*. Uma *Pessoa* não possui nenhum acesso ao sistema enquanto não for registada, ação esta que lhe concede um papel de *Utilizador* com o qual poderá autenticar-se de forma a ter acesso à manipulação dos seus sinóticos.

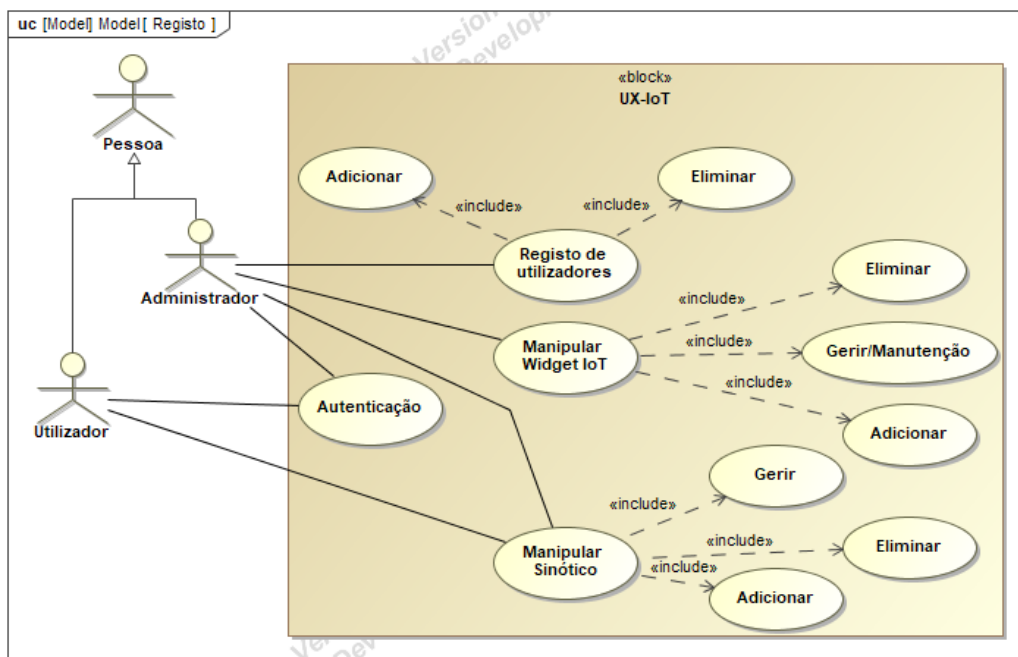


Figura 5 – Diagrama casos de uso

### 3.3. Modelo de Dados

No domínio do problema UX-IoT estão presentes alguns elementos, descritos no modelo conceptual representado na Figura 6, que foi modelado a partir do conceito *Extended Entity Relationship* (EER) [33] simplificado, tendo como base o livro de referência *Fundamentals of Database Systems* de Ramez Elmasri [34].

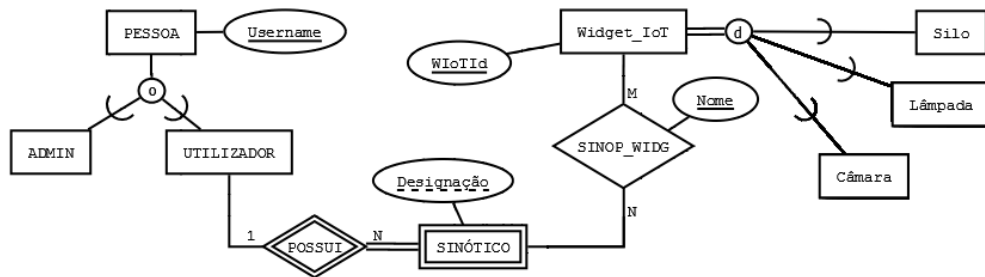


Figura 6 – Modelo EER simplificado

Como é possível observar na Figura 6, existem duas entidades-tipo principais e independentes de todas as outras, nomeadamente a PESSOA e o Widget\_IoT. A entidade-tipo PESSOA possui uma associação classe/subclasse do tipo *overlapping*, o que significa que existe a possibilidade de pertencer a ambas as subclasses (ADMIN e UTILIZADOR). Já o Widget\_IoT, apesar de também possuir uma associação classe/subclasse, esta é do tipo *disjoint*, ou seja, cada Widget\_IoT pode apenas pertencer a uma das subclasses de entre Silo, Lâmpada ou Câmara.

A subclasse UTILIZADOR relaciona-se com a entidade SINÓTICO, de uma forma forte, isto é, um sinótico não pode existir de forma independente ao utilizador e possui uma cardinalidade de 1 : N, o que significa que cada utilizador possui diversos sinóticos e vários sinóticos podem pertencer a um UTILIZADOR.

As entidades Widget\_IoT e SINÓTICO relacionam-se de forma associativa, pois estes são integrados em conjunto de forma a criar a interface gráfica mostrada ao utilizador. Apesar de omitida, a cardinalidade desta relação é de N : M, ou seja, vários *widgets* podem pertencer a vários sinóticos e vice-versa, uma vez que as suas utilizações não se encontram limitadas.

### 3.3.1. Entidade PESSOA

Na Figura 7 encontra-se a representação da entidade PESSOA, esta que é constituída por seis atributos que por sua vez representam toda a informação que o caracteriza.

