

Universidade da Beira Interior

Departamento de Informática



Departamento de
Informática

Internet das Coisas - Sistema de Vigilância Domiciliar para Idosos

Elaborado por:

M12246 - Alexandre Monteiro

M12306 - Ângelo Morgado

M12839 - Miguel Manteigueiro

Professor Doutor Bruno Silva

7 de junho de 2023

Acrónimos

API	application programming interface
DOS	<i>Denial-of-Service</i>
DDOS	<i>Distrubuted Denial-of-Service</i>
HTTP	hyper text transfer protocol
HTTPS	hyper text transfer protocol secure
IP	internet protocol
JSON	JavaScript object notation
REST	Representational state transfer application programming interface
IOT	internet of things
MCI	mild cognitive impairment
MQTT	message queuing telemetry transport
SSL	secure sockets layer
SWOT	Strengths, Weaknesses, Opportunities, Threats
URL	uniform resource locator

Conteúdo

Conteúdo	ii
Lista de Figuras	v
1 Introdução	1
1.1 Enquadramento	1
1.2 Motivação	1
1.3 Objetivo	1
1.4 Organização do Documento	2
2 Estado da Arte	3
2.1 Soluções Existentes	3
2.1.1 Solução 1	3
2.1.2 Solução 2	3
2.1.3 Solução 3	4
2.1.4 Solução 4	4
2.1.5 Solução 5	5
2.1.6 Solução 6	5
2.2 A Nossa Solução	6
3 Engenharia de Software e Arquitetura de Sistema	9
3.1 Introdução	9
3.2 <i>Personas</i> e <i>User Stories</i>	9
3.2.1 <i>Personas</i>	9
3.2.1.1 Persona 1: O Idoso	10
3.2.1.2 Persona 2: A Pessoa Responsável pelo Idoso	10
3.2.1.3 Persona 3: O Administrador do Sistema . .	10
3.2.2 <i>User Stories</i>	11
3.3 Requisitos de Utilizador	12
3.3.1 Requisitos Funcionais	12
3.3.1.1 <i>Dashboard</i>	12

3.3.1.2	Sistema	13
3.3.2	Requisitos Não Funcionais	13
3.4	Tecnologias utilizadas	14
3.4.1	Tecnologias utilizadas no domicílio do idoso	15
3.4.2	Tecnologias utilizadas no servidor	15
3.5	Diagramas de Casos de Uso	16
3.6	Diagrama de Atividade	20
3.7	Arquitetura do Sistema	25
3.8	Base de Dados	26
3.9	Conclusões	27
4	Interoperabilidade	29
4.1	Interoperabilidade de Dispositivos	29
4.2	Interoperabilidade de Plataforma	29
4.3	Interoperabilidade Semântica	30
4.4	Interoperabilidade Sintática	30
4.5	Interoperabilidade de Rede	31
5	Ética e Segurança	33
5.1	Código de Ética	33
5.1.1	Sistema Domiciliar	33
5.1.1.1	Deontologia Profissional (ação)	33
5.1.1.2	Sistema e Sociedade	33
5.1.2	<i>Dashboard</i>	35
5.2	Segurança da informação	35
5.2.1	Ataque de Homem no Meio (<i>Man-in-the-Middle Attack</i>)	36
5.2.2	<i>Malware</i> – Criação de <i>Botnets</i>	36
5.2.3	<i>Denial-of-Service</i> (DOS)	37
5.2.4	<i>Zero Day Exploits</i>	37
6	Metodologia e Implementação	39
6.1	Introdução	39
6.2	Demonstrador do Sistema Físico	39
6.3	Cliente	41
6.3.1	<i>Design</i> Gráfico	41
6.3.2	Autenticação	42
6.3.3	<i>Dashboard</i> Utilizador	44
6.3.4	<i>Dashboard</i> Administrador	45
6.3.5	<i>Logs</i> do sistema	46
6.4	APIs	47

6.4.1	Transmissão de Dados Entre o <i>Gateway</i> e o Servidor – REST API	48
6.4.1.1	<i>Gateway</i>	48
6.4.1.2	Servidor	48
6.5	Chatbot do Telegram	50
6.5.1	Configuração	51
6.5.2	Guia de utilização	51
6.5.3	Autenticação	51
6.5.4	Visualização do Último Evento Alarmante	52
6.5.5	Visualização das Informações das Divisões	53
6.5.6	Notificação de eventos alarmantes	53
6.6	Características Interessantes	55
6.6.1	Criptografia	55
6.6.2	Estatísticas	55
6.6.3	Notificações	56
6.6.4	Implementação de um Domínio	57
6.6.5	Implementação de SSL	57
6.7	Desafios de Implementação	59
6.7.1	Configuração do <i>gateway</i> como um <i>hotspot</i>	59
6.8	Conclusões	59
7	Conclusão	61
7.1	SWOT	61
7.1.1	Forças	61
7.1.2	Fraquezas	61
7.1.3	Oportunidades	62
7.1.4	Ameaças	62
7.2	Conclusão final	62
7.3	Trabalho futuro	63
7.3.1	Adaptação a Redes Constrangidas	64
7.3.2	Aprimoramento da Segurança e Desempenho	64
7.3.3	Integração de Tecnologias Emergentes	64
A	Excertos de Código Relevantes	65
A.1	Código utilizado no ESP32	65
A.2	Código utilizado no Gateway	68
	Bibliografia	71

Lista de Figuras

3.1	Diagrama de caso de uso para as principais funções do Administrador.	16
3.2	Diagrama de caso de uso para as principais funções do Utilizador.	17
3.3	Diagrama de caso de uso sobre a recolha e transmissão de dados.	18
3.4	Diagrama de caso de uso sobre a transmissão de dados do Orange Pi para o servidor.	19
3.5	Diagrama de caso de uso sobre a formatação de dados e posterior registo na base de dados.	20
3.6	Diagrama de atividade relativamente à consulta de informação por parte do utilizador.	21
3.7	Diagrama de atividade relativamente à alteração dos valores considerados limites.	22
3.8	Diagrama de atividade relativamente ao envio de um evento.	23
3.9	Diagrama de atividade relativo à alteração da permissão de registo.	24
3.10	Diagrama de atividade relativo à recolha e envio de dados provenientes dos vários sensores.	25
3.11	Diagrama de componentes do sistema.	26
3.12	Modelo da base de dados.	27
6.1	Representação física dos vários sensores interligados ao ESP32.	40
6.2	Representação física do <i>gateway</i>	40
6.3	Página <i>login</i> do <i>dashboard</i>	42
6.4	Página <i>registo</i> do <i>dashboard</i>	43
6.5	Página principal do responsável, no <i>dashboard</i>	43
6.6	Página principal do responsável, no <i>dashboard</i>	43
6.7	Página principal do responsável, no <i>dashboard</i>	44
6.8	Página principal do administrador, no <i>dashboard</i>	44
6.9	Código implementado no servidor <i>NodeJS</i> para o tratamento dos dados capturados pelos sensores.	50
6.10	<i>Prompt</i> após ser introduzido o comando <i>/help</i> quando não se tem o <i>login</i> efetuado.	51

6.11	<i>Prompt</i> após ser introduzido o comando <code>/help</code> quando se tem o <i>login</i> efetuado.	52
6.12	<i>Prompt</i> após ser introduzido o comando <code>/check</code>	52
6.13	Mensagem que detalha o último evento alarmante.	53
6.14	Mensagem que mostra os botões com as diversas opções.	53
6.15	Mensagem que mostra os botões com as diversas opções.	54
6.16	Mensagem de evento alarmante.	54
6.17	Gráfico de eventos alarmantes, com base nos dados recebidos através do uso dos sensores.	56
6.18	<i>E-mail</i> exemplo enviado a todos os responsáveis, no caso de um evento alarmante.	57
6.19	Trecho de código usado para a implementação de hyper text transfer protocol secure (HTTPS) com redirecionamento de hyper text transfer protocol (HTTP) para HTTPS.	60

Lista de Excertos de Código

A.1	Código utilizado no ESP32.	65
A.2	Código utilizado no <i>gateway</i>	68

Capítulo 1

Introdução

1.1 Enquadramento

Este documento enquadra-se na cadeira de Internet das Coisas, do primeiro ano de mestrado de engenharia informática.

1.2 Motivação

Portugal é um país que tem ficado cada vez mais envelhecido, especialmente na área do interior, longe das grandes cidades. Com isso é expectável que em zonas mais remotas, onde a quantidade de vilas e aldeias é superior, haja carência de cuidados à população devido à falta de meios para tal [1].

1.3 Objetivo

O objetivo deste trabalho é criar um sistema de vigilância inteligente para não só monitorizar os idosos, como alertar para possíveis desastres que possam ocorrer para os minimizar. Este sistema deverá não só ser utilizado pelo idoso, como também por cuidadores desse idoso, como cuidadores formais e informais, embora o principal foco deste trabalho seja cuidadores informais, já que estes representam a maioria em Portugal [1]. Para isso, foram tidos em conta os seguintes objetivos parciais:

1. Este sistema integra vários sensores no lar da pessoa, daí que foi necessário, recorrendo a um ESP32, ligar os vários sensores a este dispositivo;

2. Enviar os dados através do protocolo message queuing telemetry transport (MQTT), para um *gateway* num Orange Pi;
3. Através de uma Representational state transfer application programming interface (REST) application programming interface (API), enviar os dados do Orange Pi para um servidor remoto;
4. Disponibilização de uma *dashboard* que permite aos cuidadores (sejam estes formais ou não) visualizar os dados em tempo real.

Assim, esta implementação não só dará mais independência à pessoa, como aliviará o *stress* de trabalhadores de saúde e familiares encarregues do idoso

1.4 Organização do Documento

Este relatório encontra-se subdividido nas seguintes secções:

1. O primeiro capítulo – **Introdução** – introduz o trabalho desenvolvido, o respetivo enquadramento e organização do documento;
2. O segundo capítulo – **Estado da Arte** – faz um estudo a implementações ou investigações já existentes acerca do assunto em questão e procura encontrar uma possível implementação que não tenha sido diretamente aplicada por estes, mas sim complementando-os;
3. O terceiro capítulo – **Engenharia de Software e Arquitetura de Sistema** – detalha os diversos aspetos de engenharia de *software* relacionados com o presente trabalho. É nesta secção onde são criadas *personas* e *user stories* responsáveis pela criação dos requisitos funcionais e não funcionais do sistema. Após isso são mostrados os diagramas de engenharia de *software* e da arquitetura do sistema. Por último são elaboradas as tecnologias e materiais utilizados para o desenvolvimento do projeto.

Capítulo 2

Estado da Arte

Nesta seção, são descritos os estudos e pesquisas relevantes já realizados na área, bem como as principais tecnologias e abordagens utilizadas. O objetivo é oferecer um panorama atualizado do conhecimento existente na área, para que se possa identificar oportunidades para novas pesquisas ou inovações. Para isso são abordados seis estudos/implementações existentes, e, após isso, é apresentada a nossa ideia para complementar as ideias apresentadas. Para facilitar a comparação entre as diversas ideias com a nossa é mostrada uma tabela.

2.1 Soluções Existentes

2.1.1 Solução 1

Existem diversas aplicações de sistemas internet of things (IOT) para casas de pessoas idosas. Um deles, por exemplo, destina-se a pessoas com *mild cognitive impairment (MCI)* e demência [2]. Onde são aplicados múltiplos sensores para medir a temperatura, mas acima de tudo, medem as atividades diárias da pessoa e oferecem-lhe uma pulseira que pode ser usada em caso de emergência. Esta é uma solução muito boa que permite que o idoso mantenha um nível razoável de independência, sem sobrecarregar familiares e trabalhadores da área da saúde.

2.1.2 Solução 2

Uma nova tendência que tem aparecido nesta área são os *wearables*. Estes são peças de roupa ou acessórios que conseguem monitorizar a pessoa que os usa. O *paper* [3], aborda este assunto de uma forma bastante eficiente e

intuitiva, mostrando o potencial desta nova tecnologia relativamente à monitorização de idosos; com o mesmo objetivo que este nosso estudo, ou seja, garantir a independência do idoso. Este documento apresenta também novas tecnologias que se apresentam como um potencial alvo de investigações futuras, tais como tecnologia robótica, e aplicações integradas, no entanto, o principal foco continua a ser os *wearables*, e como estes devem continuar a ser investigados para serem mais acessíveis ao público geral.

2.1.3 Solução 3

Investigadores da Universidade da Beira Interior fizeram um estudo aprofundado acerca do estado da IOT em apoios domiciliários, no que chamam *enhanced living environment* [4]. Embora este artigo não apresente nenhum produto ou solução, discute os mais recentes avanços na área e apresenta alguns pontos em que a área poderia melhorar, oferecendo assim um *insight* valioso para qualquer investigação acerca da mesma. Falam sobre *wearables* e como o *smartphone* é uma ferramenta valiosa para o desenvolvimento da área. Também comparam diversas ferramentas existentes para o desenvolvimento de aplicações IOT, e elaboram uma tabela que torna a sua comparação acessível. Por último, estes investigadores acreditam que o futuro nesta área se encontra na segurança dos dados, já que com o crescimento de IOT, é expectável um crescimento exponencial dos dados transmitidos, e por consequência, uma necessidade maior de os manter seguros, visto conterem informações importantes. Outro trabalho futuro a mencionar foi na necessidade de criar mais sistemas ubíquos, para permitir que mais pessoas consigam usá-los sem se preocupar com o seu funcionamento.

2.1.4 Solução 4

Também é importante mostrar soluções práticas existentes na área. Uma delas é uma solução fornecida pela Delveco na área dos cuidados domiciliários [5]. Estes produtos integram uma variedade de tecnologias IOT, incluindo *wearables* para monitorizar os seus clientes. Este serviço oferece uma caixa com os vários sensores e aparelhos necessários para fazer esta infraestrutura funcionar; e estes incluem sensores de janela e de movimento, tomadas inteligentes, um botão de pânico, e vários aparelhos de medição para medir as condições de saúde da pessoa. Em geral, esta é uma solução muito prática e inteligente, que não só se preocupa com a segurança de casa como a segurança da saúde do idoso.

2.1.5 Solução 5

A Mokosmart, uma grande empresa de produtos IOT sediada na China, desenvolveu um artigo [6], onde descreve o problema do envelhecimento da população mundial, e foca-se na necessidade de fornecer independência aos mais idosos por meio da tecnologia fornecida pela IOT. Os seus métodos são muito interessantes, tentam automatizar certas funções como abrir a porta, desligar as luzes; mas também tentam controlar as atividades da pessoa, como, por exemplo, se o idoso se afastar demasiado da sua área de conforto, os cuidadores deste receberão uma notificação. Para mostrar as informações relativas à sua casa e às suas atividades, esta empresa oferece um *dashboard* customizável, tanto para os idosos, como para os cuidadores. Em suma, este artigo agiu como publicidade aos produtos da empresa, que são bastante interessantes, porém o mesmo pode ser feito com produtos de outras marcas, tornando esta uma solução bastante atraente e flexível.

2.1.6 Solução 6

Por último, importa destacar a apresentação de uma solução nacional e, por conseguinte, propomos a solução da empresa Tecnosenior, uma entidade portuguesa que desenvolveu o produto Contactto [7]. Mais uma vez, é ressaltada a relevância de conceder independência à população idosa, o que levou a empresa a investir na área de IOT e inteligência artificial. Conforme mencionado pelo seu CEO no artigo [8], esta solução é prática e inovadora, empregando o *IOT Watson* da IBM para a transmissão de dados dos dispositivos instalados na residência do idoso ou dos dados de GPS do telemóvel do mesmo, quando fora de casa. Para a tomada de decisões, é utilizada a inteligência artificial que, mediante os dados, executa determinadas ações, alcançando assim um novo patamar para esses sistemas. Um exemplo do uso da inteligência artificial é o seguinte: *"Dona Ligia saiu de casa para ir à aula de pintura que ocorre todas as quartas-feiras, mas já passou uma hora e ela ainda não chegou ao atelier. Enviarei uma mensagem de texto para a filha com a localização atual de dona Ligia, para que ela verifique se ocorreu algum imprevisto"*. Esta solução procura empregar sensores responsáveis por manter a residência segura (e.g., sensores de quedas, fumo, gás, temperatura, inundação, entre outros), não enfatizando tanto a facilidade de vida do idoso ou suas condições físicas. Essa abordagem é bastante relevante, sobretudo, devido ao emprego da inteligência artificial para a tomada de decisões, em vez de deixar esse trabalho aos cuidadores.

Existem diversas possibilidades no que se refere ao cuidado domiciliar de idosos, todas elas visando manter a independência dos mesmos e de não so-

brecarregar trabalhadores e familiares responsáveis por cuidar deles. Cada uma das soluções aborda o assunto de forma distinta, algumas focando-se na saúde física da pessoa, enquanto outras procuram oferecer uma melhor qualidade de vida e ainda outras decidiram concentrar-se na segurança domiciliar e atividades do idoso. Considerando todas as soluções apresentadas, há ainda espaço para abordar áreas que ainda não foram exploradas ou que merecem um destaque especial.

2.2 A Nossa Solução

Como verificado, grande parte da área dos cuidados de idosos recai sobre a monitorização de aspetos da sua saúde como as suas atividades ou nível sanguíneo e temperatura corporal. Também há implementações relacionadas com a vigilância da casa com sensores de janela e de presença.

