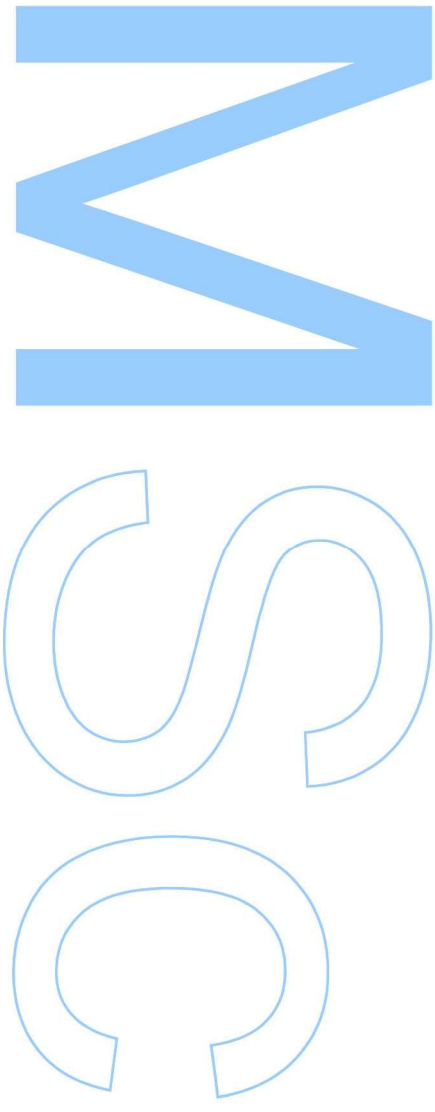

# Sistema de evacuação de edifícios em caso de emergência

André Miguel Resende Pinheiro
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciências dos Computadores
2020

**Orientador**
Sérgio Armindo Lopes Crisóstomo, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto
Instituto de Telecomunicações

**Coorientador**
Rui Pedro de Magalhães Claro Prior, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto
Instituto de Telecomunicações



# Abstract

The safety of people, inside buildings, has become an increasingly important issue. All buildings are represented through a blueprint, which indicates all the features and details present. Through this blueprint, an emergency plan is developed, which despite being static, in possible changes in the environment, in the existence of a fire, informs about a fixed emergency exit route. However, due to the limitations they present, it will be necessary to implement a dynamic evacuation system in the building, which allows different routes, depending on the temporary characteristics of the building.

So that an evacuation system can be built, several interconnected factors are needed. It will be necessary to represent the building computationally, so that it is able to locate the individuals, the respective fire spots and all the important details for the discovery of the best route. In the search for different routes, diversified metrics, dealing with factors such as distance, time or danger, are used in the search algorithm for the safest way to an exit, where some of these values are related to the information received from the fire sensor network, spread out through different locations in the building.

After a study about the existing systems developed in this area, innovation in each of the different parameters of an evacuation system was the main contribution of this project, defining rules for the implementation of a computational blueprint and the development of an algorithm, through certain metrics and restrictions, which allows providing guidance routes to the safest exit, that can make a difference in the safety of people during their evacuation of buildings. After these new ideologies, a practical system was developed, where mechanisms were structured to exchange information between sensors and the server and all the processing of the system, until the best path is revealed, represented in the interface of a mobile device.

# Resumo

A segurança das pessoas, dentro de edifícios, tem vindo a tornar-se cada vez mais um assunto de enorme relevo. Todos os edifícios são descritos através de uma planta, representando esta, todas as características e detalhes presentes nestes. Através desta, desenvolve-se uma planta de emergência, que apesar do seu aspeto estático, em possíveis mudanças no ambiente, na existência de um incêndio, informa acerca de uma rota de saída de emergência fixa. Porém, por causa das limitações que estas apresentam, será necessário implementar, no edifício, um sistema de evacuação dinâmico, que permita elaborar diferentes rotas, consoante as características momentâneas do edifício.

Um sistema de evacuação necessita de diversos fatores interligados para se poder construir. Será necessário a representação do edifício computacionalmente, para se conseguir localizar os indivíduos, os respetivos focos de incêndio e todo o detalhe importante para a descoberta da melhor rota. Na procura das diferentes rotas, diversificadas métricas, lidando com fatores como distância, tempo ou perigo, são utilizadas no algoritmo de procura do caminho mais seguro até uma saída, estando alguns destes valores relacionados com as informações recebidas da rede de sensores de incêndio, espalhados por diferentes locais do edifício.