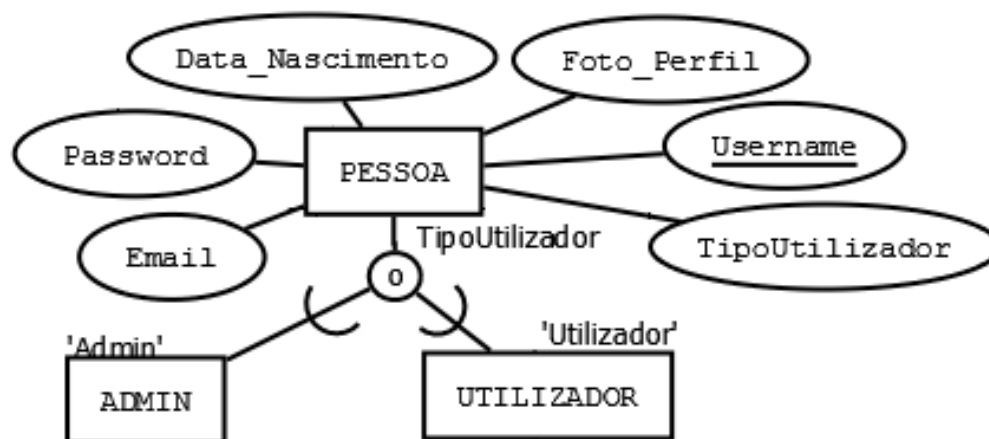


Figura 7 – Entidade PESSOA do modelo de dados

No conjunto dos atributos existem alguns que são de existência obrigatória, nomeadamente o Username, que representa o identificador da PESSOA, a Password que será utilizada para realizar a autenticação e o TipoUtilizador que irá condicionar as ações que a PESSOA irá conseguir realizar dentro do sistema. Para além disso, é possível adicionar o Email de contacto, uma Foto\_Perfil e a Data\_Nascimento.

O atributo TipoUtilizador pode ser de diversos tipos, mas apenas assumir um deles uma vez que como referido no início deste capítulo, possui uma associação do tipo *overlapping*, ou seja, a PESSOA pode ser um UTILIZADOR, um ADMIN, ou ambos ao mesmo tempo.

### 3.3.2. Entidade Widget\_IoT

A entidade Widget\_IoT encontra-se representada pela Figura 8, onde é possível observar que esta entidade é composta por quatro atributos, contudo, ao contrário do que acontecia com a entidade PESSOA, todos estes atributos são de existência obrigatória.

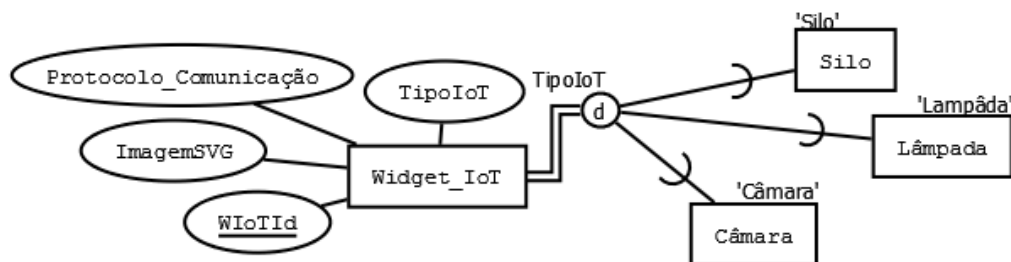


Figura 8 – Entidade *Widget\_IoT* do modelo de dados

O *WIoTId* é o identificador desta entidade e é gerado automaticamente pela base de dados no ato da sua criação, a *ImagemSVG* é responsável por representar visualmente o *TipoIoT*. Isto é possível uma vez que devido à associação ser *disjoint* esta entidade apenas pode ser de um dos tipos representados: *Silo*, *Lâmpada*, *Câmara*. Para além dos atributos referidos, é também necessário existir um *Protocolo\_Comunicação* que indica como o *Widget\_IoT* irá comunicar com o sistema ciberfísico.

### 3.3.3. Entidade SINÓTICO

A entidade *SINÓTICO*, possui uma constituição mais simples possuindo apenas 2 atributos tal como ilustra a Figura 9.

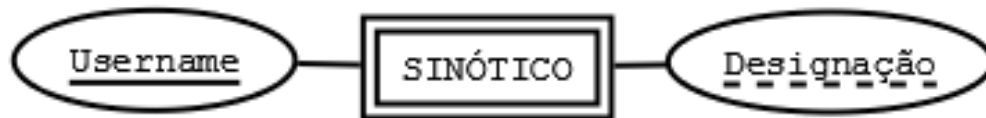


Figura 9 – Entidade *SINÓTICO* do modelo de dados

Ao contrário das entidades anteriores, esta é identificada pelos seus dois atributos, sendo importante destacar que um deles, o *Username*, que é herdado da entidade tipo *PESSOA*. Para além deste atributo, como é visível na entidade *SINÓTICO*, esta possui também uma *Designação* própria, que em conjunto com o *Username* formam a chave desta entidade.

### 3.3.4. Relação SINOP\_WIDG

A entidade `SINOP_WIDG`, como é possível observar na Figura 10, possui uma característica diferente de todas as outras que se denomina de entidade associativa, isto é, esta entidade associa outras duas entidades, neste caso: `SINÓTICO` e `Widget_IoT`.