O nosso projeto de monitorização de idosos em casa é uma implementação de um sistema de sensores IOT, composto por diversos dispositivos que permitem monitorizar o ambiente doméstico dos idosos. Para garantir a segurança do idoso, implementámos sensores de temperatura, sensores de gás e sensores magnéticos de portas e janelas, que detetam eventuais situações consideradas perigosas, permitindo a tomada de medidas preventivas de forma atempada. Além disso, os sensores de janela e porta permitem não só proteger a casa de entrada de estranhos, mas também controlar se o idoso saiu a horas irregulares, permitindo alertar os familiares e profissionais de saúde em caso de anomalias.

Todos os dados são apresentados num *dashboard* de acompanhamento intuitivo, que permite aos utilizadores monitorizar o estado do ambiente doméstico e do idoso. Além disso, implementámos notificações de alerta para situações alarmantes, que permitem aos familiares e profissionais de saúde agir prontamente em caso de emergência.

Para facilitar a visualização das soluções mostradas com a nossa, foi elaborada a tabela 2.1, que apresenta um conjunto de características para cada solução.

Em suma, a nossa solução é projetada para garantir a segurança e o bem-estar dos idosos, oferecendo uma abordagem abrangente e holística para a monitorização da sua casa.

	Sol. 1	Sol. 2	Sol. 3	Sol. 4	Sol. 5	Sol. 6	Nossa Sol.
Orientado aos idosos	✓	✓	✓	✓	✓	✓	✓
Focado na demência e MCI	✓	✗	✗	✗	✗	✗	✗
Aborda wearables	✓	✓	✓	✓	✗	✗	✗
Usa inteligência artificial	✗	✗	✗	✗	✗	✓	✗
Propõe o uso de dashboard	✗	✗	✗	✗	✓	✗	✓
Facilidade de vida do idoso	✗	✗	✗	✗	✓	✓	✗
Sensores de saúde corporal	✓	✓	✓	✓	✗	✗	✗
Sensores domiciliares	✗	✗	✓	✓	✓	✓	✓
Avalia as atividades do idoso	✓	✗	✗	✗	✓	✓	✓

Tabela 2.1: Comparação das várias implementações das soluções mostradas quando comparado à nossa solução.

Capítulo 3

Engenharia de Software e Arquitetura de Sistema

3.1 Introdução

O presente capítulo descreve a Engenharia de *Software* aplicada ao desenvolvimento do sistema. A secção 3.2 dá a conhecer as *personas* e *user stories*, essenciais para definir os requisitos funcionais e não funcionais que definem o sistema, na secção 3.3. A secção 3.4 apresenta as diversas tecnologias utilizadas para o desenvolvimento da aplicação. Nas secções 3.5 e 3.6 apresentam-se os diagramas de casos de uso e de atividade referentes às principais ações do sistema. Na secção 3.7 é apresentada a arquitetura do sistema que foi implementada. Por fim, a secção 3.8 expõe a base de dados desenvolvida num diagrama de classes.

3.2 *Personas e User Stories*

3.2.1 *Personas*

Personas são representações fictícias de utilizadores reais, que ajudam a entender as suas necessidades, comportamentos e objetivos. Elas são usadas para guiar o *design* de produtos e serviços de maneira mais centrada no utilizador, permitindo que as empresas atendam melhor as expectativas dos clientes e criem soluções mais eficientes. Neste documento são abordados três tipos de utilizadores: o **idoso** em questão para o qual o sistema foi projetado, a **pessoa responsável** por monitorizar o idoso, e por último o **administrador** responsável por configurar o sistema.

3.2.1.1 Persona 1: O Idoso

- **Nome:** Alexandre
- **Idade:** 75 anos
- **Profissão:** Aposentado
- **Descrição:** Alexandre vive sozinho na sua casa, numa aldeia remota, e com alguma dificuldade em realizar atividades diárias devido aos seus problemas de saúde. Ele valoriza a sua independência, mas está disposto a utilizar tecnologia para ajudá-lo a manter-se seguro e confortável na sua casa.

3.2.1.2 Persona 2: A Pessoa Responsável pelo Idoso

- **Nome:** Ângelo
- **Idade:** 32 anos
- **Profissão:** Engenheiro
- **Descrição:** Ângelo é o filho do Alexandre e é responsável por garantir que ele esteja seguro e cuidado na sua casa. Ele não mora com o pai devido à localização da casa do mesmo, mas visita-o regularmente e verifica se tudo está em ordem. Ele está interessado em utilizar tecnologia para monitorizar a casa do pai e receber alertas em caso de emergência.

3.2.1.3 Persona 3: O Administrador do Sistema

- **Nome:** Miguel
- **Idade:** 26 anos
- **Profissão:** Técnico informático
- **Descrição:** Miguel é um técnico informático responsável pela configuração e manutenção do sistema IOT na casa do Alexandre. Ele tem conhecimentos técnicos avançados e é capaz de solucionar problemas técnicos em tempo hábil. Ele consegue entender as necessidades do Alexandre e do Ângelo e personalizar o sistema para atender às suas necessidades, de forma a manter a segurança do idoso, mantendo ao mesmo tempo, a sua independência.

3.2.2 *User Stories*

User stories são histórias que descrevem o que o utilizador deseja fazer com o produto, escritas de uma forma simples e acessível. O objetivo é compreender as necessidades do utilizador de uma forma clara e sucinta, permitindo que seja possível criar requisitos funcionais e não funcionais que satisfaçam essas necessidades. As *user stories* também são úteis para acompanhar o progresso do desenvolvimento e garantir que as funcionalidades desenvolvidas são úteis e valorizadas pelo utilizador final. Estas *user stories* são:

- Como Alexandre, quero poder visualizar de forma fácil as informações relativas ao meu domicílio, tal como temperatura das divisões, clima da região, nível de gás na cozinha, entre outros;
- Como Alexandre, quero que os sensores magnéticos me protejam durante a noite para que pessoas estranhas não entrem na minha casa de forma despercebida;
- Como Alexandre, quero ser notificado em caso de emergência, para poder colocar-me em segurança o mais rapidamente possível.
- Como Ângelo, quero poder fazer *login* no *dashboard* de forma simples e rápida, em qualquer lugar;
- Como Ângelo, quero poder visualizar as informações relativas aos sensores colocados ao longo da casa do meu pai, assim como ter um histórico de todas as ocorrências alarmantes;
- Como Ângelo, quero filtrar os eventos alarmantes no *log* por data e por ocorrências;
- Como Ângelo, quero poder ter acesso ao *dashboard* em qualquer dispositivo, tanto num computador, como num telemóvel;
- Como Ângelo, quero ser informado mal ocorra algum evento alarmante na casa do meu pai.
- Como Miguel, quero poder fazer *login* no *dashboard* de forma simples e rápida, em qualquer lugar;
- Como Miguel, quero poder registar novos clientes no servidor para que estes tenham acesso ao *dashboard* remotamente;
- Como Miguel, quero poder eliminar utilizadores que já não sejam clientes.

3.3 Requisitos de Utilizador

Os requisitos de utilizador englobam pequenas frases e expressões, escritas numa linguagem natural, onde são especificados quais os serviços e/ou funcionalidades a que o sistema deve responder, bem como os constrangimentos sobre os quais deve operar. Os requisitos de utilizador podem ir de definições amplas do que o sistema deve responder até descrições detalhadas, precisas e coerentes das funcionalidades do sistema. Estes podem ser definidos em duas categorias: funcionais e não funcionais [10, p. 102].

3.3.1 Requisitos Funcionais

Os requisitos funcionais definem as funções que o sistema deve ser capaz de efetuar. Os requisitos funcionais identificados para a aplicação a desenvolver estão divididos em dois grupos, para o *dashboard* e para o sistema, e estes são:

3.3.1.1 *Dashboard*

1. **RF1:** O *dashboard* deve ter um portal de *login*, permitindo a visualização dos dados recolhidos pelo sistema;
2. **RF2:** O *dashboard* deve permitir a visualização dos *logs* de eventos através de uma interface *web*;
3. **RF3:** O *dashboard* deve permitir ao administrador ligar o acesso ao portal de registo na plataforma aos utilizadores;
4. **RF4:** O *dashboard* deve enviar alertas sempre que haja um evento alarmante, para os utilizadores registados no domicílio em questão;
5. **RF5:** O *dashboard* deve permitir, ao administrador, a alteração dos valores considerados limites, antes de despoletar um evento alarmante;
6. **RF6:** O *dashboard* deverá permitir ao utilizador filtrar o *log* por data e por ocorrências;
7. **RF7:** A base de dados deve guardar por efeitos de segurança, os *logs* de *login* dos utilizadores, incluindo o administrador;
8. **RF8:** O *dashboard* deve permitir, ao utilizador, a visualização dos estados ativo/desativo de cada alarme;

9. **RF9:** O *dashboard* deve permitir, ao utilizador, a visualização dos dados entre várias divisões (cozinha, sala, quarto e *hall*);
10. **RF10:** O *dashboard* deve permitir, ao utilizador, a visualização dos dados relativos ao clima, ao nível de gás e o estado dos sensores magnéticos.
11. **RF11:** O *dashboard* deve permitir, ao utilizador, trocar a divisão visualizada através de selecionadores;
12. **RF12:** O *dashboard* deve permitir, ao utilizador registar-se, assim que o administrador o permitir.

3.3.1.2 Sistema

1. **RF1-s:** O sistema deverá fazer uma ligação à base de dados sempre que for necessário adicionar, remover ou editar dados;
2. **RF2-s:** O sistema deverá permitir a introdução de eventos na base de dados;
3. **RF3-s:** O sistema deve armazenar na base de dados um *log* com eventos que sejam considerados alarmantes;
4. **RF4-s:** O sistema deve transmitir os dados através de REST API, recorrendo ao protocolo hyper text transfer protocol (HTTP) sobre TCP/IP;
5. **RF5-s:** Os dispositivos deverão conectar-se ao *gateway* através de MQTT;
6. **RF6-s:** O *gateway* deverá estar na mesma rede local que os dispositivos IOT.

3.3.2 Requisitos Não Funcionais

Os requisitos não funcionais descrevem restrições e prioridades do sistema. De seguida, são enumerados os requisitos não funcionais da aplicação desenvolvida:

1. **RNF1 – Eficiência:** O *software* deve estar implementado com vista à sua eficiência, realizando tarefas unicamente quando solicitado, tais como aceder à base de dados ou enviar avisos;

2. **RNF2 – Organização:** O *software* e as suas componentes informáticas, tais como a base de dados e os ficheiros que contêm o código-fonte, devem estar bem organizados;
3. **RNF3 – Layout:** O *layout* da aplicação deve ser desenvolvido de maneira responsiva e que permita a utilização por vários dispositivos;
4. **RNF3 – Intuição:** O sistema deve estar arquitetado de tal forma intuitiva que um utilizador não deva demorar mais do que sete minutos a aprender a utilizar o *dashboard*;
5. **RNF5 – Segurança-1:** Para utilizar o sistema, o utilizador deve estar autenticado. As credenciais de acesso devem estar armazenadas de maneira segura na base de dados;
6. **RNF6 – Segurança-2:** Deve estar implementado o protocolo hyper text transfer protocol secure (HTTPS) para garantir um nível de segurança acrescido;
7. **RNF7 – Confidencialidade:** Os dados armazenados na base de dados são confidenciais, e apenas devem ser vistos por pessoal autenticado;
8. **RNF8 – Conectividade:** O sistema deve estar dotado de ligação à Internet, para transmissão dos dados em tempo real e para a sua visualização ser possível;
9. **RNF9 – Resiliência:** O sistema deve ser resiliente a erros que possam ocorrer durante a sua execução, tal como *inputs* mal formatados, entre outros;
10. **RNF10 – Energia:** Os sensores e o *hotspot* deverão ser alimentados através da eletricidade disponível na casa onde o sistema é instalado, ligados a tomadas elétricas.

3.4 Tecnologias utilizadas

Nesta secção são apresentadas as diversas tecnologias e materiais utilizados na elaboração do projeto de forma breve. Esta secção vai ser dividida em duas, ao nível do domicílio do idoso, e ao nível do servidor, visto serem duas partes relativamente isoladas. A secção 3.7, elabora mais aprofundadamente os usos destas tecnologias e como elas se interligam.

3.4.1 Tecnologias utilizadas no domicílio do idoso

Na residência onde o sistema será aplicado é onde irão se situar os diversos sensores e o sistema que gere a informação destes. Os materiais utilizados foram:

- Sensores:
 - dois sensores de temperatura;
 - dois sensores magnéticos.
 - um sensor de gás;
- Um ESP32;
- Orange Pi PC.

As tecnologias utilizadas foram as seguintes:

- Python;
- VSCode + PlatformIO (em detrimento do Arduino IDE);
- MQTT *broker* (Mosquitto).

3.4.2 Tecnologias utilizadas no servidor

É o servidor o responsável por gerir todos os dados das residências na base de dados e também por gerir os *dashboards* utilizados para consultar informações acerca de cada casa.

O único material utilizado para esta parte foi um servidor DigitalOcean, cujas especificações podem ser encontradas na tabela 3.1.

TODO

Sistema Operativo	Ubuntu 22.04
CPU	4 cores
Memória RAM	8GB
Armazenamento	160GB

Tabela 3.1: Especificações do servidor DigitalOcean.

As tecnologias utilizadas para a elaboração da base de dados e do *dashboard* foram:

- CSS;

- Express.js;
- HTML;
- JSON;
- MySQL;
- Node.js.

3.5 Diagramas de Casos de Uso

Os diagramas de casos de uso procuram representar visualmente os requisitos do sistema. Nesta secção, apresentam-se os casos de uso para as principais funções do administrador, utilizador, bem como acerca da parte de recolha e envio de dados pelo ESP32.

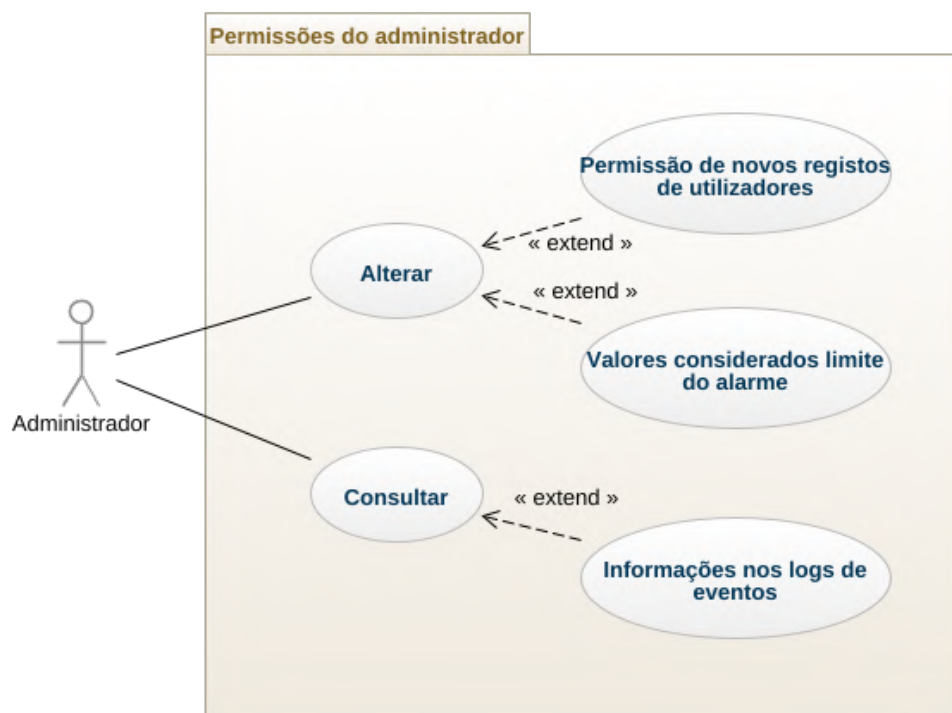


Figura 3.1: Diagrama de caso de uso para as principais funções do Administrador.

A figura 3.1 ilustra as opções disponíveis para um administrador aquando a sua autenticação no sistema. Mais especificamente, indica que o administrador tem a capacidade de alterar as permissões de novos registos de utilizadores, bem como informações nos *logs* de eventos e os valores limite do alarme.

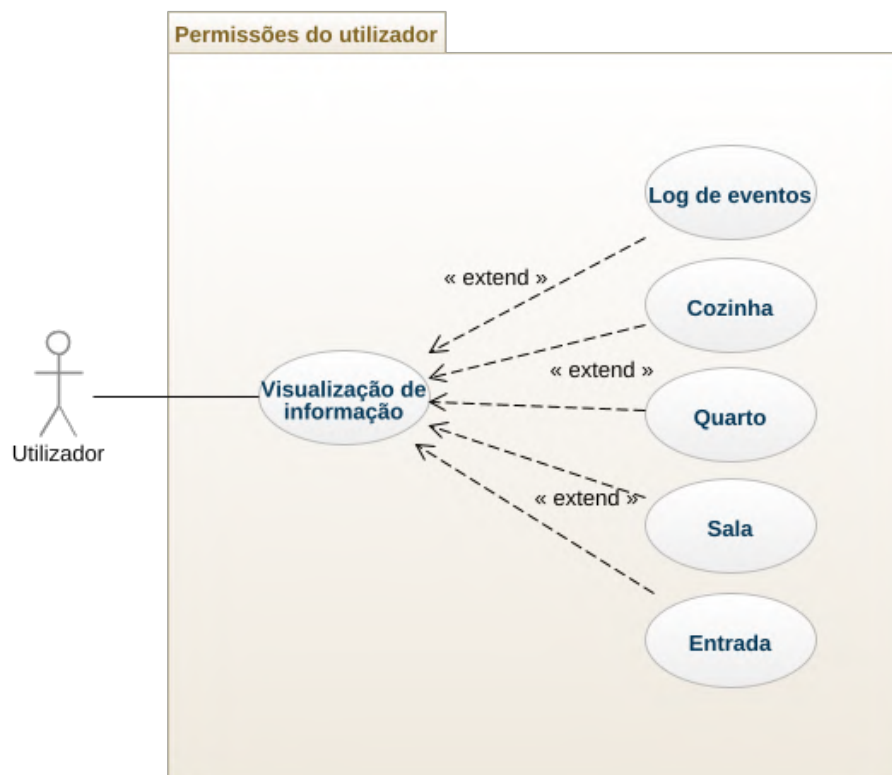


Figura 3.2: Diagrama de caso de uso para as principais funções do Utilizador.