Após um estudo, acerca dos sistemas já existentes e desenvolvidos neste âmbito, a inovação em cada um dos diferentes parâmetros de um sistema de evacuação foi a principal contribuição deste projeto, definindo-se regras de implementação de uma planta computacionalmente e do desenvolvimento de um algoritmo, mediante determinadas métricas e restrições, o qual permite fornecer rotas de orientação até à saída mais segura, que possam fazer toda a diferença na segurança das pessoas durante a sua evacuação de edifícios. Após estes novas ideologias, foi desenvolvido um sistema prático, onde se estruturou mecanismos para troca de informação entre sensores e servidor e todo o processamento do mesmo, até à exposição do melhor caminho, representado na interface de um dispositivo móvel.

**Palavras-chave**: auxílio à evacuação, incêndio, planta, caminho mais curto

# Agradecimentos

Gostaria de agradecer à minha família, aos meus amigos e todas as pessoas que contactaram comigo até chegar a este momento da minha vida. Continuamente, destacaria duas pessoas importantes neste projeto, os meus dois orientadores, Rui Pedro de Magalhães Claro Prior e Sérgio Armindo Lopes Crisóstomo, por toda a ajuda demonstrada neste ano difícil de pandemia, onde passamos por momentos complicados e me auxiliaram na superação de todos os problemas presenciados.

Obrigado.

**À minha familía!**

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Lista de Blocos de Código

# Acrónimos

**DCC** Departamento de Ciências dos Computadores, da Faculdade de Ciências da Universidade do Porto

**FCUP** Faculdade de Ciências da Universidade do Porto

**HTTP** *Hypertext Transfer Protocol*

**WS** *Web Service*

**SOAP** *Simple Object Access Protocol*

**REST** *Representational State Transfer*

**XML** *Extensible Markup Language*

**URI** *Uniform Resource Identifier*

**HTML** *HyperText Markup Language*

**JSON** *JavaScript Object Notation*

**PS** *Publish-Subscribe*

**BD** Base de dados

**SVG** *Scalable Vector Graphics*

**ED** Estrutura de dados

**FE** *Front-End*

**BE** *Back-End*

**WSN** *Wireless Sensor Network*

**AD** Algoritmo Dijkstra

# Capítulo 1

# Introdução

Pelo mundo fora, atualmente, existem diversos assuntos a necessitar de respostas. Nos dias de hoje, um dos que está em foco é a segurança no interior de edifícios. Esta tem tido um efeito bastante mediático, visto que é algo que nos engloba a todos nós e que preocupa cada vez mais. O surgimento deste exponencial desenvolvimento, a nível de infraestruturas, tem sido um dos pontos chave na ameaça da segurança da população, desde a construção de edifícios de pequenas dimensões, a grandes dimensões. Tal preocupação agrava-se, a partir do momento em que qualquer pessoa se encontra dentro destes. A sua sobrevivência, caso ocorra algum desastre, fica exposta a diversos factores condicionantes, que podem colocar em causa a sua evacuação.

## 1.1 Motivação

A maioria dos edifícios são representados graficamente por uma planta a uma determinada escala, que permite saber a localização de cada compartimento, a relação entre cada um e também medições subjacentes a estes. Estas são necessárias, de forma a conseguir-se ter uma noção do meio físico onde se está inserido. Por outro lado, são precisas quando falamos de segurança. Através destas, pode-se elaborar ideias, que permitam ajudar no combate ao perigo iminente, num edifício. Uma das deduções feitas através de uma planta é a elaboração de uma planta de emergência, como podemos ver na figura 2.4, permitindo esta a identificação de uma rota de saída urgente, em caso de alarme e, consequentemente, que seja necessário a cada indivíduo a sua saída do edifício. Sendo estes mapeamentos de evacuação importantes na orientação das pessoas, em situação de fuga necessitam portanto, de estar em locais em que possam ser visionados com bastante facilidade, permitindo a fácil deteção de alguém.