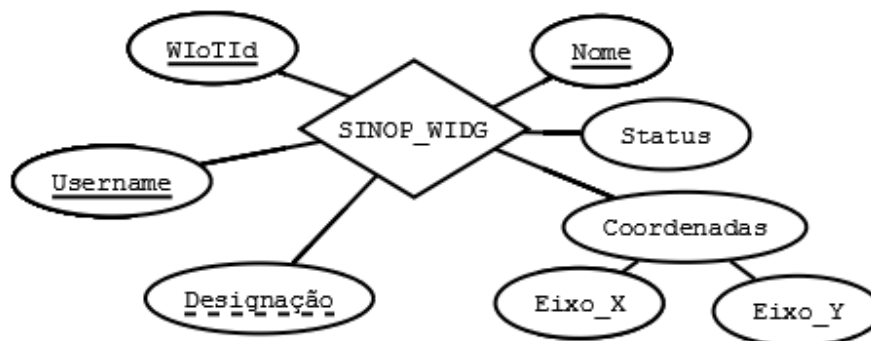


Figura 10 – Relação `SINOP_WIDG` do modelo de dados

Por consequência desta característica, tal como é visível na figura, esta entidade possui os identificadores de ambas as entidades referidas. Como esta entidade destina-se a ser a representação do *widget* num sinótico, possui os atributos: `Nome`, que o identifica, permitindo ter diversos *widgets* do mesmo tipo, o `Status` que corresponde a um *feedback* do funcionamento do sistema ciberfísico e as `Coordenadas` em que o *widget* é representado no sinótico.

O modelo de dados completo, onde são apresentadas todas as entidades que o constituem e em que constam não só todas as relações entre as entidades como também todos os seus atributos, encontra-se no Apêndice 1.

## 3.4. Contextos de Aplicação

Para o desenvolvimento do sistema UX-IoT, foi adotada uma abordagem de divisão de blocos de implementação, representado na Figura 11 em que cada um destes elementos possui funções específicas.

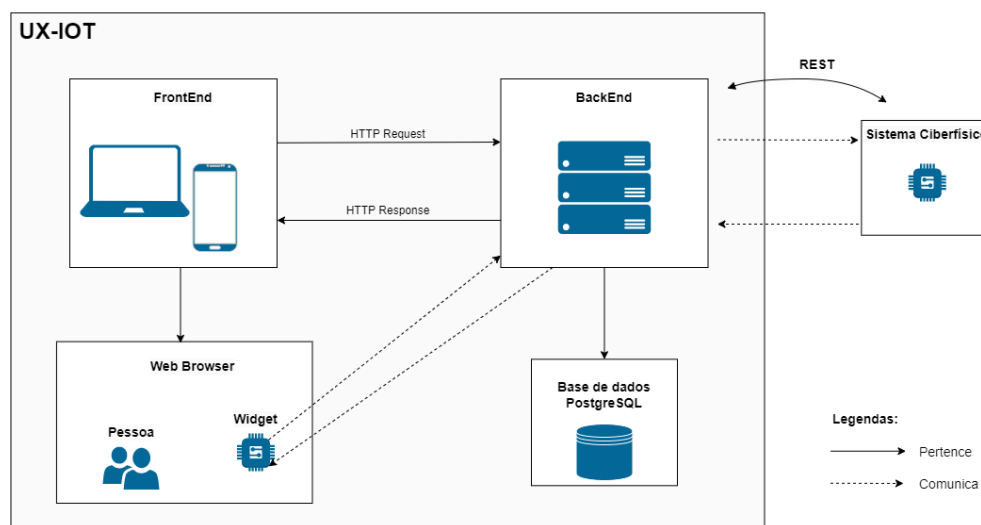


Figura 11 – Modelo que representa as partes do sistema

O módulo *front-end* corresponde ao código de desenvolvimento dos *Componentes Web*, fazendo o consumo de uma API e gerando o conteúdo de interação gráfica no *Web browser*, onde se situa o utilizador. Possibilita a interação com os *widgets*, a visualização e o manuseamento dos componentes gráficos, utilizando tecnologias como HTML, CSS e Javascript.

O módulo *back-end*, por sua vez, é responsável por ligar os dados provenientes do *browser* e os dados armazenados na base de dados PostgreSQL [35], que por sua vez utiliza uma linguagem de programação SQL [36]. Possui também as validações dos dados, dando garantias de que um utilizador não possa aceder ou manipular algo do qual não deveria ter acesso.

Os elementos anteriores são da responsabilidade do sistema, contudo, o sistema ciberfísico é um elemento que, apesar da responsabilidade ser de terceiros, disponibiliza serviços (REST API) para a comunicação do servidor com os *widgets* e vice-versa. A informação pode ser requisitada pelo servidor ou desencadeado pelo próprio *widget* através de eventos.

## 4. Desenvolvimento do Sistema UX-IoT

Com base na análise dos diagramas e modelos de dados abordados no capítulo anterior, foram aplicados os passos necessários para concretizar o sistema. Neste capítulo são apresentadas as especificações e implementações do Componente Web e da interface de utilizador.

De forma a dar suporte aos casos de uso a ser implementado pelo sistema, foi desenvolvido um painel de controlo que dá acesso a todos os contextos. O painel de controlo, como ilustrado na Figura 12 oferece uma vista com diversas opções que reagem ao *click* do botão, navegando para a opção seleccionada, a partir de uma rota específica definida para cada. É de se notar que a vista apresentada pertence a um ADMINISTRADOR, pois um UTILIZADOR não contém as opções de registo de *widgets* e de PESSOAS.