A figura 3.2 ilustra as opções disponíveis para um utilizador aquando da sua autenticação no sistema. Este diagrama incide nas capacidades de visualizar informações sobre diferentes áreas da casa, bem como sobre o *log* de eventos.

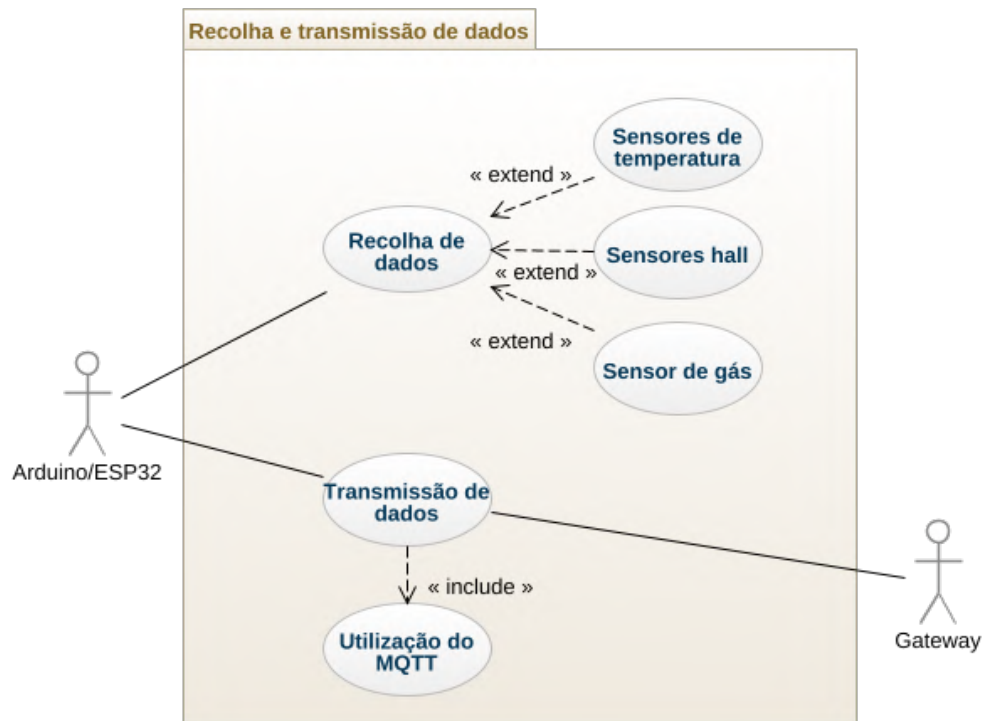


Figura 3.3: Diagrama de caso de uso sobre a recolha e transmissão de dados.

O diagrama apresentado na figura 3.3 ilustra as capacidades disponíveis para um dispositivo ESP32, representando a capacidade de recolher dados de vários sensores, bem como a sua transmissão através do protocolo MQTT.

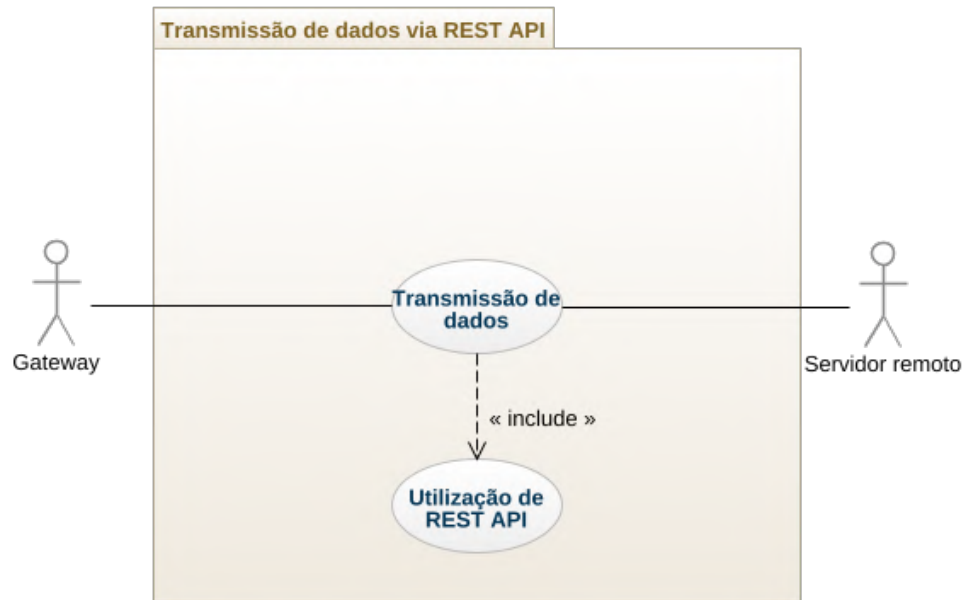


Figura 3.4: Diagrama de caso de uso sobre a transmissão de dados do Orange Pi para o servidor.

O diagrama apresentado na figura 3.4 ilustra a transmissão de dados de um *gateway* para um servidor remoto. Para a transmissão de dados do Orange Pi para o servidor remoto, é utilizada uma REST API. O uso desta API permite que os dados sejam enviados e recebidos de forma segura e confiável, além de ser uma forma padronizada de comunicação entre diferentes sistemas.

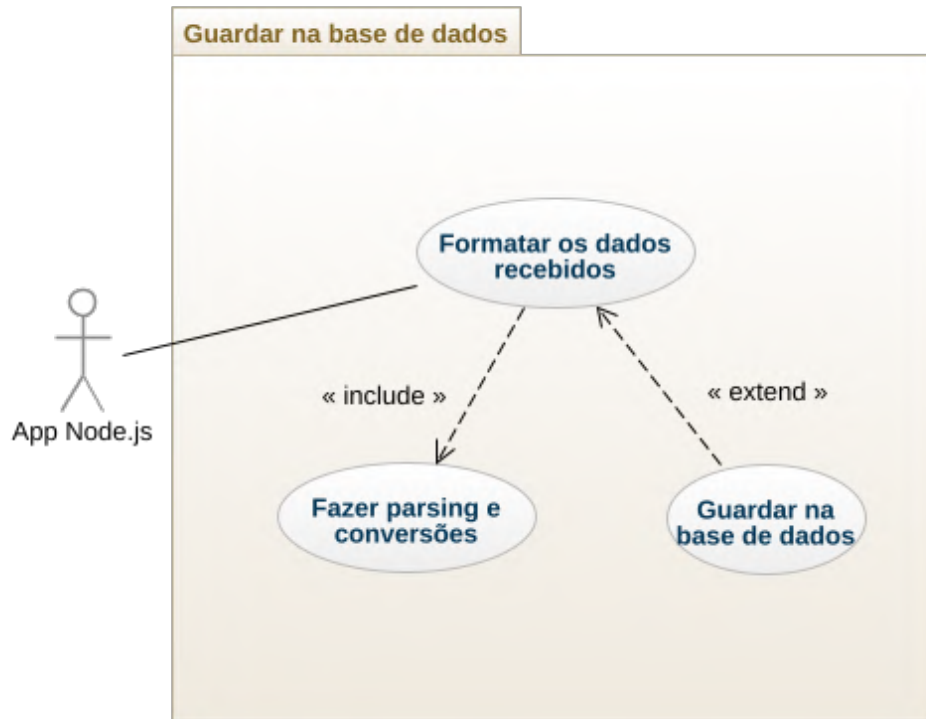


Figura 3.5: Diagrama de caso de uso sobre a formatação de dados e posterior registo na base de dados.

O diagrama apresentado na figura 3.5 ilustra a forma como a aplicação Node.js formata os dados recebidos. Para guardar os dados na base de dados, é necessário efetuar *parsing* e conversões, de modo que os dados estejam num formato que seja apropriado para ser guardado na base de dados implementada. A opção de guardar na base de dados é opcional visto que existem dados temporários que não são relevantes (e.g., a informação do clima da zona).

3.6 Diagrama de Atividade

Os diagramas de atividade, tal como o nome indica, visam demonstrar a lógica de um algoritmo, de modo a simplificar o funcionamento da aplicação, para determinada ação. Estes diagramas também pretendem modelar elementos da arquitetura de *software*, como métodos, funções e operações realizadas [11]. Nesta secção, apresentam-se os diagramas de atividade para

as principais atividades, visto que ilustram o funcionamento geral da plataforma.

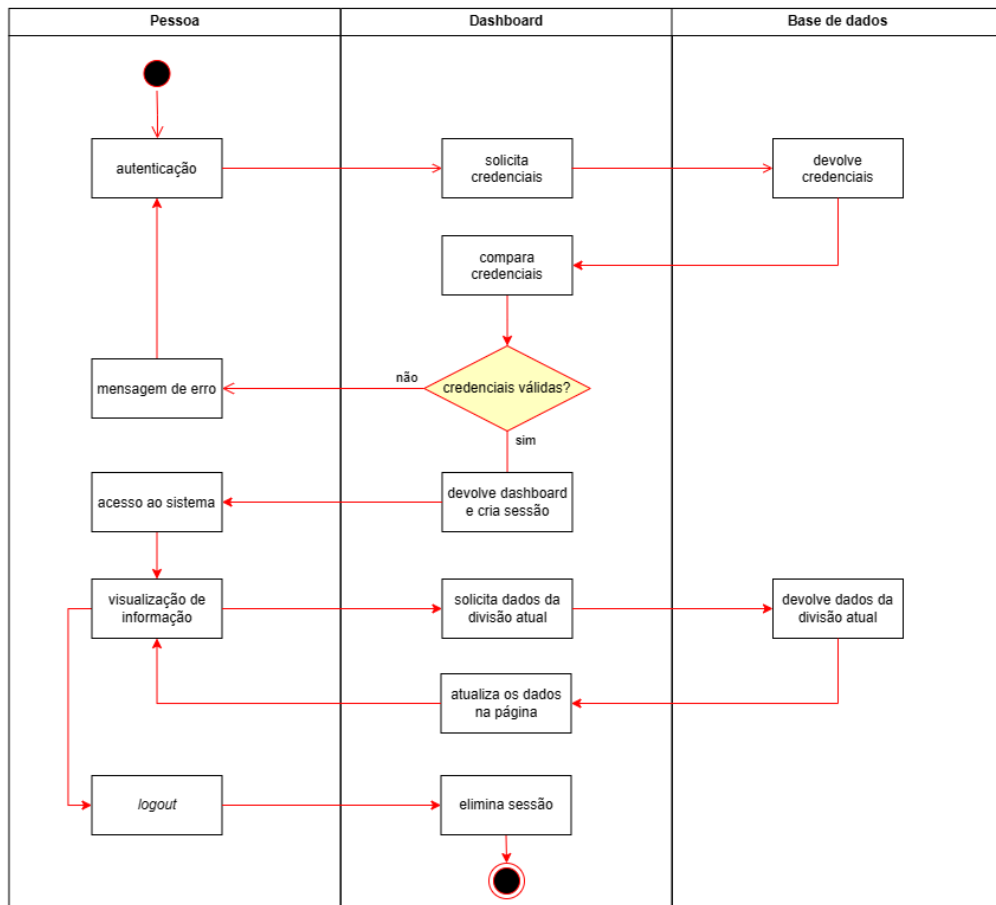


Figura 3.6: Diagrama de atividade relativamente à consulta de informação por parte do utilizador.

O diagrama 3.6 representa a visualização de informação por parte do utilizador, seja este uma pessoa formal ou não. Para tal, é necessário passar pelo processo de autenticação, sendo necessário a introdução das suas credenciais. Para visualizar a informação, o *dashboard* solicita, em tempo real, os dados presentes na base de dados, e atualiza de forma dinâmica o mesmo.

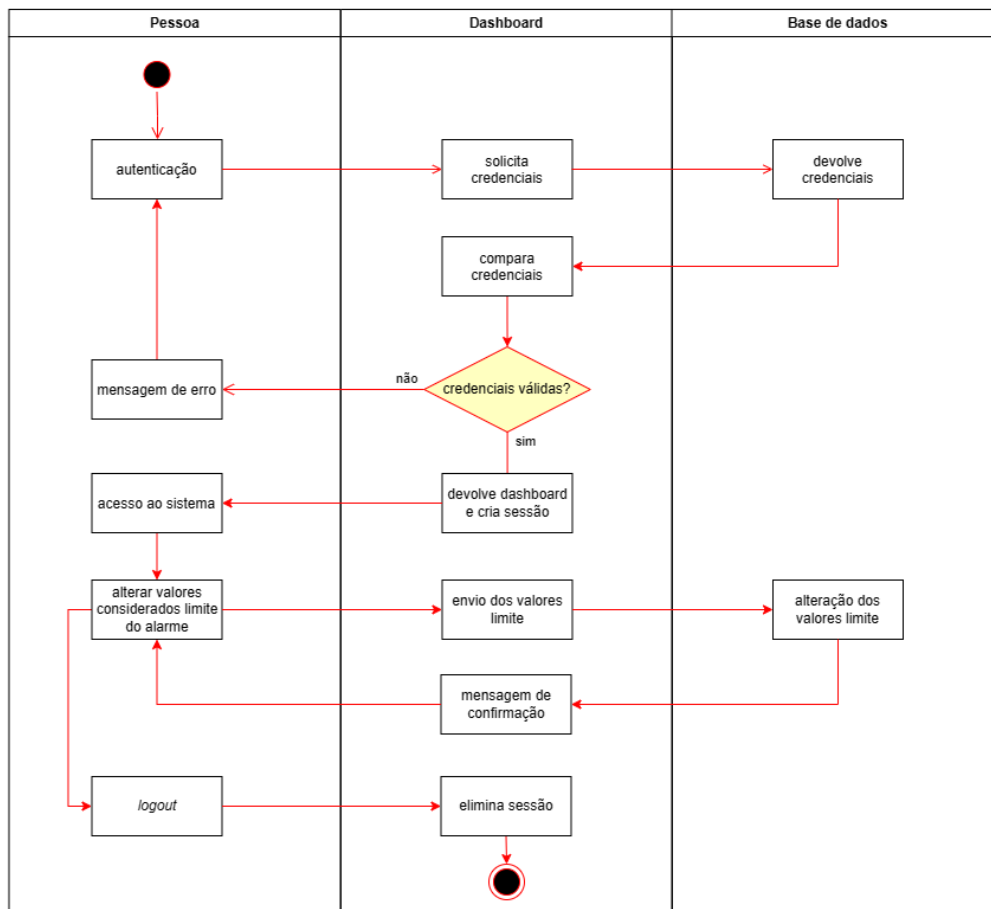


Figura 3.7: Diagrama de atividade relativamente à alteração dos valores considerados limites.

A figura 3.7 ilustra a alteração dos valores considerados limite, para despoletar um evento/alarme, permitindo ao administrador alterar os mesmos. Este processo necessita de autenticação, e que o utilizador em questão seja o administrador da plataforma. O *dashboard* está dotado desta funcionalidade, que pode ser útil para a configuração do ambiente em que o sistema está inserido.

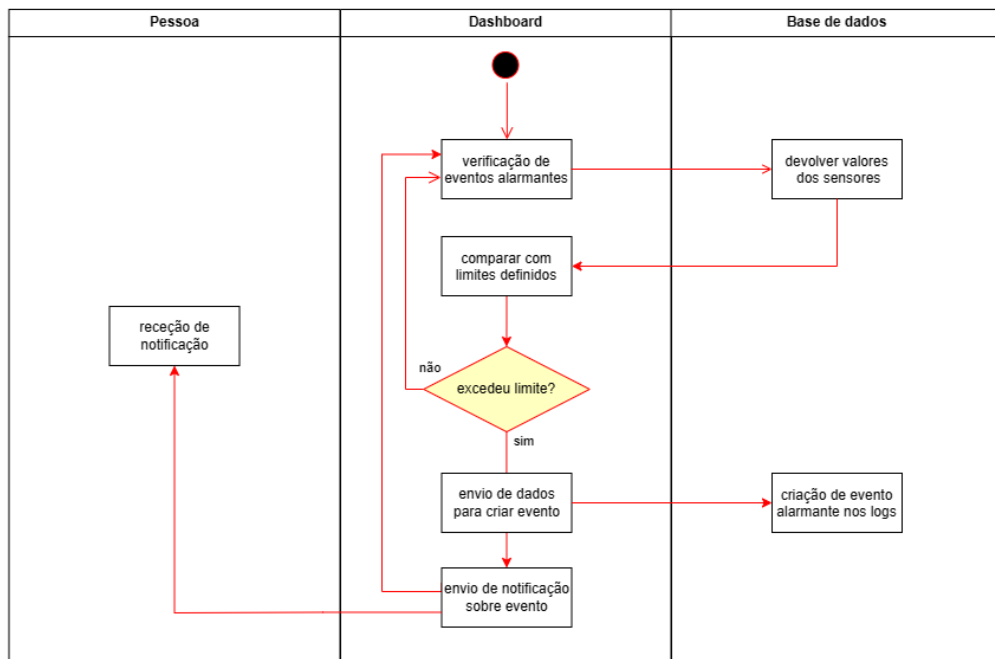


Figura 3.8: Diagrama de atividade relativamente ao envio de um evento.