Apesar da elaboração destas rotas de emergência referidas em 2.4, através das setas a verde, em cada construção, estas são estáticas para qualquer tipo de problema que possa existir, não havendo assim mudança de rotas consoante a situação vivida no momento de alarme. Este facto remete para limitações que estas plantas de emergência têm, podendo o caminho escolhido por estas, não ser o mais apropriado num determinado momento. Consequentemente, o problema destas rotas estáticas, leva o evacuado, a não ter uma noção precisa do que vai encontrar pela frente, ao longo do percurso.

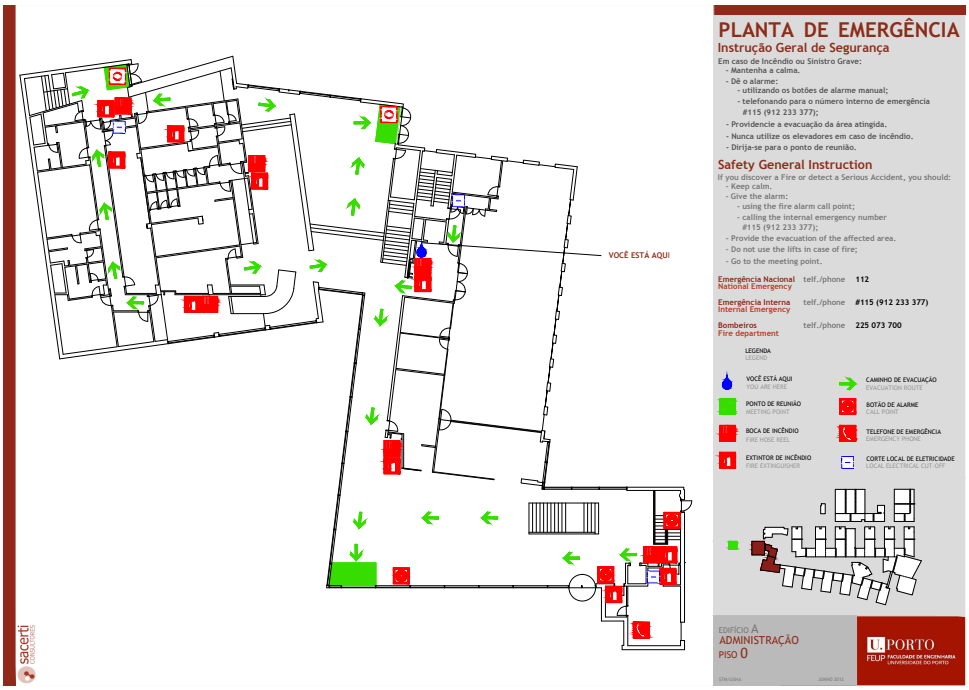



Figura 1.1: Planta de emergência da Faculdade de Engenharia da Universidade do Porto, edifício 0, piso 0, retirado de [1].

Desenvolver novas ideias e arranjar mecanismos que permitam a cada pessoa, no interior do edifício, uma maior segurança, é uma das grandes preocupações relativas a este tema. Posto isto, colaborarmos nesta pesquisa, na obtenção de soluções inovadoras, que contribuam para a segurança das pessoas a evacuar, consoante a variação das condições numa situação de incêndio, foi algo que valorizamos, impulsionando-nos na realização deste projeto.

## 1.2 Objetivos da pesquisa

Ao longo desta dissertação, direcionamos a nossa atenção para situações onde possam ocorrer incêndios. Como tal, com o desenrolar do problema, múltiplos acontecimentos são constatados. Ocorrências a fazer referência, como o facto de chamas que poderão propagar a rápida velocidade e de uma forma descontrolada, falta de visibilidade, em determinados troços do percurso, são uma possibilidade. Os gases e as partículas produzidas, podem afetar de forma adversa a saúde dos indivíduos expostos. Continuamente, também podemos considerar a mudança na topologia do edifício, o abaulamento dos sistemas de comunicação no edifício ou mesmo a inexistência de saídas principais para o exterior. Por outro lado, valorizando a fase de evacuação, é preciso ter em conta a alteração a nível comportamental, das pessoas, originando problemas como engarrafamentos nas vias ou nas saídas e desorientação na rota a tomar. Como tal, conseguimos perceber que devido à alta probabilidade da ocorrência de problemas, subjacentes a um incêndio, a existência de um plano de emergência, com rotas estáticas não é, de todo, o melhor processo de auxílio durante um desastre.