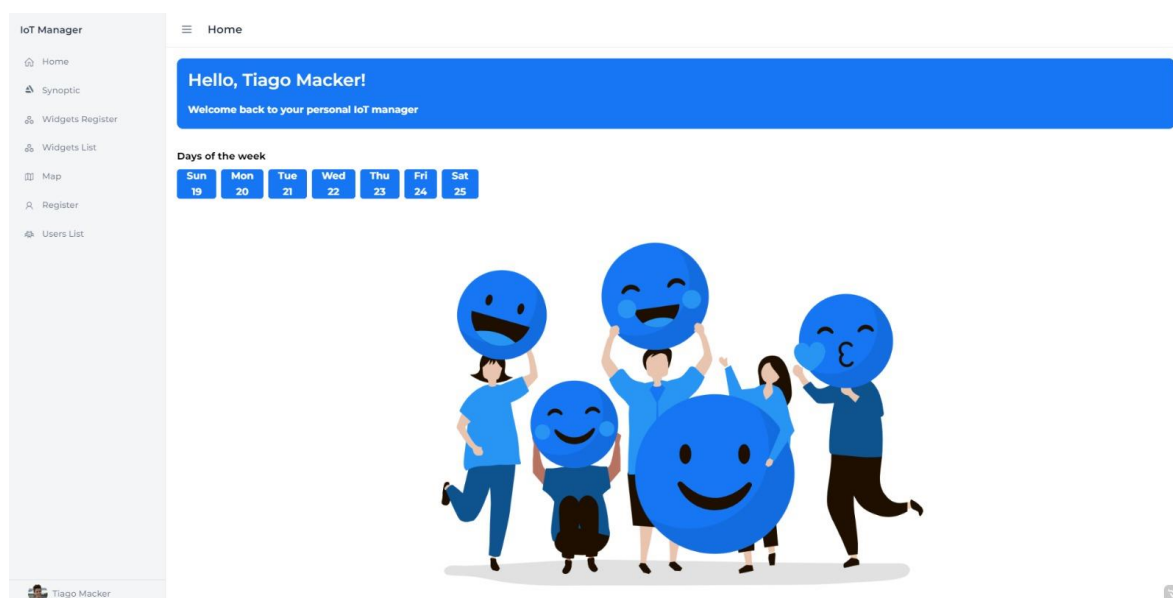


Figura 12 – Painel de controlo do sistema UX-IoT

Os requisitos e funcionalidades de cada um dos casos de uso são descritos nas secções que se seguem, de forma a instruir uma pessoa de como deverá utilizar o sistema, com o objetivo de evitar quaisquer problemas e garantir uma melhor experiência na utilização do mesmo.

## 4.1. Autenticação

A autenticação apresenta uma interface simples, mas que, no entanto, as funcionalidades por de trás desta podem ser complexas. Como é possível observar na Figura 13 existem dois campos onde o utilizador pode escrever, nomeadamente o `Username` e a `Password`, ou seja, as credenciais do utilizador, e existe também o botão que desencadeia o processo de autenticação, contudo este apenas funciona caso ambos os campos estejam preenchidos.

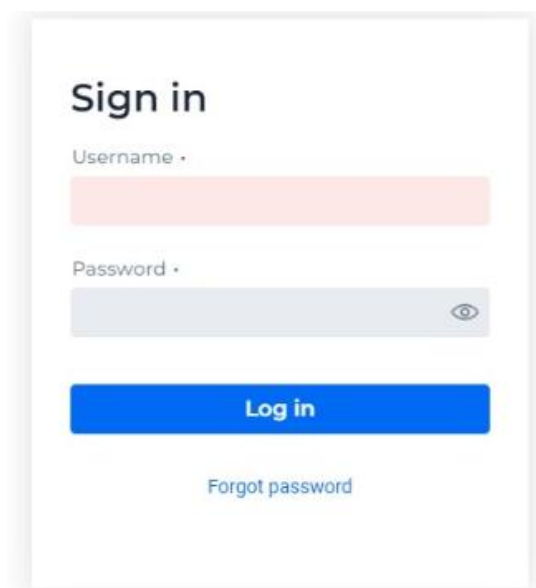
The image shows a 'Sign in' form. At the top, the text 'Sign in' is displayed in a bold, dark font. Below it, there are two input fields. The first is labeled 'Username' and has a light red border. The second is labeled 'Password' and has a light blue border with a small eye icon on the right side. Below the password field is a blue button with the text 'Log in' in white. At the bottom of the form, there is a link that says 'Forgot password' in a smaller, blue font.

Figura 13 – Formulário de autenticação

O processo de autenticação encontra-se num formato de fluxograma ilustrado na Figura 14, onde inicialmente irá ser realizado um pedido à base de dados de forma a tentar encontrar o utilizador correspondente, caso este não seja encontrado irá aparecer visualmente uma mensagem de que as credenciais estão incorretas. Caso contrário, irá prosseguir com o processo de autenticação, onde a password inserida será encriptada por um método de criptografia denominado Bcrypt [37] e comparada com a que se encontrava na base de dados. No caso da comparação ter sucesso, é adicionado ao `Username` o papel que este desempenha no sistema e é garantido o acesso, mantendo a autenticação a partir da criação de um contexto de segurança que funciona como um *HTTP Cookie* [38], guardando os detalhes do utilizador, nomeadamente as suas credenciais, no entanto este contexto tem a diferença de ser interno ao sistema.