No diagrama 3.8 está representado o diagrama de atividade relativamente à deteção e envio de uma notificação, quando é despoletado um evento considerado alarmante. Desta forma, o *dashboard* verifica constantemente os valores recebidos pela base de dados, e assim que sejam detetados valores considerados anormais, é enviada uma notificação para os utilizadores do sistema, sejam formais, não formais ou administradores.

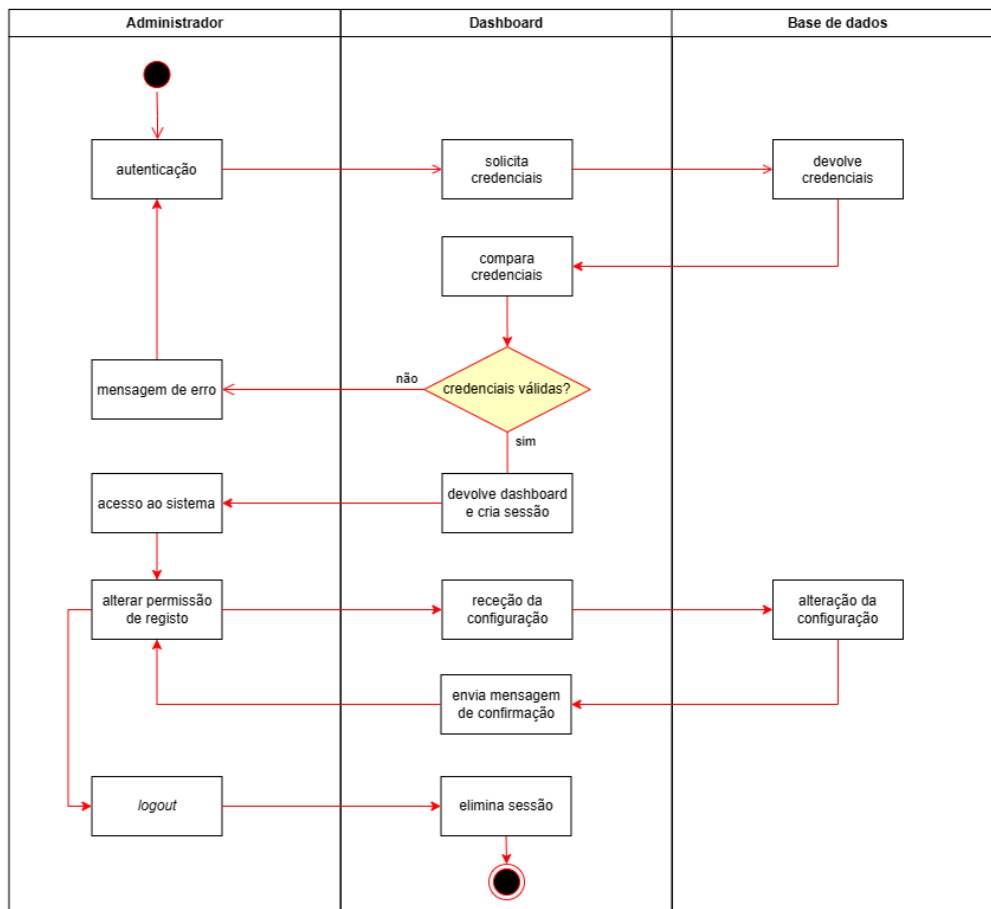


Figura 3.9: Diagrama de atividade relativo à alteração da permissão de registo.

O diagrama 3.9 representa a alteração, por parte de um administrador, da permissão de registo, permitindo ou negando o registo de novos utilizadores na plataforma. Desta forma, é possível controlar os acessos mais eficientemente, dotando a plataforma de capacidades acrescidas de segurança.

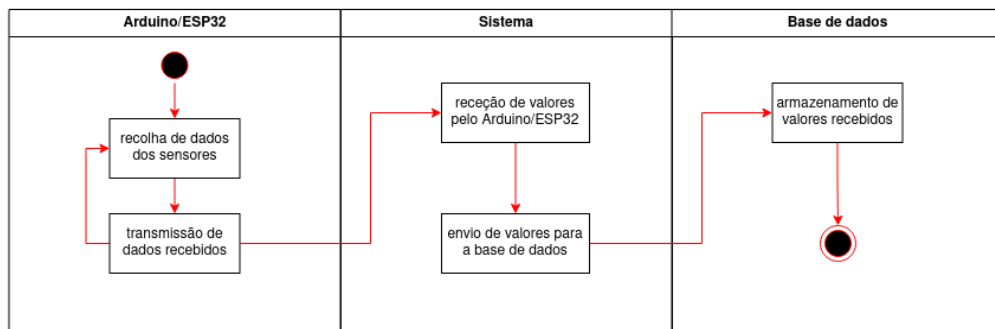


Figura 3.10: Diagrama de atividade relativo à recolha e envio de dados provenientes dos vários sensores.

A figura 3.10 ilustra o envio de dados recolhidos dos vários sensores por parte do ESP32 para o sistema, sendo estes posteriormente trabalhados e enviados num formato compatível para a base de dados, onde vão ser registados e armazenados indefinidamente.

3.7 Arquitetura do Sistema

A arquitetura do sistema relaciona os vários componentes que constituem o sistema, permitindo elaborar uma visão técnica deste.

O diagrama de componentes do sistema visa mostrar o relacionamento que existe entre as várias componentes que o constituem. Estes diagramas são particularmente úteis, pois ajudam a visualizar a estrutura física do sistema e enfatizam o comportamento do serviço quanto à interface [12]. A figura 3.11 demonstra a estrutura das componentes utilizadas e que compõem a arquitetura do sistema a ser implementado.

A leitura desta arquitetura deve começar no **Sensors Node**, onde os vários sensores de gás, temperatura e *hall* estão ligados a um ESP32 com capacidades de ligação sem fios. A recolha de dados é feita neste componente, e através do protocolo MQTT, é possível enviar os dados para um MQTT *broker* (neste caso, o Mosquitto), instalado no Orange Pi, e representado como **Gateway** na figura. O **Servidor remoto** tem uma aplicação Node.js, desenvolvida em Express.js, com uma implementação REST API, que permite ao Orange Pi enviar os dados recolhidos, bem como disponibilizar através de HTTPS o *dashboard* para visualização dos dados. Além disso, a aplicação é responsável por tratar os dados, comunicando com a base de dados através do MySQL, sempre que necessário. Por fim, qualquer cliente, representado como **Desktop** nesta figura, pode aceder à *dashboard* através de HTTPS.

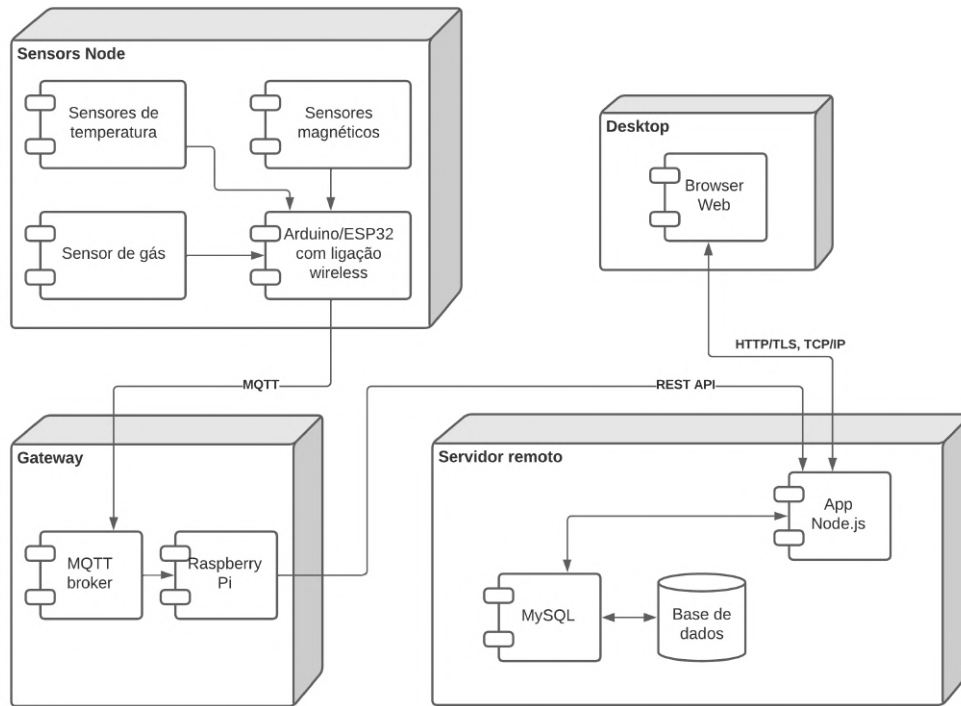


Figura 3.11: Diagrama de componentes do sistema.

3.8 Base de Dados

O principal objetivo deste projeto foi desenvolver uma aplicação IOT que permitisse visualizar e monitorizar os dados enviados através de vários sensores num dashboard em permanente atualização. Para tal, foi necessário implementar uma base de dados de suporte. A base de dados é composta por cinco tabelas: a tabela **users** armazena as credenciais dos vários utilizadores registados na aplicação; a tabela **sensor_data** permite guardar os dados enviados pelo Orange Pi no servidor, para serem posteriormente trabalhados; a tabela **alert_values** guarda os valores de limiar, que quando ultrapassados, despoletam um evento; a tabela **events** guarda os vários eventos ocorridos, isto é, quando um sensor ultrapassa o valor limiar previamente definido; a tabela **logs** guarda na base de dados a informação sobre os últimos acessos à *dashboard*, como um mecanismo de controlo.

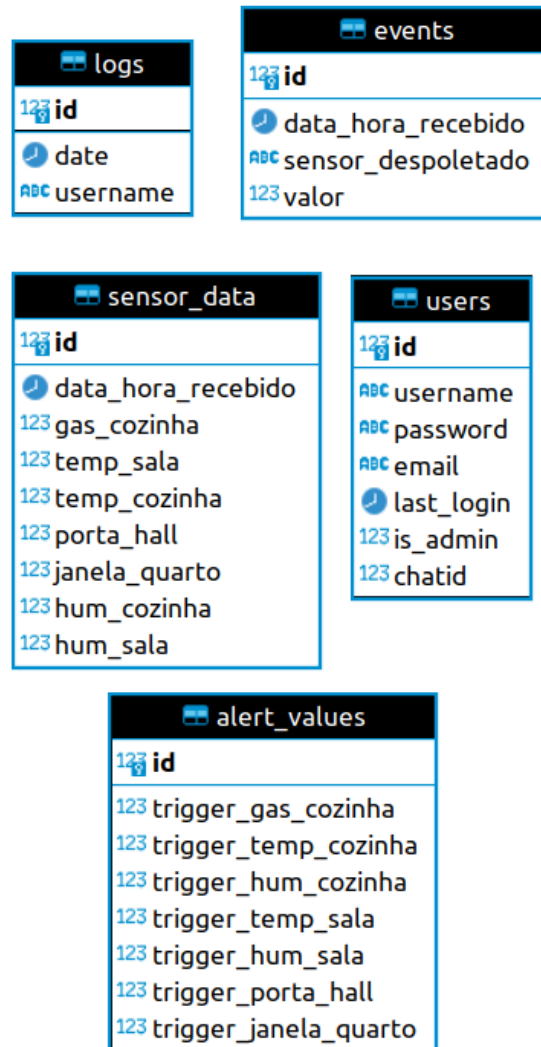


Figura 3.12: Modelo da base de dados.

3.9 Conclusões

O presente capítulo reflete sobre toda a Engenharia de *Software* trabalhada neste projeto. São apresentados os requisitos funcionais e não funcionais, bem como todos os diagramas representativos da plataforma. Finalmente, apresenta-se a arquitetura do sistema e o esquema da base de dados desenvolvida.

Capítulo 4

Interoperabilidade

Este capítulo aborda o tema da interoperabilidade no contexto do nosso sistema. Serão abordados os diversos problemas acerca de interoperabilidade e como o nosso sistema procura resolver tais problemas.

4.1 Interoperabilidade de Dispositivos

Um dos desafios que podem afetar o nosso sistema é a interoperabilidade de dispositivos. Diferentes dispositivos podem utilizar protocolos de comunicação diferentes ou seguir padrões próprios, o que pode dificultar a integração com o sistema existente. Esse problema está em parte resolvido devido aos dispositivos escolhidos, sendo estes sensores analógicos, não existe grande problema de interoperabilidade de dispositivos. Porém, mesmo que fossem introduzidos novos sensores mais avançados com a sua própria API, tais como sensores da Shelly ou Bosch, estes contêm APIs, que são fáceis de usar e que compatibiliza dispositivos que outrora têm diferentes arquiteturas.

4.2 Interoperabilidade de Plataforma

Outro desafio importante é a interoperabilidade de plataforma. Diferentes plataformas de nuvem ou sistemas operativos podem ter requisitos específicos, ou incompatibilidades que dificultam a integração perfeita do sistema. Para lidar com esse problema, decidiu-se projetar o sistema de forma modular e independente da plataforma. Isso permite que o sistema seja implementado em várias plataformas de nuvem e sistemas operativos, adaptando-se facilmente a diferentes ambientes. Assim, não importa se o sistema está a correr em sistemas operativos como, por exemplo, Windows

ou Linux, o sistema irá estar sempre a funcionar. Isso está provado pelo facto dos sistemas todos terem sido desenvolvidos em Windows, e depois foram facilmente passados para um sistema Linux Ubuntu, que não teve que ser configurado de nenhuma forma e conseguiu dar *deploy* do sistema sem qualquer problema. Também foram testados dois diferentes provedores de nuvem, sendo o primeiro o Okeanos e depois o DigitalOcean, e novamente não houveram quaisquer problemas.

4.3 Interoperabilidade Semântica

A interoperabilidade semântica é de facto um aspeto crítico a ser considerado. É comum que diferentes dispositivos e sistemas tenham representações de dados distintas ou utilizem terminologias diferentes para descrever as mesmas informações. Essa divergência pode levar a problemas de interpretação e entendimento dos dados.

Para superar esse desafio, é importante estabelecer uma ontologia ou vocabulário comum, garantindo que todos os dispositivos e sistemas envolvidos compartilhem a mesma compreensão dos dados. No caso do nosso sistema de monitorização remota de idosos, optamos por utilizar o formato JavaScript object notation (JSON) para a formatação dos dados. Com o JSON, foi possível associar cada sensor a uma divisão específica e a uma função particular, por exemplo, o sensor de gás da cozinha está associado à divisão *cozinha* e à função *gás*. Essa abordagem garante que os dados tenham um significado consistente em todo o sistema, facilitando a interpretação e análise dos dados coletados.

4.4 Interoperabilidade Sintática

A interoperabilidade sintática diz respeito à conformidade das mensagens e formatos de dados trocados entre os diferentes componentes do sistema. Variações na sintaxe das mensagens podem dificultar a comunicação correta entre os dispositivos e a integração com a base de dados.

Para solucionar esse problema, definimos um formato de mensagem padronizado, como o JSON, para garantir a consistência e uniformidade na comunicação. Isso permite que os dispositivos e a infraestrutura do sistema interpretem e processem as informações corretamente. O uso de uma REST API também contribuiu de forma significativa para a consistência dos dados durante a comunicação entre o domicílio e o servidor na nuvem, e também

entre o servidor e a aplicação de Telegram, visto que apenas são utilizados pedidos HTTP *POST* e *GET* para ser realizada a comunicação.

4.5 Interoperabilidade de Rede

A interoperabilidade de rede aborda os desafios de comunicação num ambiente distribuído, onde diferentes dispositivos e sistemas podem estar conectados por redes heterogêneas. Problemas como latência, perda de pacotes ou instabilidade de rede podem afetar a transmissão eficiente e confiável dos dados de monitorização.

Para lidar com essas questões, todos os dispositivos estão conectados à mesma rede, sendo esta rede uma rede própria para os dispositivos IOT onde todos se conectam pelo protocolo IEEE 802.11g (i.e., Wi-Fi).

Capítulo 5

Ética e Segurança

A ética pode ser definida como parte da filosofia que estuda os fundamentos da moral, ou um conjunto de regras de conduta de um indivíduo, ou de um grupo. O presente capítulo aborda um pequeno código de ética, descrito na secção 5.1, bem como a segurança da informação tratada pelo sistema, na secção 5.2.

5.1 Código de Ética

Esta secção pretende estabelecer diretrizes claras e princípios fundamentais para garantir o uso ético e responsável do nosso sistema IOT. O objetivo é orientar o processo de desenvolvimento e assim garantir que este se desenvolve de forma ética.

5.1.1 Sistema Domiciliar

5.1.1.1 Deontologia Profissional (ação)

- O código de ética deve ser rigorosamente cumprido;
- Deve haver a mais absoluta transparência.