Devido a todas as limitações que uma planta de emergência obrigatória, de um edifício, não consegue superar, a importância de desenvolver novas ideias, que tentem ultrapassar estas dificuldades, é bastante útil na segurança das pessoas. Como tal, tem-se vindo a desenvolver diversas formas de contornar estes problemas, através de mecanismos que possibilitam a dinamização das rotas de evacuação, em casos de emergência.

O nosso foco estará ligado à exploração de novas ideias, para um melhor aproveitamento da informação, em ambientes em alerta. Ideias, de como representar uma planta, num meio computacional, serão expostas. Não pretendemos elaborar uma plataforma que faça a conversão da planta diretamente para um sistema, nem contornar todas as especificidades deste tema, apenas estabelecer regras de representação para os diferentes tipos de espaço num edifício, que permitirão ajudar numa implementação futura da tal plataforma.

Novas métricas, juntamente com novos algoritmos desenvolvidos, em função destas, serão retratados ao longo desta dissertação. A inclusão de todos os fenómenos a lidar, durante um incêndio, será uma dificuldade a ter em conta, pois foge à nossa área de estudo. Porém, numa generalização destes factos e tendo em atenção os mesmos, pretendemos apresentar uma solução no final, que permita que cada pessoa a evacuar faça um percurso seguro e rápido até uma saída.

Nestes ambientes, temos de lidar com sensores. Destes provêm informações importantes de alerta, acerca do ambiente. Para a sua utilização, decidimos desenvolver uma API que permita lidar com esses dados, de forma rápida e esclarecedora, na medida em que, a essa mesma informação se atribuir um valor de perigo, a uma zona limitada pelo raio de observação do sensor.

Apresentando novas ideias e metodologias, inserir tudo isto num sistema será importante. Teremos como objetivo englobar todo o estudo e a discussão de novas ideias num sistema final. Pretendemos estruturar, devidamente, o nosso sistema, conseguindo proporcionar uma simulação, de todas as nossas ideias propostas, através de uma interface final.

## 1.3 Estrutura da dissertação

Todo este processo de investigação, de busca pelas melhores ideias, para atingir o nosso objetivo, apresenta várias etapas ao longo da tese. Esta tese está estruturada em sete capítulos. Segue-se uma breve descrição para cada um deles.

**Capítulo 1 - Introdução.** Breve introdução à nossa área de pesquisa, todo o contexto relacionado com o nosso tema, objetivos da nossa pesquisa, algumas contribuições, tanto como a organização da tese.

**Capítulo 2 - Contextualização e trabalho relacionado.** Introduzimos a noção de sistema de evacuação, de forma genérica, e discutimos várias metodologias já desenvolvidas, relacionadas com o nosso tema: tipos de sistemas usados, diferentes formas de representação de um edifício computacionalmente, como é mapeada essa representação numa rede geral e os algoritmos utilizados. Por último, neste capitulo, damos exemplos de alguns sistemas, com interfaces de orientação já existentes.

**Capítulo 3 - Análise e especificação.** Estabelecimento da abordagem utilizada para atingir os nossos objetivos. Apresentação de metodologias para a representação da planta a nível computacional, como também o algoritmo pensado, métricas e restrições discutidas. Detalhada explicação das diferentes ideias desenvolvidas através de imagens e também algum pseudo-código, de forma a se entender melhor cada conceito.

**Capítulo 4 - Implementação do sistema.** Descreve a implementação das nossas ideias, expostas no capítulo anterior, numa vertente prática. Neste, desenvolvemos um sistema completo para todas as etapas a realizar, no processamento de informação vinda dos sensores, até à parte de mostrar ao utilizador a melhor rota.

**Capítulo 5 - Resultados e análise.** Mostra uma simulação feita a todo o nosso sistema criado, consequentemente a todas as ideias pensadas. Apresentamos uma vítima, localizada numa determinada posição e variamos o resultado da melhor rota, conforme os disparos de determinados sensores. Fazemos também uma estatistica temporal para cada uma das etapas do sistema desenvolvido.