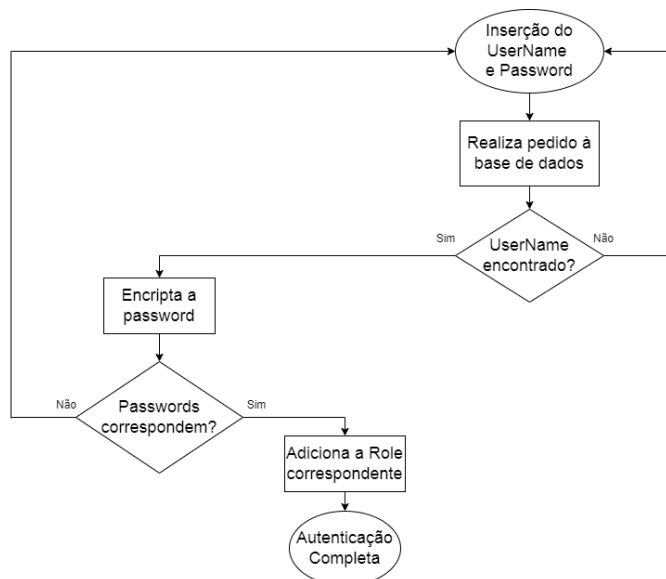


Figura 14 – Fluxograma do processo de autenticação

## 4.2. Registo de Utilizadores

O registo das pessoas como utilizadores é realizado apenas por alguém que já possua um registo e cujo papel no sistema seja de ADMINISTRADOR, neste processo, uma vez que o sistema foi projetado para um contexto de empresa, é interessante que sejam guardadas algumas informações sobre a entidade PESSOA, como por exemplo a data de nascimento, o email de contacto e uma foto de perfil. Para além das informações referidas é necessário também associar um Username que irá identificar a pessoa no sistema e a sua respetiva password para autenticação. Os campos de inserção destes dados são acompanhados de quatro botões, tal como ilustra a Figura 15, cada um com a sua própria funcionalidade sendo elas respetivamente a de criar um utilizador no sistema, *Save*, atualizar as informações do utilizador pretendido, *Update*, eliminar o utilizador pretendido da base de dados do sistema, *Delete*, ou limpar os dados inseridos nos campos, *Cancel*.

**Personal information**



Username • John Dane	Password • Must have at least 6 characters
Birthday • DD/MM/YYYY	Email address • username@example.com
Role • Admin or User	
Image 	
<div>Upload File...  Drop file here</div>	
<div>Save Update Delete Cancel</div>	

Figura 15 – Formulário de registo de utilizadores

Todos estes botões à exceção do último desencadeiam pedidos à base de dados, sendo o processo de criação um pouco mais complexo, pois tal como demonstra a Figura 16, o pedido é antecedido tanto da criação de um ficheiro com a foto de perfil adicionada, caso tenha sido submetida alguma, como da encriptação da password a partir do mecanismo BCrypt.

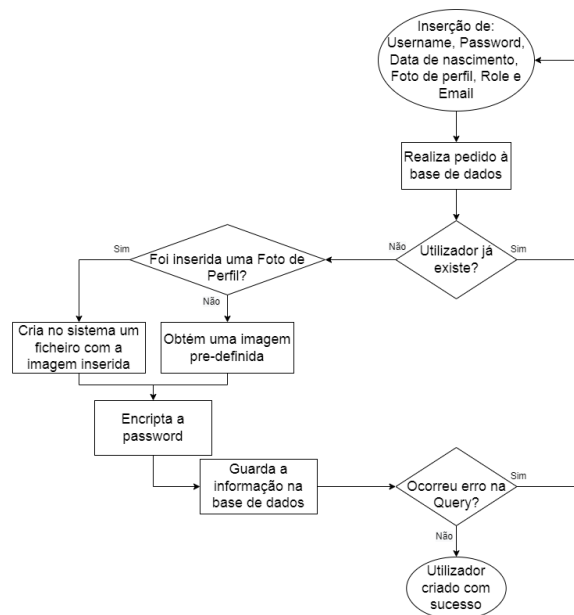


Figura 16 – Fluxograma do processo de registo de pessoas

### 4.3. Registo de Widget IoT

Em semelhança com o registo de utilizadores, o registo de `Widgets IoT`, apenas pode ser realizado por um utilizador com permissões de `ADMINISTRADOR`. A vista criada para a realização desta tarefa, como é possível observar na Figura 17, também se assemelha à utilizada na secção anterior, existindo os mesmos quatro botões com as mesmas funções, no entanto, esta tarefa necessita da inserção de um número de campos inferior, uma vez que apesar de existir um campo para o `Widget ID`, este só necessita de ser preenchido para as ações de `Delete` e `Update`.

Na criação do `Widget`, de forma a que dois elementos não possuam o mesmo identificador a geração deste é feita automaticamente através da base de dados por uma função denominada `uuid_generate_v4` que provem da extensão `uuid-oss`. Além do campo referido anteriormente, existem outros três que são de inserção obrigatória, nomeadamente o tipo de `Widget`, que identifica qual o sistema ciberfísico que este irá representar graficamente (ex: câmara), o `Protocol`, destinado a referenciar o protocolo que irá ser utilizado para comunicar entre os sistemas e por fim, a imagem que é utilizada para representar visualmente o elemento sendo esta do tipo `SVG`.

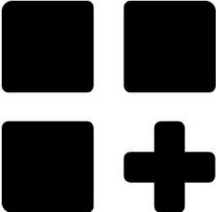
### Widget information


Widget ID: Generated automatically by the database i

Type: Camera, Recorder, Lamp

Protocol: Onvif, Rest

Image



Upload File...  Drop file here

Save Update Delete Cancel

Figura 17 – Formulário de registo de widgets IoT

## 4.4. Widgets-IoT

Na criação de um *Widget-IoT* foi adotado a utilização da linguagem Typescript com a biblioteca Lit que permite criar os componentes que já se enquadram nos padrões de um Componente Web. Para uma melhor estruturação tanto logica como visual foi feito uma subdivisão de tarefas em 2 web componentes. O primeiro trata do acesso ao simulador da camara de forma a obter os dados referentes às características desta tais como: a resolução, o zoom, a sensibilidade e o ângulo através da interface REST, passando estes valores como propriedade ao segundo. Este por sua vez consiste tanto na parte estética através da utilização de CSS, como na produção do corpo do elemento através de HTML que por sua vez reage aos vários eventos associados a este tal como a alteração de características da camara que por sua vez também utiliza uma interface REST para atualizar os dados.