5.1.1.2 Sistema e Sociedade

- Consentimento informado
 - O cliente deve ser informado de todas as nuances relacionadas com os dados recolhidos e sobre como os sensores monitorizam o seu domicílio e tanto o idoso, como os responsáveis por ele devem consentir a recolha destes dados;

- Sempre que o sistema domiciliar for atualizado de qualquer forma ou maneira, tanto o idoso, quanto as pessoas responsáveis por este devem ser informadas nas alterações do mesmo;
- Qualquer recomendação de uma atualização de um sistema domiciliar deve ser originada pelos clientes e apenas pelos clientes.
- Privacidade
 - Os dados recolhidos através dos vários sensores apenas podem ser visualizados por pessoas credenciadas para tal ação, sejam eles cuidadores formais, não formais ou os próprios utilizadores onde o sistema está instalado;
 - A interconectividade dos vários sensores não pode ser manipulada, bem como os dados por estes enviados devem ser mantidos íntegros, desde a sua recolha à receção e tratamento pelo sistema;
 - Os administradores do sistema não podem manipular os dados enviados pelos sensores.
- Confiabilidade e Resiliência
 - O sistema deve ser extensamente testado, verificando cada requisito e ação múltiplas vezes, antes deste ser implementado no domicílio;
 - O sistema deve ser rapidamente atualizado assim que for descoberto algum erro que possa pôr em causa a *performance* deste.
- Transparência
 - As várias práticas e protocolos implementados devem ser explicados de forma clara e aberta aos seus utilizadores;
 - O código desenvolvido deve estar em formato *open source*, para que qualquer parte interessada possa verificar o funcionamento do sistema para uma determinada ação por este tomada.
- Segurança Física
 - Os sensores aplicados no domicílio não podem ter qualquer impacto nos restantes componentes do mesmo, nem mesmo afetar negativamente o(s) residente(s).
 - Qualquer dano causado ao(s) residente(s), e cuja responsabilidade seja passível de ser apurada, deve ser imputada a quem direito;

- A segurança física dos dispositivos deve ser salvaguardada.
- Consumo
 - O sistema implementado não deve consumir demasiada energia ao domicílio.

5.1.2 *Dashboard*

- Informações relevantes
 - O *dashboard* não deve fornecer mais informações do que as estritamente necessárias.
- Acesso seguro
 - É essencial garantir que o acesso ao *dashboard* seja seguro e protegido. A autenticação e a criptografia devem ser implementadas para proteger a privacidade e a confidencialidade dos dados do lar do idoso.
- Histórico e análise de dados
 - O *dashboard* pode permitir que os responsáveis acessem o histórico de dados coletados ao longo do tempo, porém, não deve fornecer mais do que é permitido visualizar, a um qualquer utilizador.

5.2 Segurança da informação

A IOT revolucionou o mundo interligado, proporcionando uma ampla gama de possibilidades e avanços tecnológicos. No entanto, essa transformação digital também trouxe à tona desafios cruciais no que diz respeito à segurança. Como destaca o ditado popular, “*O S em IoT significa segurança*”. Portanto, é imprescindível enfrentar os problemas relacionados à segurança na transmissão de dados entre os sistemas e os protocolos utilizados.

Esta secção visa fornecer uma análise completa das vulnerabilidades de segurança potenciais do nosso sistema e, em seguida, detalhar as medidas adotadas para garantir a sua segurança efetiva contra essas vulnerabilidades. É fundamental identificar e compreender as possíveis ameaças que o sistema pode enfrentar, a fim de implementar medidas preventivas e corretivas adequadas.

5.2.1 Ataque de Homem no Meio (*Man-in-the-Middle Attack*)

Um ataque de homem no meio (*Man-in-the-Middle Attack*), é uma forma de ataque cibernético em que um terceiro mal-intencionado intercepta e escuta a comunicação entre duas partes legítimas sem o conhecimento ou consentimento delas. Nesse tipo de ataque, o atacante posiciona-se entre o remetente e o destinatário, interceptando os pacotes de rede (tornando-se assim um ataque que funciona maioritariamente na camada de rede), e tornando-se um intermediário invisível, capaz de ler, alterar e até mesmo injetar novos dados na comunicação. O ataque geralmente ocorre em redes de comunicação, como a Internet, e é facilitado pela falta de autenticação e criptografia adequadas.

No contexto deste projeto, visto que o sistema domiciliar é local, dificilmente poderá haver um ataque deste calibre no sistema, porém, na comunicação entre o domicílio e a *cloud*, é possível um atacante interceptar os dados dos sensores com ferramentas como, por exemplo, Ettercap, PacketCreator e Wireshark.

Assim, o sistema foi desenhado e pensado para tentar combater esta vulnerabilidade - na implementação da ligação do gateway MQTT à REST API, através da internet, as comunicações dos vários sensores, enviadas através de uma estrutura de dados do tipo JSON são cifradas com AES-128-CBC, com uma chave secreta previamente partilhada de forma segura, resolvendo assim um ataque de homem no meio.

5.2.2 *Malware* – Criação de *Botnets*

Uma *botnet* é uma rede de dispositivos infetados por *malware*, um *software* malicioso, controlados remotamente por um atacante, geralmente chamado *botmaster*. Os atacantes podem infetar um grande número de dispositivos com *malware*, controlando-os e transformando-os em *bots*. Esses dispositivos comprometidos são então usados para formar a *botnet*, controlada pelo atacante.

No contexto dos dispositivos IOT, as *botnets* podem ser especialmente preocupantes, pois os dispositivos geralmente possuem recursos limitados de processamento e segurança, tornando-os alvos atraentes para os atacantes. Além disso, muitos dispositivos IOT são projetados com ênfase na conectividade e funcionalidade, em vez de segurança, o que facilita a exploração de vulnerabilidades e a infeção desses dispositivos por *malware*.

No contexto do nosso trabalho, caso um dispositivo malicioso se consiga ligar à rede, este conseguiria injetar *malware* pela rede atacando assim os

dispositivos IOT.

Para mitigar esse problema o sistema domiciliar emprega uma rede independente à principal, de forma a segregar os dispositivos IOT dos restantes. De forma similar o *sysadmin* responsável pela montagem do sistema domiciliar deverá certificar-se que esta nova rede de dispositivos não é facilmente acedida por atacantes.

5.2.3 *Denial-of-Service* (DOS)

Os ataques de Negação de Serviço (DOS) são uma forma de ataque cibernético em que o objetivo é sobrecarregar um sistema alvo, tornando-o inacessível para utilizadores legítimos. Isso é alcançado por meio do envio de um grande volume de tráfego de rede, solicitações ou recursos para o sistema alvo, levando-o a ficar sobrecarregado e incapaz de responder adequadamente.

Existem várias técnicas e variantes de ataques DOS, incluindo *Distributed Denial-of-Service* (DDOS), onde múltiplos dispositivos são coordenados para realizar o ataque simultaneamente. Esses ataques podem ter consequências significativas, como interrupção de serviços, indisponibilidade de recursos e prejuízos financeiros para as organizações afetadas.

No contexto do nosso sistema, um ataque DOS poderia comprometer a disponibilidade dos dispositivos IOT, tornando-os inutilizáveis ou prejudicando a sua funcionalidade normal. Isso teria um impacto negativo na experiência do utilizador e poderia causar falhas na operação do sistema domiciliar.

Para mitigar esse tipo de ataque, o sistema domiciliar adota medidas de proteção, como implementação de firewalls. Além disso, é importante manter o sistema atualizado com *patches* de segurança e adotar práticas de segurança recomendadas para prevenir ou minimizar os efeitos de ataques DOS. No lado do servidor, este foi implementado num serviço cujos servidores têm uma resiliência elevada, de forma a garantir a disponibilidade do sistema mesmo após ataques deste teor.

5.2.4 *Zero Day Exploits*

Zero Day Exploits são vulnerabilidades de segurança desconhecidas e não corrigidas que são exploradas por atacantes antes que os desenvolvedores tenham a oportunidade de corrigi-las. Essas vulnerabilidades não são divulgadas publicamente e, portanto, não há uma solução oficial disponível para proteger os sistemas vulneráveis.

No contexto do nosso sistema, se um *Zero Day Exploit* for explorado com sucesso, poderá comprometer a segurança dos dispositivos IOT e do sistema

como um todo. Os atacantes podem explorar essa vulnerabilidade para obter acesso não autorizado aos dispositivos, roubar informações confidenciais, executar comandos maliciosos ou realizar outras atividades prejudiciais.

Para mitigar esse risco, é importante testar de uma forma bem minuciosa o sistema contra vulnerabilidades comuns a dispositivos deste tipo para garantir que problemas como estes não existem. Além disso, é fundamental monitorar as informações de segurança e estar ciente de quaisquer vulnerabilidades conhecidas ou emergentes. Desta forma é possível aplicar medidas preventivas, como a implementação de soluções alternativas temporárias ou a aplicação de medidas adicionais de segurança para proteger os dispositivos e o sistema contra-ataques baseados em *Zero Day Exploits*.

Capítulo 6

Metodologia e Implementação

6.1 Introdução

O presente capítulo aborda os pormenores mais relevantes da implementação do sistema. Na secção 6.2 é feita uma breve demonstração da implementação real do sistema, incluindo os vários sensores e a *gateway*. A secção 6.3 descreve a *interface* acessível ao cliente, por meio de uma *dashboard*. A secção 6.4 apresenta as duas APIs essenciais para este projeto. A secção 6.5 descreve o *chatbot* implementado para a receção de notificações, através do Telegram. A secção 6.6 fala sobre algumas das características interessantes desenvolvidas ao longo deste projeto e que acabam por completá-lo. A secção 6.7 fala sobre alguns dos desafios da implementação tomada pelo grupo. Finalmente, é feita uma conclusão a este capítulo, na secção 6.8.

6.2 Demonstrador do Sistema Físico

O sistema físico é composto por um *ESP32*, dois sensores de temperatura DHT11, dois sensores de proximidade do tipo *hall* e um sensor de gás MQ-7. As figuras 6.1 e 6.2 ilustram o sistema físico, tanto para o *ESP32*, como para o *gateway*. Uma vez que o *ESP32* está dotado nativamente de capacidades *wireless*, a comunicação com o *gateway* é também feita por *wireless*.

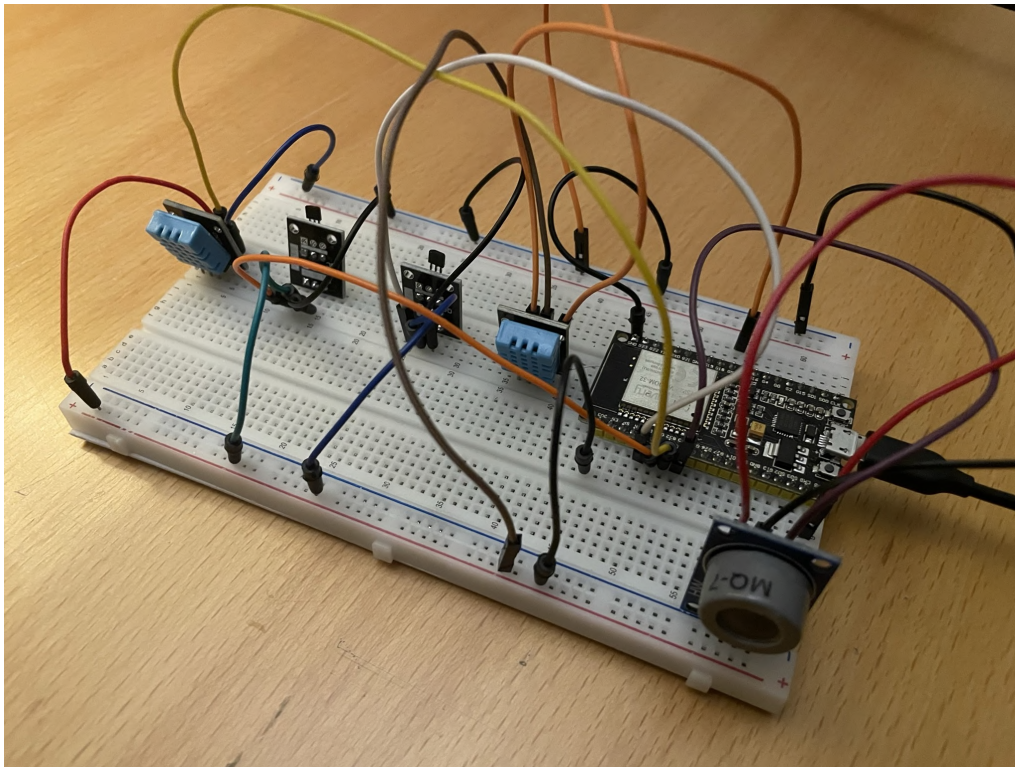


Figura 6.1: Representação física dos vários sensores interligados ao ESP32.



Figura 6.2: Representação física do *gateway*.

6.3 Cliente

O sistema de cliente consiste numa *interface* gráfica (*dashboard*) que permite aos responsáveis aceder e visualizar o estado atual dos sensores instalados em cada divisão da casa. Através dessa *interface*, é possível obter informações em tempo real das diferentes áreas do domicílio. Além do *dashboard*, o sistema de cliente também oferece a funcionalidade de um *chatbot*, disponível no Telegram. O *chatbot* opera de forma semelhante à *dashboard*, permitindo que os responsáveis visualizem o estado atual dos sensores e recebam notificações acerca de eventos alarmantes. Esta secção irá detalhar cada uma destas implementações.

6.3.1 Design Gráfico

Ao implementar um *dashboard* de *design* gráfico, a escolha do *Bootstrap* como *framework* de desenvolvimento ajuda a criar uma aparência simples e *clean*. O estilo minimalista, tipografia legível e sistema de *grid* responsivo contribui para um *design* eficiente e visualmente atraente. A utilização dos componentes pré-estilizados também agiliza o processo de desenvolvimento, resultando num *dashboard* profissional e funcional.

Nas figuras 6.3 e 6.4 é possível apreciar a elegância das páginas de *login* e registo, respetivamente, que foram cuidadosamente criadas utilizando o *framework Bootstrap*. As cores selecionadas foram meticulosamente escolhidas para criar um *Look and feel* mais acolhedor, proporcionando uma sensação envolvente e agradável aos responsáveis. Com esta abordagem, adiciona-se um toque de sofisticação, transformando a experiência de *login* e registo em algo verdadeiramente encantador e memorável.

Nas figuras 6.5, 6.6 e 6.7, é possível observar com mais detalhes a utilização sofisticada do *Bootstrap*. Com a ajuda da biblioteca *Charts*, é possível ilustrar um gráfico de barras exibindo o histórico de eventos alarmantes, conforme apresentado na figura 6.5. Na figura 6.6, podemos apreciar a implementação elegante dos ícones *Font Awesome* para criar cartões, oferecendo ao utilizador uma experiência fluida e intuitiva ao apresentar a disposição da informação dos sensores. Por fim, na figura 6.7, temos a visualização de uma tabela de *logs* contendo todos os dados recebidos e considerados alarmantes.

Essas implementações impressionantes não apenas agregam valor estético ao *dashboard*, mas também fornecem uma experiência agradável e funcional aos utilizadores, garantindo uma fácil interpretação dos dados e ações rápidas em resposta a eventos importantes.

Na figura 6.8, dedicada à *dashboard* do administrador, pode-se apreciar elementos de destaque que conferem um toque elegante e funcional ao *design*. Consegue-se observar a presença de um botão distintivo, permitindo ao administrador a capacidade de permitir ou bloquear novos registos de utilizadores. Essa funcionalidade fornece um controlo adicional sobre o acesso ao sistema, proporcionando maior segurança e autoridade.

Além disso, uma tabela interativa é exibida, permitindo ao administrador modificar os valores considerados alarme. Esta funcionalidade avançada permite ajustar facilmente as configurações para melhor atender às necessidades e requisitos específicos do ambiente.

Por fim, é possível encontrar uma tabela de *logs*, na qual cada *login* efetuado no *dashboard* é registado. Essa tabela oferece um histórico completo e detalhado dos acessos realizados, fornecendo uma visão geral das atividades realizadas pelos utilizadores.

Esses elementos cuidadosamente integrados na figura 6.8 não só proporcionam uma experiência visualmente apelativa, mas também fornecem ao administrador as ferramentas necessárias para gerir eficientemente o sistema, monitorizar eventos importantes e tomar decisões informadas.

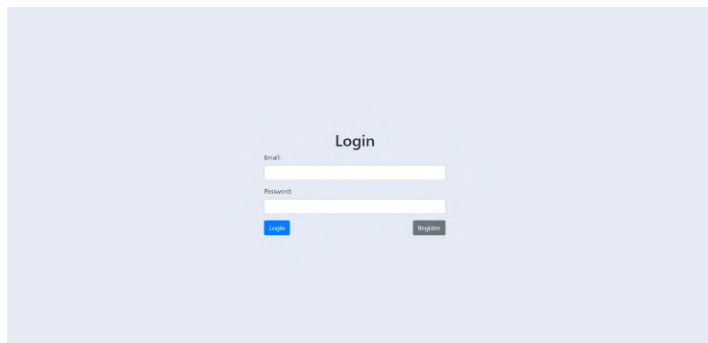
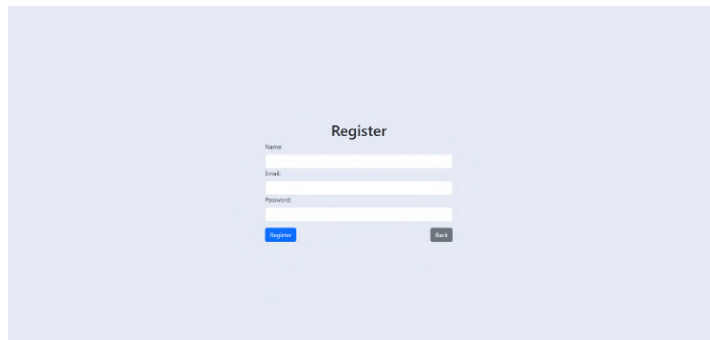
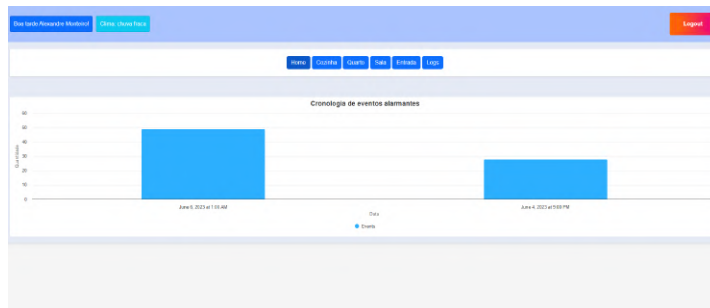
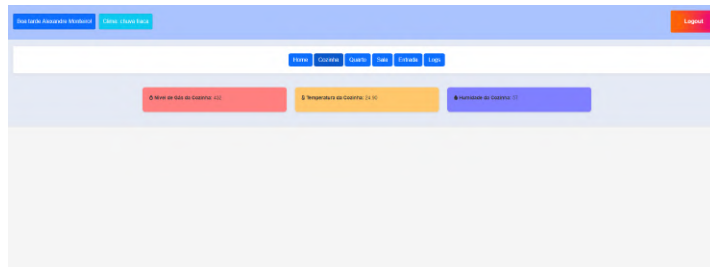


Figura 6.3: Página *login* do *dashboard*.