**Capítulo 6 - Conclusões e trabalho futuro.** É dada uma perspetiva do trabalho efetuado e também o que deve ser feito no futuro, com consideráveis aperfeiçoamentos, de forma a conseguirmos melhorar cada vez mais o repositório de pesquisa nesta área.

# Capítulo 2

# Contextualização e trabalho relacionado

Ao longo deste capítulo será apresentada informação genérica das etapas percorridas durante uma evacuação e algumas ideias já usadas na elaboração de sistemas de evacuação de pessoas. Sendo o propósito deste trabalho implementar diferentes ideias, estamos principalmente interessados em compreender algoritmos já existentes para o efeito, tipos de sistemas utilizados, formas de representar uma planta de um edifício computacionalmente e de que forma passar a informação às pessoas. Contudo, devemos perceber que existem mecanismos que não têm uma grande precisão, visto que para o terem, precisavam de apresentar outras características para complementar o demonstrado, podendo carecer de segurança.

## 2.1 Sistema de evacuação

A evacuação dos edifícios é uma das acções mais importantes a realizar, na segurança das pessoas dentro deles. Qualquer pessoa gosta de saber que existe um plano para sair do edifício, caso ocorra alguma emergência. Para este sistema funcionar será necessário a identificação e o uso de diversas noções que permitam tal procedimento.

Este sistema pode ser feito através da saída dos evacuantes de uma forma simultânea, podendo desta forma existir problemas de "engarrafamentos", durante a evacuação, pois poderá não existir espaço, ao longo dos percursos, para aglomerados de pessoas. Por outro lado, poderá ser efetuado por etapas, faseando a retirada das pessoas dos compartimentos do edifício. Este tipo de evacuação tem-se direccionado mais para edifícios altos, sendo que estes podem capacitar um número elevado de indivíduos [32] .

A maior parte dos edifícios tem como pertences diversos aparelhos sensoriais. No caso da prevenção de incêndios, existem diversos tipos de sensores, destinados a informar todos os utilizadores de um determinado local, da iminência da ocorrência de um incêndio ou do princípio do mesmo. São aparelhos que medem a temperatura ambiente, outros a ocorrência de fumo ou mesmo de chamas que, quando os valores recebidos são diferentes dos normais, significam que estaremos perante uma situação de perigo. Após esta identificação, de que alguma coisa não está correta, o alarme poderá ser estabelecido a um nível geral, para todo

o edifício, restritamente para o pessoal da segurança e localmente só na zona determinada em que é acionado [16] .

A partir do momento em que tal alarme é disparado, o método mais tradicional e estático, nos dias de hoje, é o caminho das saídas de emergências do edifício, como retratado no capitulo 1. Estas, normalmente, são identificadas por sinalização de segurança, que orienta os ocupantes do mesmo, a seguirem um percurso com diferentes especificidades, para permanecerem em segurança na ocorrência de um incidente. Este tipo de sinalização pode estar representado através de sinais coloridos, acústicos, verbais ou gestuais. Por sua vez, deve-se inserir a sinalização em sítios de fácil visualização dos mesmos, sendo locais relativamente iluminados, que permitam uma intuição espontânea e um vislumbre rápido por parte das pessoas em fuga. Esta sinalização estará disposta , nas vias de evacuação, estas vias podem ser horizontais ou verticais. Por vias horizontais denominamos corredores e espaços amplos. Por outro lado, às vias verticais são rampas, escadas ou de tapetes rolantes inclinados.

As saídas de emergência estão em caminhos pré-estudados, de forma a colmatarem diversos problemas que poderão surgir, aquando de uma evacuação. Por exemplo, o número de pessoas que pode passar pelos respetivos percursos do caminho escolhido, a uma distância mínima até uma saída segura, o tempo que demora a chegar a um sítio seguro ou mesmo a tentativa de evitar caminhos com obstruções.