A Figura 18 ilustra a representação visual dada pelo segundo Componente Web com as informações passadas pelo seu pai como propriedade, neste caso o nome, a imagem e o status que conforme o seu valor altera visualmente a cor do *widget* transmitindo a informação de que está a operacionalizar normalmente ou que possui alguma avaria ou até mesmo que este se encontra desligado.



Figura 18 – Componente Web (Câmara)

Um dos eventos criados foi o *click* onde o utilizador ao pressionar com o botão esquerdo em cima do componente mencionado anteriormente, gera a apresentação de uma janela, ilustrada na Figura 19, onde consta a *stream* de vídeo referente à camara, com as configurações das suas características, que por sua vez podem ser alteradas pelo utilizador através da seleção das opções disponíveis de configuração da camara, sendo que o utilizador, no final, pode submeter estas alterações para o servidor REST.

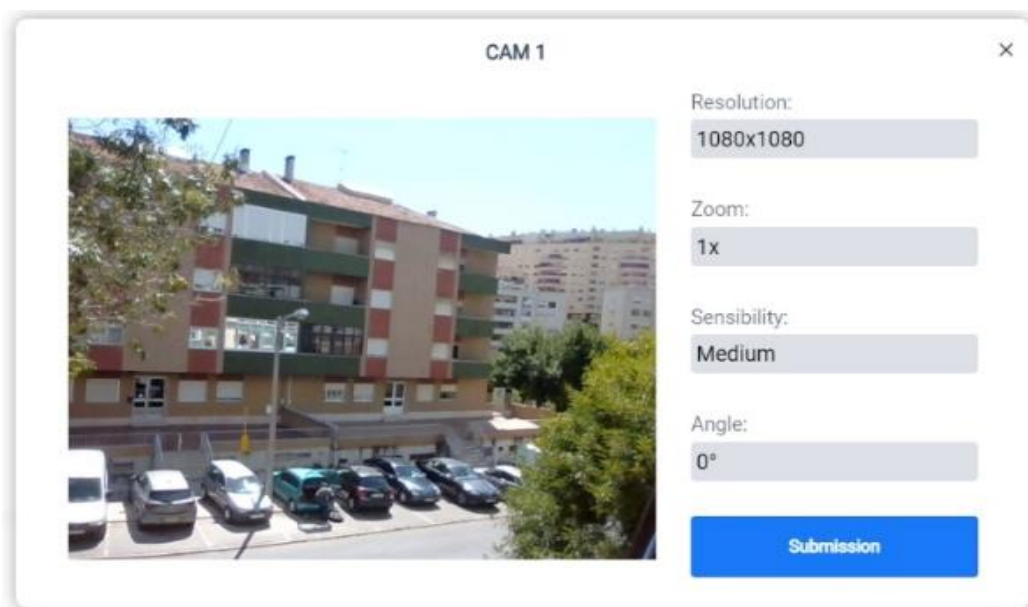


Figura 19 – Menu de contexto do Componente Web

Algumas características do primeiro componente a se realçar é que este de forma a simular a obtenção dos dados em tempo real executa o método `GET` de segundo a segundo passando como propriedades ao segundo componente os novos dados obtidos.

## 4.5. Simulador de Câmara

De modo a testar os pedidos em tempo real por parte dos componentes referidos na secção anterior, foi desenvolvido um servidor REST de onde é possível obter ou alterar os valores de uma câmara “fictícia”. No seguimento deste objetivo, foi utilizado o ambiente de execução NodeJS [39] em conjunto com o *middleware* Express [40]. Esta escolha foi tomada, dado a utilização do mesmo em unidades curriculares passadas que nos ofereceram alguma experiência com o mesmo, contudo, poderia ter sido utilizado outro ambiente de execução, como é o exemplo do Deno [41] que se tem vindo a tornar uma alternativa viável ao NodeJS.

Este servidor é composto apenas por um URL que tem como parâmetro o identificador da câmara, no entanto este possui duas rotas distintas, uma que realiza um pedido `GET`, ou seja, que tem como função obter informação, neste caso pertencentes à câmara identificada no URL. A segunda rota realiza um pedido `PUT`, utilizado para atualizar os valores dos recursos da câmara, sendo estes recebidos no corpo do pedido, no entanto, como não é obrigatório o utilizador alterar todos os campos para realizar este pedido, são previamente verificados quais necessitam de atualização, de forma a deixar os restantes intactos.

No desenvolvimento destas rotas, é tido em conta os atributos referentes a cada tipo de pedido, nomeadamente a segurança no pedido `GET` e a idempotência no pedido `PUT`, garantindo assim que ambos estão em ótimo funcionamento e de acordo com as regras definidas no HTTP.

## 4.6. Operações sobre um Sinótico

De forma a visualizar o Componente Web criado, existe uma opção de contexto que permite ao utilizador navegar para uma página onde possui um quadro sinótico, ilustrada na Figura 20. No momento em que é gerada esta página, o sistema acede à base de dados com o intuito de verificar se o utilizador já possui algum elemento a ser ilustrado, dando seguimento à criação de um Componente Web para cada elemento encontrado e a sua respetiva localização no sinótico. Juntamente com o Componente Web é colocado um cabeçalho com o nome que o utilizador atribuiu anteriormente ao sistema ciberfísico, sendo este cabeçalho responsável por realizar a movimentação do elemento dentro do sinótico,

processo este que é realizado numa perspetiva *point & click*, ou seja, inicialmente o utilizador pressiona o cabeçalho e de seguida pressiona no local em que pretende colocar o elemento, realizando assim a deslocação.