6.3.2 Autenticação

A autenticação num *dashboard* é um processo essencial para garantir a segurança e controlar o acesso ao sistema, verificando assim a identidade do utilizador que está a tentar entrar, permitindo apenas o acesso a utilizadores autorizados.

- Página de registo
 - Esta página permite que novos utilizadores se registem no sistema. Nesta etapa, é importante recolher informações necessárias

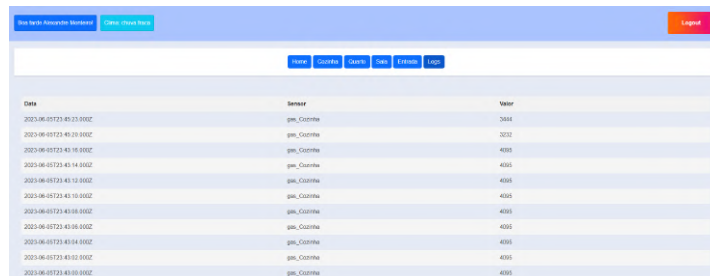
A screenshot of a web form titled "Register". It contains three input fields labeled "Name", "Email", and "Password". Below the "Password" field is a "Register" button in blue and a "Back" button in grey.Figura 6.4: Página *registo* do *dashboard*.Figura 6.5: Página principal do responsável, no *dashboard*.Figura 6.6: Página principal do responsável, no *dashboard*.

para criar uma conta, como nome de utilizador, senha e endereço de *e-mail*.

- A página de registo só é acessível se o administrador permitir, o que ajuda a controlar o acesso ao sistema.

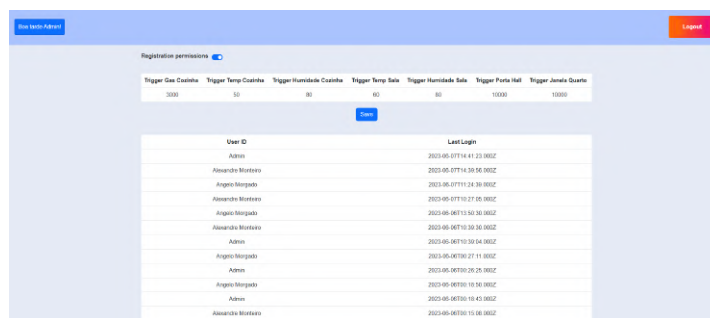
- Criptografia de senhas

- Ao receber a senha fornecida pelo utilizador no registo, é importante cifrá-la antes de armazená-la na base de dados. O *bcrypt* é uma biblioteca comumente usada para cifrar senhas de forma



Data	Sensor	Valor
2023-06-01T23:40:23.000Z	gas_Co2inha	3884
2023-06-01T23:40:29.000Z	gas_Co2inha	3232
2023-06-01T23:43:16.000Z	gas_Co2inha	4393
2023-06-01T23:43:14.000Z	gas_Co2inha	4395
2023-06-01T23:43:12.000Z	gas_Co2inha	4395
2023-06-01T23:43:10.000Z	gas_Co2inha	4395
2023-06-01T23:43:08.000Z	gas_Co2inha	4395
2023-06-01T23:43:06.000Z	gas_Co2inha	4395
2023-06-01T23:43:04.000Z	gas_Co2inha	4395
2023-06-01T23:43:02.000Z	gas_Co2inha	4395
2023-06-01T23:43:00.000Z	gas_Co2inha	4395

Figura 6.7: Página principal do responsável, no *dashboard*.



Trigger Gas Co2inha	Trigger Temp Co2inha	Trigger Humidade Co2inha	Trigger Temp Sala	Trigger Humidade Sala	Trigger Porta Hall	Trigger Janela Quarto
3000	50	80	60	80	10000	10000

User ID	Last Login
Admin	2023-05-07T14:41:23.000Z
Alexandre Monteiro	2023-05-07T14:26:56.000Z
Angelo Morgado	2023-06-01T19:26:30.000Z
Alexandre Monteiro	2023-05-07T10:27:09.000Z
Angelo Morgado	2023-06-01T10:30:30.000Z
Alexandre Monteiro	2023-05-06T10:30:30.000Z
Admin	2023-05-06T10:30:04.000Z
Angelo Morgado	2023-05-06T09:27:11.000Z
Admin	2023-05-06T09:26:26.000Z
Angelo Morgado	2023-05-06T09:10:43.000Z
Admin	2023-05-06T09:10:43.000Z
Alexandre Monteiro	2023-05-06T09:10:46.000Z

Figura 6.8: Página principal do administrador, no *dashboard*.

segura. Ele aplica uma função de *hash* unidirecional à senha, tornando-a ilegível e garantindo que a senha original não possa ser recuperada.

- Página de *login*
 - Após o registo, os utilizadores podem entrar no sistema utilizando a página de *login*. Nessa página, eles fornecem as suas credenciais de acesso, como o endereço de *e-mail* e senha. O sistema verifica as informações inseridas com as armazenadas na base de dados. Para comparar as senhas, o *bcrypt* é usado para cifrar a senha recebida no *login* e compará-la com a senha armazenada na base de dados. Se as informações correspondem, o utilizador é autenticado com sucesso e recebe acesso ao *dashboard*.

6.3.3 Dashboard Utilizador

Este *dashboard* é dedicado à visualização dos dados recebidos pelos vários sensores em tempo real. Esta *dashboard* é utilizada pelos cuidadores formais e não formais.

- Acesso restrito
 - O *dashboard* é acessível apenas aos cuidadores do(s) idoso(s).
- Histórico de eventos alarmantes
 - Os responsáveis podem visualizar um gráfico que exhibe o histórico de eventos alarmantes registados pelo sistema, permitindo-lhes acompanhar ocorrências anteriores e avaliar a situação geral.
- Valores atualizados dos sensores
 - Os responsáveis têm acesso aos valores atualizados dos sensores instalados em cada divisão da casa. Desta forma, fornecem-se informações em tempo real sobre o *status* e as condições de cada divisão.
- Clima atual nas coordenadas da casa do idoso
 - Além das informações dos sensores internos, os cuidadores podem visualizar o clima atual nas coordenadas geográficas da casa do idoso. Tal funcionalidade pode incluir dados como temperatura, humidade, condições climáticas gerais, entre outros.

6.3.4 *Dashboard* Administrador

Este *dashboard* é focado ao controlo do sistema por parte do administrador, como tal, este *dashboard* não visa observar as condições do domicílio, mas sim gerir as definições do mesmo, garantindo assim, mais controlo e segurança.

- Acesso restrito
 - A página do administrador é acessível apenas aos administradores do sistema.
- Alteração de valores alarmantes dos sensores
 - Os administradores conseguem alterar os valores considerados alarmantes para os sensores. Isso permite que eles ajustem os limiares para ativação de notificações e alertas.
- Visualização de *logins* dos utilizadores

- O administrador tem acesso a informações sobre os *logins* realizados pelos utilizadores. Isso pode incluir detalhes como a data e a hora do *login*, o nome de utilizador e outras informações relevantes para fins de auditoria e segurança.
- Permissão de registo de novos utilizadores
 - O administrador tem a opção de permitir ou bloquear o registo de novos utilizadores. Esta funcionalidade oferece um controlo adicional sobre quem pode criar contas no sistema, assegurando que apenas utilizadores autorizados possam registar-se e aceder às funcionalidades disponíveis.

6.3.5 *Logs* do sistema

Os *logs* do sistema são registos importantes que fornecem informações detalhadas sobre eventos relevantes que ocorrem no sistema. No projeto, existem dois tipos principais de *logs*:

- Logs de eventos alarmantes
 - Propósito: Os *logs* de eventos alarmantes registam todas as ocorrências consideradas alarmantes pelo sistema, como níveis críticos de temperatura, níveis elevados de gás, entre outros.
 - Conteúdo: Cada entrada no *log* de eventos alarmantes deve incluir informações como a data e hora do evento e o nome do sensor envolvido no evento alarmante.
 - Notificações: Quando ocorre um evento alarmante, os responsáveis pela casa do idoso recebem uma notificação. Essa notificação pode ser enviada por meio de *e-mail* e via *chatbot* de Telegram, caso o responsável possua a aplicação e esteja registado na mesma.
- Logs de *login* dos utilizadores
 - Propósito: Os *logs* de *login* dos utilizadores registam as informações relacionadas aos acessos realizados pelos utilizadores no sistema, sejam eles administradores ou cuidadores.
 - Conteúdo: Cada entrada no *log* de *login* deve incluir o nome de utilizador, a data e hora do *login*.

- Auditoria e segurança: Esses *logs* são importantes para fins de auditoria e segurança. Eles permitem visualizar quem acessou o sistema e quando, ajudando a identificar atividades suspeitas ou não autorizadas.

Os *logs* do sistema são fundamentais para regular e registrar eventos importantes no sistema. Eles auxiliam na análise de incidentes, na monitorização da segurança e no acompanhamento do uso do sistema pelos utilizadores.

6.4 APIs

Nesta secção, serão apresentadas duas APIs essenciais utilizadas no projeto:

- REST para enviar dados do *gateway* para o servidor Node.js:
 - Uma REST foi implementada para enviar dados dos sensores do sistema domiciliário para o servidor Node.js. Essa API utiliza o método POST para receber os dados, enviados em formato *JSON* no corpo da solicitação. Essa abordagem permite que os dados dos sensores sejam enviados de forma estruturada e segura para o servidor, onde serão processados e armazenados na base de dados. Desta forma, é possível obter informações atualizadas e precisas sobre as condições do domicílio e tomar medidas adequadas com base nos dados. Este tópico é abordado de forma mais aprofundada na subsecção 6.4.1.
- API em *Python* para criar um *chatbot* no *Telegram*:
 - Foi desenvolvida uma API em *Python* para criar um *chatbot* no *Telegram*. Esse *chatbot* visa fornecer uma *interface* interativa para os responsáveis pelo idoso, permitindo-lhes visualizar os dados atuais dos sensores e receber notificações em casos alarmantes. O *chatbot* utiliza a API do *Telegram* para estabelecer uma comunicação bidirecional entre os responsáveis e o sistema domiciliário. Por meio de comandos específicos, os responsáveis podem solicitar informações sobre os sensores, como valores atuais e histórico de eventos alarmantes. Além disso, o *chatbot* consegue enviar notificações aos responsáveis em tempo real, alertando sobre situações críticas detetadas pelos sensores. Este tópico é abordado de forma mais aprofundada na secção 6.5.

6.4.1 Transmissão de Dados Entre o *Gateway* e o Servidor – REST API

O *gateway* atua como um ponto central de conexão que recolhe informações provenientes de sensores e dispositivos domésticos, enquanto o servidor é responsável por processar e armazenar esses dados.

Nesse contexto, a adoção de um REST API emerge como uma solução viável. O REST é um estilo arquitetural que define um conjunto de princípios para projetar serviços *web* escaláveis e interoperáveis. O REST API baseia-se nesses princípios e oferece uma estrutura consistente para a comunicação entre o *gateway* e o servidor.

Uma das principais razões para utilizar uma REST API é a sua natureza orientada a recursos. Essa abordagem facilita a identificação e manipulação dos dados, permitindo que sejam criadas solicitações específicas para obter, criar, atualizar ou excluir informações. A implementação de uma API bem estruturada para receber dados cifrados e decifrá-los antes de inseri-los na base de dados adiciona uma camada extra de segurança aos dados captados pelos sensores.

A implementação do REST API no nosso projeto teve duas etapas, a implementação no lado do *gateway* (subsubsecção 6.4.1.1) e a implementação no lado do servidor (subsubsecção 6.4.1.2).

6.4.1.1 *Gateway*

A implementação no lado do *gateway* passou pela instalação do Mosquitto MQTT e posterior desenvolvimento de um pequeno código em *Python* para enviar os dados recebidos através de um JSON, cifrá-los e enviá-los para a REST API. Desta forma, garante-se a integridade dos dados, uma vez que a cifra utilizada é de qualidade e qualquer tentativa de modificação dos dados durante o seu transporte gerará um texto-limpo ilegível. O anexo A.2 demonstra todo o código utilizado para este propósito. É de referir duas importantes variáveis: `receiver_url` e `mqtt.broker_ip` - a primeira indica o uniform resource locator (URL) que disponibiliza a API de receção de dados, através de um JSON cifrado, e a segunda o endereço internet protocol (IP) do próprio *gateway*.

6.4.1.2 Servidor

O servidor dá *expose* de um método **POST**, de forma a escutar possíveis chamadas provenientes do *gateway*. Na figura 6.9 é possível visualizar, passo a passo, o fluxo do seu processo constituído por:

- Rota *"/api/receivedata"*
 - Esta rota é acedida mediante um pedido *POST* e recebe o *JSON* cifrado no *req.body*, contendo os dados captados pelos sensores e o *Initialization Vector* necessário para a decifra da mensagem;
 - O *JSON* cifrado é recebido como entrada e será decifrado usando o algoritmo *AES-128-CBC*.
- Decifra dos dados
 - Uma função específica de decifra *AES-128-CBC* é utilizada para decifrar o *JSON* recebido.
 - A função de decifra requer uma chave de decifra que foi previamente acordada de forma segura (e.g., numa conversa de café), sendo utilizada para decifrar os dados.
- Valores captados pelos sensores
 - Após a decifra, os valores captados pelos sensores são extraídos do *JSON* decifrado;
 - Esses valores são agora legíveis e podem ser processados e inseridos na base de dados.
- Inserção na base de dados
 - Os valores captados pelos sensores são inseridos na base de dados, numa tabela específica para armazenar essas informações (i.e., a tabela `sensor_data`);
 - É importante garantir que os dados sejam corretamente validados e *sanitized* antes da inserção na base de dados para evitar possíveis vulnerabilidades, tais como injeção de código.
- Verificação e notificação de limites ultrapassados
 - Após a inserção dos valores captados na base de dados, é realizada uma verificação para determinar se algum desses valores ultrapassa os limites definidos pelo sistema;
 - Se algum valor exceder esses limites, é acionado um mecanismo de notificação para alertar os responsáveis, informando sobre o evento e fornecendo os detalhes relevantes do mesmo.



```
1 app.post('/api/receivedata', async (req, res) => {
2   /*
3    JSON:
4    {
5      "iv": "xx",
6      "cipher_text": "xx"
7    }
8   */
9   // Access the request body data
10  const requestBody = req.body;
11
12  // Decrypt the data
13  try {
14    let decryptedPayload = await decrypt(requestBody.iv, requestBody.cipher_text);
15    decryptedPayload = decryptedPayload.replace(/'/g, '');
16    console.log(decryptedPayload);
17    // Convert the decrypted payload back to JSON
18    const sensorData = JSON.parse(decryptedPayload);
19    console.log(sensorData);
20    // Save the sensor data to the database
21    const status = await insertSensorDataToDB(sensorData);
22    res.status(201).json({ message: 'Status:201' });
23  } catch (error) {
24    console.log(error);
25    res.status(500).json({ message: 'An error occurred.', error });
26  }
27  });
```

Figura 6.9: Código implementado no servidor *NodeJS* para o tratamento dos dados capturados pelos sensores.

O fluxo demonstra uma boa prática de segurança, onde os dados são cifrados durante a transmissão e decifrados somente no momento de processamento e armazenamento. Isso ajuda a proteger os dados confidenciais dos utilizadores e garante que apenas as pessoas autorizadas possam visualizá-los. Além disso, a verificação e notificação de limites ultrapassados contribuem para a segurança e vigilância efetiva do sistema.

6.5 Chatbot do Telegram

Uma forma extremamente interessante e flexível de aceder aos dados do domicílio e receber notificações acerca de eventos alarmantes do mesmo é ao utilizar um *chat* interativo que permite ao utilizador interagir com o sistema sem ter que acessar o *dashboard*. Para isso foi desenvolvido um *chatbot* na plataforma Telegram com esse mesmo intuito. Este *chatbot* não

contém tantas características como o *dashboard*, porém este apresenta-se como uma forma fácil e rápida de verificar o domicílio do idoso.

6.5.1 Configuração

Este *chatbot* foi desenvolvido em *Python* utilizando a biblioteca `pyTelegramBotAPI`. Esta permite criar uma função para cada comando que se deseja criar. De forma a simplificar o processo de comunicação com o servidor, o *chatbot* foi colocado a correr na mesma máquina DigitalOcean que o servidor e a base de dados, podendo assim poder fazer chamadas para o *localhost*. O *chatbot* tem que estar sempre a correr em *background*, e o tempo de *deployment* é bastante curto, aumentando assim a sua testabilidade e disponibilidade.