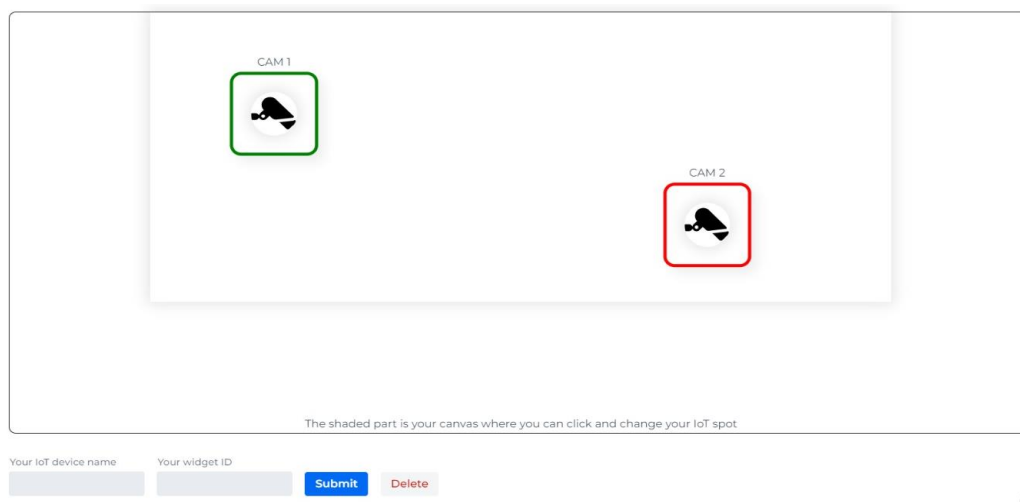


Figura 20 – Vista do Quadro Sinótico

Esta interface permite ainda duas funcionalidades, a adição de um novo elemento ao sinótico, sendo necessário apenas introduzir o nome que se pretende atribuir e o identificador do *widget* a adicionar. A outra funcionalidade corresponde a eliminação do elemento do sinótico sendo necessário também tanto o nome com o identificador do *widget*. Ao pressionar o botão pretendido, estas informações serão processadas na base de dados sendo que na adição é gerado um identificador necessário para realizar os pedidos ao servidor REST desenvolvido para simular as câmaras.





## 5. Conclusões

---

Este relatório, apresenta não só o trabalho desenvolvido no projeto, como toda a análise necessária para tal, tendo sempre como objetivo oferecer uma interface que permita aos utilizadores observar os seus sistemas ciberfísicos representados graficamente através dos *widgets*. Com base nos conhecimentos teóricos foi possível a conceção com sucesso dos diagramas presentes no capítulo 3, mais concretamente, o levantamento de requisitos e a criação do modelo conceptual.

No planeamento do projeto previu-se a existência de *Pessoas* que podem ser registadas no sistema a partir de um *Administrador*, podendo criar vários sinóticos, nos quais serão inseridos os *widgets* pretendidos, de forma a oferecer o manuseamento e operacionalidade aos seus sistemas ciberfísicos. Um dos intuítos deste projeto foi também torná-lo abrangente, ambicionando a possibilidade dos *widgets* serem de diversos tipos tais como, câmaras, silos, lâmpadas, bicicletas elétricas, etc.

Contudo, durante o desenvolvimento, foi constatado que pelo tempo disponível não seria possível nem alcançar a possibilidade de oferecer mais do que um sinótico, nem a diversidade nos tipos dos *widgets*, sendo então oferecido o acesso a um único sinótico, que apesar de ser uma limitação não inviabiliza a construção de mais no futuro, e também de um único *widget*, nomeadamente uma câmara.

A implementação do serviço Web REST permitiu deslocar uma parte do processamento dos dados reduzindo a carga computacional do sistema, cumprindo com os objetivos especificados para o serviço nomeadamente: expor as informações através de uma API e possibilitar a simulação de obtenção de dados em tempo real.

No requisito da interface gráfica do sistema Web foi possível proporcionar uma fácil e agradável interação ao utilizador, além de disponibilizar funcionalidades conforme o tipo de papel desempenhado em um contexto empresarial, tais como: a criação de utilizadores, a criação de *widgets*, o acesso aos respetivos *streams* multimédia e um *design* responsivo garantindo a sua compatibilidade numa variedade de dispositivos, incluindo móveis.

## 5.1. Pesquisa e Trabalho Futuro

Ao longo deste projeto foi apenas concebido um Componente Web que representa uma câmara, contudo, é possível a associação deste componente a qualquer sistema ciberfísico. Assim sendo, podemos considerar que o sistema desenvolvido é apenas uma base, para o que pode vir a ser algo maior, uma vez que foi discutido diversas formas de utilizar as ideias utilizadas no decorrer do desenvolvimento do projeto, como no caso de uma *start-up* que seria responsável pela criação de novos modelos de Componentes Web, como também da manutenção do site.

Aliando a possibilidade da criação de uma *start-up* com um modelo de negócios, uma das possíveis abordagens seria em alterar a forma de registo, permitindo que qualquer pessoa consiga registar-se no sistema, no entanto limitar o número de *widgets* a que estes podem apresentar no sinótico de acordo com uma subscrição paga, como acontece em diversos sistemas atualmente. Esta ideia prevê a existência de um utilizador com uma conta do tipo *Free* onde este apenas poderia colocar no máximo três *widgets*, mas se adquirisse a subscrição *Premium* por um certo montante seria lhe garantido o acesso à utilização de um número mais elevado de *widgets*.

A autenticação neste momento passa por aceder à base de dados onde está armazenado o nome de utilizador e a sua password encriptada pelo mecanismo BCrypt, no entanto, caso um atacante consiga aceder à base de dados, este mecanismo está desprotegido a ataques de dicionário [42]. Assim sendo, seria interessante a implementação de mecanismo que oferecesse uma maior proteção às credenciais dos utilizadores, para que estes consigam operar no sistema com a devida segurança, possivelmente aplicando conceitos de cifra como a utilização de *seeds* de forma a garantir uma maior aleatoriedade na password armazenada.