6.5.2 Guia de utilização

Para ajudar o utilizador a utilizar o *chatbot* foram implementadas mensagens de ajuda que podem ser acedidas com o comando `/help`¹. O resultado desta *prompt* irá depender se o utilizador tem o *login* efetuado ou não. Se não estiver aparece a mensagem que está presente na figura 6.10, e quando se tem aparece a que está na figura 6.11. Assim, o utilizador sempre saberá quais os comandos disponíveis na sua presente situação.

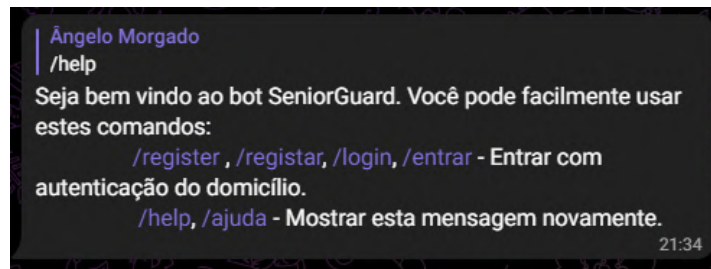


Figura 6.10: *Prompt* após ser introduzido o comando `/help` quando não se tem o *login* efetuado.

6.5.3 Autenticação

Como os *chatbots* do Telegram são públicos e globais, foi preciso implementar uma forma de fazer com que apenas pessoas do domicílio, ou responsáveis por monitorizar o domicílio, tenham acesso às informações do

¹Os comandos `/ajuda` e `/start` funcionam de forma equivalente.

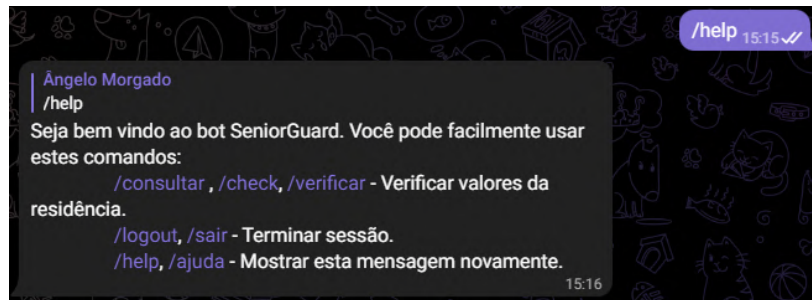


Figura 6.11: *Prompt* após ser introduzido o comando `/help` quando se tem o *login* efetuado.

mesmo. Para isso foi criado um comando `/login`² que pede à pessoa no *chat* pelo *email* e *password* do registo que fizeram no *dashboard*, sendo assim uma conta sincronizada entre os dois sistemas.

Para sair da conta é necessário utilizar o comando `/logout` ou `/sair`.

Quando é feita uma autenticação, é guardado o *chat ID* da conversa numa *hash table* para manter os *logins* locais e não globais, e o *chat ID* é também colocado na base de dados associado à conta de forma a notificar o utilizador em caso de evento alarmante.

6.5.4 Visualização do Último Evento Alarmante

O utilizador pode requerer saber quando é que foi a última vez que houve um evento alarmante, e saber qual o sensor que despoletou o alarme e qual o valor do mesmo. Para fazer tal coisa primeiro este tem que inserir o comando `/check`³, onde lhe vai aparecer dois botões do género ao que está na figura 6.12. Após isso o utilizador deve clicar no botão "Consultar último evento alarmante" e, ser-lhe-á apresentada uma mensagem do tipo da figura

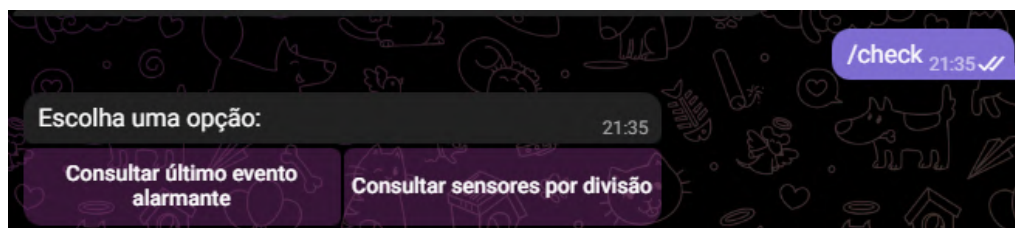


Figura 6.12: *Prompt* após ser introduzido o comando `/check`.

²Os comandos `/registar`, `/entrar` e `/register` funcionam de forma equivalente.

³Os comandos `/verificar` e `/consultar` funcionam de forma equivalente.

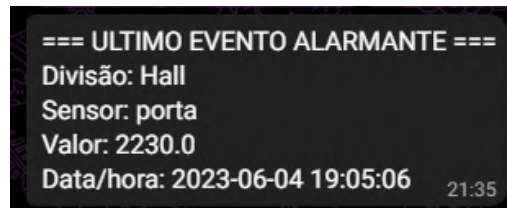


Figura 6.13: Mensagem que detalha o último evento alarmante.

6.5.5 Visualização das Informações das Divisões

Além de consultar o último evento alarmante, o utilizador poderá também consultar as informações dos sensores de uma dada divisão do domicílio. Para fazer isso o utilizador deverá mandar o comando `/check` ou algum comando equivalente, como mostrado na figura 6.12, e depois deverá clicar no botão "Consultar sensores por divisão". Após isso, ser-lhe-á apresentado outro menu com botões em que cada botão representa uma divisão da casa, tal como mostrado na figura 6.14.

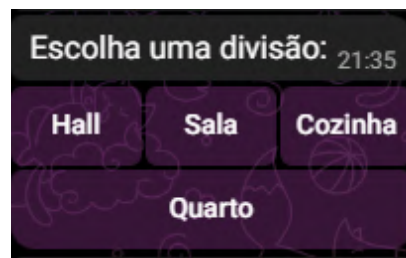


Figura 6.14: Mensagem que mostra os botões com as diversas opções.

O resultado da escolha de cada divisão pode ser observado na figura 6.15, em que para cada divisão mostra os sensores e o valor destes, assim como a data de atualização dos mesmos.

6.5.6 Notificação de eventos alarmantes

Além de consulta, este *chatbot* também tem a importante função de notificar o utilizador sempre que algum evento alarmante é despoletado, complementando assim o sistema de *email*. Para notificar os utilizadores, são utilizados os *chat IDs* na base de dados. É feita uma interrogação à base de dados que vai buscar todos os utilizadores cujo *chat ID* não seja nulo e manda mensagem para estes a alarmar. Como o mesmo utilizador pode entrar em vários dispositivos diferentes, e a base de dados só guarda um *chat ID*,

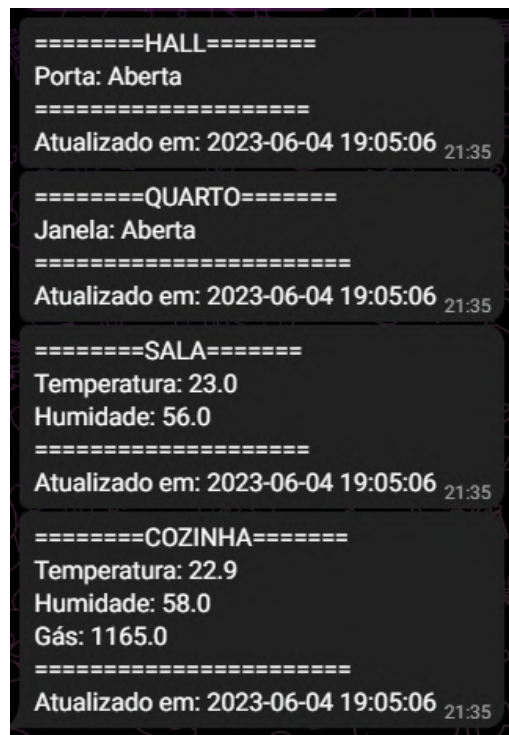


Figura 6.15: Mensagem que mostra os botões com as diversas opções.

apenas o último dispositivo que fez *login* na conta é notificado acerca do evento alarmante.

A mensagem que aparece como alarme contém todas as informações relativas ao evento assim como um conselho de contacto a entidades de segurança caso o problema seja mais grave, tal como mostrado na figura 6.16.

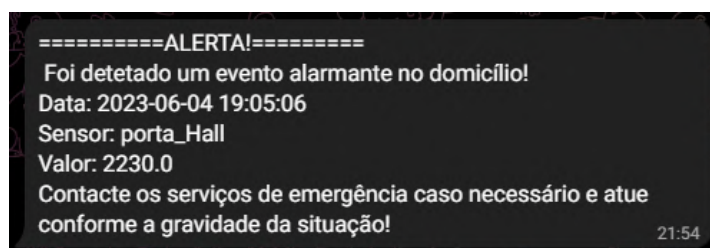


Figura 6.16: Mensagem de evento alarmante.

Para receber a mensagem de alarme foi configurado uma REST API à parte da principal com o único propósito de receber pedidos, e, assim, acionar o processo de notificar os utilizadores, foi utilizada a biblioteca Flask para o efeito. Desta forma foi necessário implementar *threads*, visto que devido à

tamanha concorrência entre a biblioteca do Telegram e o Flask, um sempre bloqueava o outro, pelo que têm que correr em processos diferentes. Como uma conexão MySQL não funciona bem entre *threads* diferentes, foi necessário criar duas conexões diferentes, uma para os pedidos de consulta (*thread* principal), e outra para as notificações(*thread* paralela).

6.6 Características Interessantes

6.6.1 Criptografia

Na hora de escolher uma biblioteca de criptografia para proteger as senhas dos utilizadores e também cifrar a comunicação entre o *gateway* e o servidor, optou-se pelo *bcrypt* por ser uma opção conhecida e confiável. Ele é amplamente utilizado na comunidade de desenvolvimento e oferece uma segurança robusta. O *bcrypt* é uma escolha sólida porque:

- É resistente a ataques de força bruta e tentativas de quebra de senha.
- Implementa automaticamente a técnica de *salting*, que adiciona uma camada extra de segurança às senhas.
- Permite ajustar a velocidade do processo de criptografia, dificultando ataques de força bruta.
- É suportado por várias linguagens de programação, incluindo o Node.js, facilitando a sua implementação no nosso projeto.
- É bastante simples de utilizar e contém uma vasta documentação a respeito do mesmo.

6.6.2 Estatísticas

Uma maneira eficaz de visualizar o histórico de eventos alarmantes é por meio de um gráfico visual que apresente essas informações ao longo do tempo, segmentadas por hora. Esse gráfico permitirá que os responsáveis pelo idoso tenham uma visão clara dos momentos em que ocorreram eventos alarmantes e possam identificar padrões ou tendências.

A implementação desse gráfico de histórico de eventos alarmantes por hora pode seguir algumas diretrizes:

- Eixo X: O eixo X do gráfico representa o tempo, geralmente em intervalos de horas. Ele mostra o período em que os eventos alarmantes

foram registados. Cada barra no gráfico corresponderá a uma determinada hora.

- Eixo Y: O eixo Y representa a quantidade de eventos alarmantes registados em cada hora. Ele mostra o número de ocorrências de eventos alarmantes ao longo do tempo. Essa dimensão vertical do gráfico fornecerá uma medida da gravidade ou intensidade dos eventos em cada período.

Na figura 6.17 é possível visualizar uma implementação dos dados considerados alarmantes, num gráfico histórico de barras.

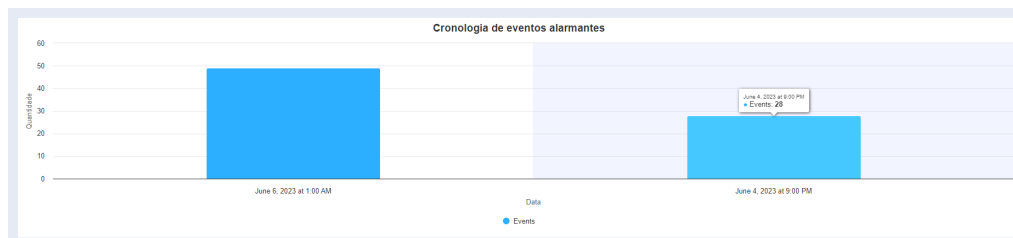


Figura 6.17: Gráfico de eventos alarmantes, com base nos dados recebidos através do uso dos sensores.

6.6.3 Notificações

Em situações de emergência, em que algum sensor deteta um valor invulgar, é necessário notificar os utilizadores a respeito do mesmo, para que possam agir prontamente e com eficiência. De forma a garantir que a mensagem chega aos cuidadores do idoso, são usados dois métodos de notificação.

Primeiramente é utilizado o *chatbot* do Telegram. Como explicado na subsecção 6.5.6, todos os utilizadores que se tenham registado no *chatbot* serão notificados prontamente com uma mensagem com um formato parecido ao mostrado na figura 6.16. Este representa o método principal de notificação visto que o telemóvel é o principal meio de comunicação mundialmente.

O segundo método de notificação é através de *email*. Este método também foi escolhido porque nem todas as pessoas estão perto do telemóvel e podem estar ao computador com a aba de *email* aberta.

Ao identificar uma situação alarmante, o sistema envia automaticamente um *e-mail* para todos os responsáveis pelo idoso (i.e., todas as contas registadas no *dashboard*, exceto o administrador), fornecendo uma breve explicação sobre o evento identificado e instruções sobre as medidas a serem tomadas, como está ilustrado na figura 6.18. Esta abordagem permite uma

resposta rápida e eficaz mediante situações críticas, possibilitando que os cuidadores ajam prontamente para garantir a segurança do idoso. O sistema de detecção e notificação contribui significativamente para a tranquilidade dos responsáveis, proporcionando uma forma proativa de vigilância e cuidado, minimizando potenciais riscos e prevenindo consequências adversas.

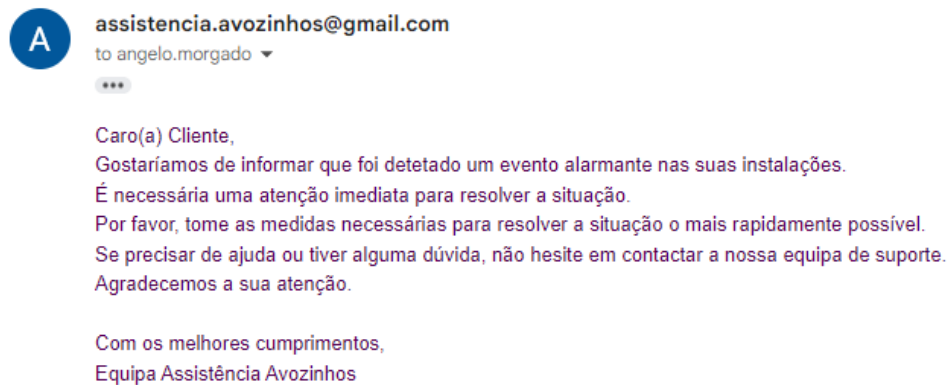


Figura 6.18: *E-mail* exemplo enviado a todos os responsáveis, no caso de um evento alarmante.

6.6.4 Implementação de um Domínio

A implementação de um domínio para acesso à *dashboard* foi de extrema importância, uma vez que é a partir da internet que é possível aceder e consultar os dados recolhidos pelos vários sensores, num único lugar. O domínio implementado foi o `avozinhoiot.tech`, adquirido de forma gratuita através do GitHub Student Pack, com a duração de um ano. Para além disso, foi necessário alterar os *name servers* no *registrar* para os *name servers* da DigitalOcean, onde se encontra hospedada a *dashboard*. É também importante referir que foi necessário obter um certificado SSL, tal como explica a subsecção 6.6.5.

6.6.5 Implementação de SSL

O código ilustrado na figura 6.19 implementa um *middleware* que redireciona o tráfego HTTP para HTTPS e configura um servidor HTTPS seguro. Desta forma, garante-se a segurança da comunicação entre o cliente e o servidor, utilizando criptografia para proteger os dados transmitidos. Para tal foi preciso implementar:

- *Middleware* de redirecionamento:
 - O trecho de código *app.use* define um *middleware* que será executado antes de todas as outras rotas. Ele verifica se a solicitação está sendo feita usando o protocolo HTTP.
 - Se a solicitação for efetuada por HTTP, o código cria um URL de redirecionamento para HTTPS e redireciona o cliente para esse URL: `https://$req.headers.host$req.url`, usando uma resposta do tipo HTTP 301.
- Configuração do servidor HTTPS:
 - O objeto *options* é criado com as opções necessárias para configurar o servidor HTTPS. Ele contém os caminhos para os arquivos de chave privada (`privkey.pem`) e certificado (`fullchain.pem`). Estas chaves constituem o certificado secure sockets layer (SSL), obtido através do *Let's Encrypt*, uma autoridade de certificação gratuita e sem custos para o utilizador.
 - O método *https.createServer* é chamado com as opções e o objeto *app* para criar um servidor HTTPS seguro.
 - O método *listen* é usado para iniciar o servidor HTTPS na porta 443, que é a porta padrão para conexões HTTPS.
- Redirecionamento do servidor HTTP:
 - O método *http.createServer* é usado para criar um servidor HTTP separado.
 - Quando uma solicitação é feita ao servidor HTTP, o código gera um URL de redirecionamento para HTTPS usando a mesma lógica do *middleware*.
 - Em seguida, uma resposta HTTP 301 é enviada ao cliente com um cabeçalho *Location* definido para a URL de redirecionamento.
 - O servidor HTTP é iniciado na porta 80, que por sua vez é a porta padrão para conexões HTTP.

Desta forma foi possível implementar uma ligação segura SSL e, ao mesmo tempo, redirecionar pedidos HTTP para um URL mais seguro HTTPS.

6.7 Desafios de Implementação

6.7.1 Configuração do *gateway* como um *hotspot*

Inicialmente, foi proposto a criação de um *hotspot*, a partir do *gateway*, e posteriormente ligar este à internet através de uma ligação com fios. No entanto, esta tarefa mostrou-se ser bastante árdua, nomeadamente, na implementação do *routing* entre a rede *wireless* e a porta Ethernet. Concluiu-se assim que tal não seria necessário, acabando por ser criada uma rede *wireless* tradicional, mas dedicada aos dispositivos IOT, resultando assim no mesmo nível de segurança ou talvez mais, e eliminando uma dependência entre os dispositivos e o *gateway*.

6.8 Conclusões

Neste capítulo foram apresentados os principais detalhes de implementação sobre o sistema desenvolvido. Desta forma, e com toda a informação enunciada no presente capítulo, é dado a conhecer de forma detalhada as várias partes que constituem o sistema num todo - o sistema físico, com os vários sensores; a *dashboard* e a sua *interface gráfica*; as APIs desenvolvidas; o sistema de notificações através de um *bot* Telegram; algumas características e desafios interessantes, aquando do desenvolvimento do presente trabalho.



```
1  const http = require('http');
2  const https = require('https');
3  const fs = require('fs');
4
5  // Middleware to redirect HTTP to HTTPS
6  app.use((req, res, next) => {
7    if (req.protocol === 'http') {
8      const redirectUrl = `https://${req.headers.host}${req.url}`;
9      console.log(redirectUrl);
10     return res.redirect(301, redirectUrl);
11   }
12   next();
13 });
14
15 const options = {
16   key: fs.readFileSync('privkey.pem'),
17   cert: fs.readFileSync('fullchain.pem'),
18 };
19
20 https
21   .createServer(options, app)
22   .listen(443, () => {
23     console.log("Server is running at port 443 (HTTPS)");
24   });
25
26 http
27   .createServer((req, res) => {
28     const redirectUrl = `https://${req.headers.host}${req.url}`;
29     console.log(redirectUrl);
30     res.writeHead(301, { Location: redirectUrl });
31     res.end();
32   })
33   .listen(80, () => {
34     console.log("Server is running at port 80 (HTTP)");
35   });
```

Figura 6.19: Trecho de código usado para a implementação de HTTPS com redirecionamento de HTTP para HTTPS.

Capítulo 7

Conclusão

7.1 SWOT

Com a finalidade de avaliar o trabalho realizado, foi desenvolvido um estudo de Análise Strengths, Weaknesses, Opportunities, Threats (SWOT) do projeto. Dessa forma, é possível identificar os pontos fortes e fracos do projeto, bem como obter uma compreensão mais assertiva de possíveis melhorias e trabalhos futuros.

7.1.1 Forças

Conforme demonstrado no artigo [1], Portugal é um dos países com uma população envelhecida, o que resulta em muitos idosos que não têm um suporte próximo para verificar regularmente o seu bem-estar. Nesse contexto, a nossa aplicação desempenha um papel crucial, pois auxilia na monitorização dessas pessoas idosas sem exigir a sua presença física, permitindo-lhes manter a sua independência, e não sacrificando a sua privacidade.

Outra força desta aplicação reside na facilidade e rapidez de implementação. O sistema já possui um modelo de *gateway* e servidor (i.e., *dashboard*, base de dados e *chatbot*) pré-desenvolvido, o que simplifica a adaptação do sistema às necessidades de cada utilizador.

7.1.2 Fraquezas

Uma das fraquezas desse sistema é a hospedagem do servidor na nuvem. Embora essa escolha apresente inúmeras vantagens, como facilidade de implementação e custo reduzido, também traz consigo preocupações de segurança e latência que um servidor local, por exemplo, não teria. No entanto, a decisão de hospedagem é uma escolha que deve ser feita pelo cliente e

pelo *sysadmin* durante a fase de planeamento, considerando-se os requisitos específicos do projeto.

Outra fraqueza relevante consiste na dependência do sistema à internet. Se por algum motivo o acesso à internet for impedido, o sistema não consegue fazer a comunicação com o servidor, embora o *gateway* consiga na mesma receber os valores dos sensores através da rede¹.

7.1.3 Oportunidades

A falta de serviços semelhantes num país com tamanha necessidade deles, como Portugal, oferece diversas oportunidades, não apenas do ponto de vista de negócio, mas principalmente para mitigar um problema que se tem tornado cada vez mais preocupante.

7.1.4 Ameaças

Embora não haja ameaças específicas direcionadas a este sistema, é possível associar ameaças comuns a outras aplicações, como problemas de segurança e desempenho, que podem comprometer a disponibilidade do programa. Essas ameaças são especialmente críticas, considerando-se que estamos a lidar com a vida de pessoas idosas.

7.2 Conclusão final

Ao longo deste trabalho, foi apresentado o desenvolvimento de um sistema para monitorização e cuidado de idosos em domicílio. O objetivo principal era criar uma solução que permitisse verificar o bem-estar dos idosos, mantendo a sua independência e privacidade.

Durante a análise da situação atual, foi evidenciado que Portugal enfrenta desafios significativos relacionados ao envelhecimento da população, com muitos idosos vivendo sozinhos e sem uma rede de apoio próxima. A aplicação desenvolvida surge como uma resposta a essa necessidade premente, proporcionando um acompanhamento contínuo e eficiente dessas pessoas vulneráveis.

Ao longo do processo de desenvolvimento, foram discutidas diversas estruturas para o sistema no domicílio do idoso, acabando por se optar por usar

¹Geralmente quando há uma falha na internet, esta não compromete a rede que o *router* de casa disponibiliza, não impedindo assim a troca de pacotes entre os sensores e o *gateway*.

MQTT na comunicação entre os sensores e o *gateway*, devido à sua natureza de *publisher-subscriber*, o que permite uma integração mais fácil de múltiplos sensores. Para a conexão entre o *gateway* e o servidor optou-se por usar um REST API, que procura facilitar a troca de pacotes com métodos POST e GET, e também resolve problemas de interoperabilidade, tal como explicado no capítulo 4.

Para tornar as conexões mais seguras a ataques cibernéticos, foi implementada criptografia em diversas partes do sistema, como a comunicação entre *gateway* e servidor, no armazenamento de *passwords* na base de dados e a implementação de *cookies* com um *hash* associado nas sessões de utilizador.

Uma das funcionalidades principais do projeto é o *dashboard*. Este foi desenvolvido usando a *framework* Express.js e permite aos utilizadores verificar os valores dos sensores do domicílio em tempo real e também permite ao administrador verificar os *logins* e alterar os valores que são considerados alarmantes. O *dashboard* foi desenhado de forma a ser intuitivo e com uma aparência limpa.

Para notificar os utilizadores em eventos alarmantes, foram implementados dois métodos de os notificar. Primeiramente, foi criado um sistema que envia um *email* genérico para avisar que ocorreu um alarme. Embora esta seja uma alternativa mais simples, é útil em algumas situações. O segundo método e mais importante é o *chatbot*. Este não só se mostra um método mais eficiente de notificar os utilizadores, como também mostra todas as informações relativas ao evento alarmante.

Como mencionado, foi desenvolvido um *chatbot* no Telegram que não só notifica os utilizadores de eventos alarmantes, como também permite visualizar os valores de todos os sensores sem ter que ir ao *dashboard*, mostrando-se ser uma opção mais simples e rápida.

Resumindo e concluindo, tropeçando e não caindo, este projeto permitiu criar uma infraestrutura de suporte aos idosos, procurando assim mitigar um problema que se tem alastrado de uma forma preocupante em Portugal.

7.3 Trabalho futuro

Com base nas oportunidades identificadas, existem várias direções que podem ser exploradas como trabalhos futuros para o aprimoramento e expansão do sistema. Alguns possíveis caminhos incluem:

7.3.1 Adaptação a Redes Constrangidas

Considerando que nem todas as áreas têm acesso a redes de internet de alta velocidade e estáveis, uma oportunidade de trabalho futuro é a adaptação do sistema para funcionar de forma eficiente em redes constrangidas. Isso envolveria a otimização dos protocolos de comunicação e o desenvolvimento de estratégias de compressão de dados para reduzir a quantidade de dados transmitidos entre o *gateway* domiciliar e o servidor. Além disso, a implementação de técnicas de *cache* e armazenamento local no *gateway* permitiria que ele continuasse a coletar e armazenar dados mesmo durante períodos de conectividade intermitente. Essas adaptações garantiriam que o sistema possa ser utilizado de forma eficaz, mesmo em áreas com acesso limitado à internet de alta velocidade, ampliando assim a sua aplicabilidade e alcance geográfico.

7.3.2 Aprimoramento da Segurança e Desempenho

Visando garantir a confiabilidade e a segurança do sistema, um trabalho futuro importante seria aprimorar a segurança da infraestrutura, implementando medidas adicionais de proteção dos dados pessoais e sensíveis dos utilizadores. Além disso, seria necessário realizar testes de desempenho e otimizações para garantir que o sistema possa lidar com um número crescente de utilizadores e permaneça disponível e responsivo mesmo em momentos de pico de utilização.

7.3.3 Integração de Tecnologias Emergentes

Explorar a integração de tecnologias emergentes, como inteligência artificial, aprendizado de máquina e internet das coisas, pode trazer benefícios adicionais ao sistema. Por exemplo, algoritmos de *machine learning* podem ser usados para identificar padrões de comportamento e detetar anomalias nos dados coletados pelos sensores, fornecendo alertas adicionais para o cuidado dos idosos. A integração de dispositivos inteligentes, como assistentes virtuais e *wearables*, também pode ampliar as funcionalidades e a interoperabilidade do sistema, implementado assim as funcionalidades adicionais falados no capítulo 2.

Apêndice A

Excertos de Código Relevantes

A.1 Código utilizado no ESP32

O presente anexo dá a conhecer o código-fonte implementado no ESP32, para receção e transmissão dos dados recebidos dos vários sensores.

```
#include <Arduino.h>
#include "DHT.h"
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

#define DHT_SALA_PIN 32
#define DHT_COZINHA_PIN 5
#define HALL_PORTA_ENTRADA 34
#define HALL_JANELA_QUARTO 35
#define GAS_COZINHA_PIN 33

#define DHTTYPE DHT11

// Replace the next variables with your SSID/Password
// combination
const char* ssid = "Rede-Wireless";
const char* password = "SenhaWireless";
const char* mqtt_server = "192.168.1.80";
const int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

DHT dht_cozinha(DHT_COZINHA_PIN, DHTTYPE);
```

```
DHT dht_sala(DHT_SALA_PIN, DHTTYPE);

int hallStateCozinha = 0;
int hallStateSala = 0;
int gasCozinha = 0;

// 1 - Cozinha Temperatura
// 2 - Cozinha Humidade
// 3 - Sala Temperatura
// 4 - Sala Humidade
// 5 - Hall Porta Entrada
// 6 - Hall Janela Quarto
// 7 - Gas Cozinha
float sensorValues[7];

// create a string array to hold sensor names
char sensorNames[7][50] = {"Cozinha Temperatura", "Cozinha
    Humidade", "Sala Temperatura", "Sala Humidade", "Hall Porta
    Entrada", "Hall Janela Quarto", "Gas Cozinha"};

void setup_wifi() {
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.println("Connecting to MQTT...");

        if (client.connect("ESP32Client")) {

            Serial.println("connected");
        }
    }
}
```

```
        } else {

            Serial.print(" failed with state ");
            Serial.print(client.state());
            delay(2000);
            Serial.println();
        }
    }
}

void setup() {
    Serial.begin(9600);
    dht_cozinha.begin();
    dht_sala.begin();
    pinMode(HALL_PORTA_ENTRADA, INPUT);
    pinMode(HALL_JANELA_QUARTO, INPUT);
    pinMode(GAS_COZINHA_PIN, INPUT);

    // MQTT e WiFi
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    delay(2000);

    hallStateCozinha = analogRead(HALL_PORTA_ENTRADA);
    hallStateSala = analogRead(HALL_JANELA_QUARTO);
    gasCozinha = analogRead(GAS_COZINHA_PIN);

    sensorValues[0] = dht_cozinha.readTemperature();
    sensorValues[1] = dht_cozinha.readHumidity();
    sensorValues[2] = dht_sala.readTemperature();
    sensorValues[3] = dht_sala.readHumidity();
    sensorValues[4] = hallStateCozinha;
    sensorValues[5] = hallStateSala;
    sensorValues[6] = gasCozinha;

    // convert the temperature readings to a float with two
    // decimal places

    DynamicJsonDocument doc(400);
    JsonObject JSONencoder = doc.to<JsonObject>();
```

```

JsonObject cozinha = JSONEncoder.createNestedObject("
    Cozinha");
cozinha["temperatura"] = sensorValues[0];
cozinha["humidade"] = sensorValues[1];
cozinha["gas"] = sensorValues[6];

JsonObject sala = JSONEncoder.createNestedObject("Sala");
sala["temperatura"] = sensorValues[2];
sala["humidade"] = sensorValues[3];

JsonObject quarto = JSONEncoder.createNestedObject("Quarto
");
quarto["janela"] = sensorValues[5];

JsonObject hall = JSONEncoder.createNestedObject("Hall");
hall["porta"] = sensorValues[4];

String JSONmessageBuffer;
serializeJson(doc, JSONmessageBuffer);

client.publish("casa/sensores", JSONmessageBuffer.c_str());
client.loop();
}

```

Excerto de Código A.1: Código utilizado no ESP32.

A.2 Código utilizado no Gateway

O presente anexo dá-nos a conhecer o código utilizado no *gateway*, que recebe os vários dados dos sensores e retransmite-os para a REST API.

```

import requests
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import base64
import paho.mqtt.client as mqtt

receiver_url = 'https://avozinhoiot.tech/api/receivedata'
topic = "casa/sensores"
mqtt_broker_ip = "192.168.1.80"

# AES encryption function
def encrypt(plain_text, key):
    cipher = AES.new(key, AES.MODE_CBC)
    cipher_text = cipher.encrypt(pad(plain_text.encode(), AES.
        block_size))
    iv = base64.b64encode(cipher.iv).decode('utf-8')

```

```
    cipher_text = base64.b64encode(cipher_text).decode('utf-8')
    return iv, cipher_text

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(topic)

# This function gets called every time the subscribed topic
# receives a new message and then posts the data to the
# receiver_url
def on_message(client, userdata, msg):
    print("Received a message on topic: " + msg.topic)
    print("Message: " + str(msg.payload))

    # Check if the received message is a JSON object
    try:
        msg.payload = msg.payload.decode('utf-8')
        msg.payload = eval(msg.payload)
        print("Message is valid!")
    except Exception as e:
        print(e)
        print("Message is not valid!")
        return

    # Encrypt the payload using AES-128-CBC
    key = b'BarricoLegend123' # 16-byte (128-bit) encryption
    key = key[:16]
    iv, cipher_text = encrypt(str(msg.payload), key)

    # Create the payload to be sent
    encrypted_payload = {
        'iv': iv,
        'cipher_text': cipher_text
    }

    response = requests.post(receiver_url, json=
        encrypted_payload)
    if response.status_code == 201:
        print('Post created successfully')
    else:
        print('Failed to create post')

if __name__ == '__main__':
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(mqtt_broker_ip)
    client.loop_forever()
```

Excerto de Código A.2: Código utilizado no *gateway*.

Bibliografia

- [1] Vitor Pinheira, Maria João Moreira, and Maria João. Dinâmicas populacionais, gerações e envelhecimento processos de decisão na criação de serviços e organizações: Avaliação das necessidades e expectativas de comunidades envelhecidas decision-making processes in creating services and organizations: Assessing the needs and expectations of aged communities, 07 2018.
- [2] Radosveta Sokullu, Mustafa Alper Akkas, and Eren Demir. Iot supported smart home for the elderly. *Internet of Things*, 11:100239, 2020.
- [3] Soe Ye Yint Tun, Samaneh Madanian, and Farhaan Mirza. Internet of things (iot) applications for elderly care: a reflective review. *Aging clinical and experimental research*, 33(4):855—867, April 2021.
- [4] Gonçalo Marques, R. Pitarma, Nuno M. Garcia, and Nuno Pombo. Internet of things architectures, technologies, applications, challenges, and future directions for enhanced living environments and health-care systems: A review, Jan 2019. <http://hdl.handle.net/10400.6/8259>.
- [5] Develco: Home care. <https://www.develcoproducts.com/home-care/>.
- [6] Brian Chang. Mokosmart: Choose iot for elderly care, Dec 2022. <https://www.mokosmart.com/choose-iot-for-elderly-care/>.
- [7] Contactto: Mais tecnologia e segurança. <https://informacoes.contactto.care/>.
- [8] Gilson Esteves. Iot no cuidado de idosos, 2018. <https://pt.linkedin.com/pulse/iot-cuidado-de-idosos-gilson-esteves>.
- [9] Stephen W Hawking. Properties of expanding universes, Mar 1966.

- [10] Ian Sommerville. *Software Engineering, 10^a Edição*. ISBN: 978-0-13-394303-0. Pearson Education, 2016.
- [11] Lucid Software Inc. Lucidchart - Diagramas de Atividade, 2022. [Online] <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-atividades-uml>. Último acesso a 08 de Abril de 2023.
- [12] Lucid Software Inc. Lucidchart - Diagramas de Componentes, 2023. [Online] <https://www.lucidchart.com/pages/pt/diagrama-de-componentes-uml>. Último acesso a 07 de Abril de 2023.