Outra das implementações possíveis para o sistema seria o conceito de grupos de trabalho, o que permitiria que vários utilizadores conseguissem manipular o mesmo sinótico ao mesmo tempo, no entanto, iria acrescentar à implementação do sistema o cuidado com a concorrência dos dados, garantindo o sincronismo entre todos os utilizadores presentes nesse grupo.

Todas estas ideias têm uma possível implementação, no entanto seria necessário um trabalho futuro e a respetiva pesquisa com intuito de as conseguir realizar, pois estes pontos

não só necessitam de um cuidado especial como acrescentam uma maior dificuldade de implementação ao sistema.



## 6. Bibliografia

---

- [1] IoT. [Online]. Available: <https://www.oracle.com/pt/internet-of-things/what-is-iot/>.
- [2] "Cyber-Physical Systems and Internet of Things," 2019. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1900-202.pdf>.
- [3] M. N. O. W. Y. C. S. & M. S. M. Sadiku, "Cyber-Physical Systems: A Literature Review," 2017. [Online]. Available: <https://doi.org/10.19044/esj.2017.v13n36p52>.
- [4] SysML. [Online]. Available: <https://sysml.org/>.
- [5] "Framework," [Online]. Available: <https://pt.wikipedia.org/wiki/Framework>.
- [6] "ISO 19514:2017," [Online]. Available: <https://www.iso.org/standard/65231.html>.
- [7] "UML," [Online]. Available: <https://pt.wikipedia.org/wiki/UML>.
- [8] "Web Components," [Online]. Available: <https://www.webcomponents.org/introduction>.
- [9] B. Farrell, Web Components in Action.
- [10] "DOM," [Online]. Available: [https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp).
- [11] "Javascript," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [12] "CSS," [Online]. Available: [https://pt.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://pt.wikipedia.org/wiki/Cascading_Style_Sheets).
- [13] "Vaadin," [Online]. Available: <https://vaadin.com/docs/v8/framework/introduction/intro-overview>.
- [14] "GWT," [Online]. Available: <https://vaadin.com/docs/v8/framework/clientsidewidgets/clientsidewidgets-gwt>.
- [15] "GUI," [Online]. Available: [https://pt.wikipedia.org/wiki/Interface\\_gr%C3%A1fica\\_do\\_utilizador](https://pt.wikipedia.org/wiki/Interface_gr%C3%A1fica_do_utilizador).
- [16] "HTML," [Online]. Available: <https://pt.wikipedia.org/wiki/HTML>.

- [17] "ReactJS," [Online]. Available: <https://reactjs.org/>.
- [18] "AngularJS," [Online]. Available: <https://angular.io/>.
- [19] "VueJS," [Online]. Available: <https://vuejs.org/>.
- [20] "JSON," [Online]. Available: <https://www.json.org/json-en.html>.
- [21] "API," [Online]. Available: <https://www.techopedia.com/definition/24407/application-programming-interface-api>.
- [22] "HTTP," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP>.
- [23] "REST," [Online]. Available: <https://pt.wikipedia.org/wiki/REST>.
- [24] "XML," [Online]. Available: <https://pt.wikipedia.org/wiki/XML>.
- [25] "LIT," [Online]. Available: <https://lit.dev/>.
- [26] "SVG," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/SVG?retiredLocale=pt-PT>.
- [27] "CANVAS," [Online]. Available: <https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta-p/45>.
- [28] "RGBA," [Online]. Available: [https://www.w3schools.com/css/css\\_colors\\_rgb.asp](https://www.w3schools.com/css/css_colors_rgb.asp).
- [29] "Onvif," [Online]. Available: <https://aprendacftv.com/entenda-o-que-e-onvif/>.
- [30] "Req\_Funcional," [Online]. Available: [https://pt.wikipedia.org/wiki/Requisito\\_funcional](https://pt.wikipedia.org/wiki/Requisito_funcional).
- [31] "Req\_NãoFuncional," [Online]. Available: [https://pt.wikipedia.org/wiki/Requisito\\_n%C3%A3o\\_funcional](https://pt.wikipedia.org/wiki/Requisito_n%C3%A3o_funcional).
- [32] "Diagrama\_CasosDeUso," [Online]. Available: [https://pt.wikipedia.org/wiki/Diagrama\\_de\\_caso\\_de\\_uso](https://pt.wikipedia.org/wiki/Diagrama_de_caso_de_uso).
- [33] "EntityRelationship," [Online]. Available:

<https://www.ibm.com/docs/en/imdm/12.0?topic=concepts-key-entity-attribute-entity-type>.

[34] R. Elmasri, Fundamentals of Database Systems.

[35] "PostgreSQL," [Online]. Available: <https://www.postgresql.org/about/>.

[36] "SQL," [Online]. Available: <https://www.thebalancecareers.com/what-is-sql-and-uses-2071909>.

[37] "Bcrypt," [Online]. Available: <https://pt.wikipedia.org/wiki/Bcrypt>.

[38] "HTTP Cookie," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>.

[39] "NodeJS," [Online]. Available: <https://nodejs.org/en/>.

[40] "Express," [Online]. Available: <https://expressjs.com/>.

[41] "Deno," [Online]. Available: <https://deno.land/>.

[42] "Ataque Dicionario," [Online]. Available: [https://pt.wikipedia.org/wiki/Ataque\\_de\\_dicion%C3%A1rio](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio).

[43] "CORS," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>.





## 7. Apêndices

### 7.1. Apêndice 1