Especificando para ocasiões de incêndio iminente, pretende-se tentar levar os ocupantes por vias com resistência ao fogo, ou mesmo com controlo de fumo, usando muitas vezes pelos especialistas neste tipo de análises, as chamadas portas de emergência, devendo estas serem equipadas com sistemas de abertura, dotados de barras anti-pânico. Tais portas pretendem evitar a propagação do fogo, dando a possibilidade de uma abertura fácil, no sentido da evacuação, e de certa forma, ser mais um indicador na navegação do evacuante, até à saída final. Do mesmo modo, em muitos edifícios existem escadas suplementares, específicas para situações de urgência.

Por outro lado, o melhor caminho, por vezes, pode ser aquele que oriente, para um sítio temporário, onde cada pessoa terá meios de proteção para se auto defender, ou para algum espaço exterior no edifício. Este tipo de caminho pode ser considerado, muitas vezes, quando não há nenhum caminho possível até umas das saídas. Uma das opções bastante utilizada, em último recurso, será a saída do edifício através de janelas de algum compartimento da construção. Embora este tipo de saída possa acarretar um risco demasiado elevado para uma pessoa, por vezes é a única opção.

Um dos tópicos relevante neste tema é a evacuação das pessoas com algum tipo de incapacidade. Uma pessoa com deficiência auditiva não consegue ouvir um alarme de incêndio. Um cego não consegue ver a sinalética de evacuação. Uma pessoa em cadeira de rodas não pode fugir pela escada. Estas incapacidades manifestam-se de diferentes formas e, como tal, podem ter implicações funcionais diferentes, durante uma evacuação de emergência. Posto isto, o ideal será haver alguém capaz de ajudar estas pessoas, corredores largos de forma a passarem as cadeiras de rodas, sinalização de emergência para pessoas com deficiência, elevadores de emergência ou rampas de acesso específicas. Porém, em indivíduos sem nenhum tipo de incapacidade, por vezes o seu comportamento altera-se e torna-se agitado, o que levará a perigos, caso não existam ferramentas de forma a amenizar este comportamento reativo [4] .

Contudo, não conseguimos dizer de uma forma realista e concreta, que estes métodos de evacuação tradicionais são os mais indicados nos seus objetivos e também que solucionam o problema de pessoas com características diferentes. Apesar de apresentarem uma proposta importante para a deslocação das pessoas, novos estudos são necessários para combater a falta de dinâmica destes sistemas. Em 2.1 temos representada as diferentes etapas mencionadas, de forma ilustrativa. Começando por o sensor captar as informações, envia-as para uma central, a qual dispara um alarme, levando as pessoas a procurarem o melhor caminho até uma saída, com a sinalética de orientação, em caso de emergência.

Figura 2.1: Resumo das etapas a percorrer desde o início de um incêndio.

## 2.2 Tipos de sistema

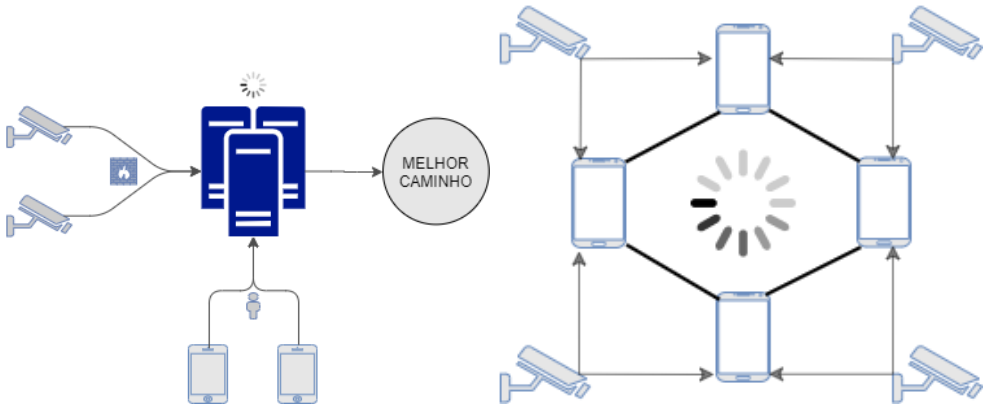
Para podermos começar a construir um sistema, temos primeiramente de definir a sua arquitetura, as suas características e perceber como o seu sistema funcionará. No âmbito em que este projeto está inserido, a implementação de um grafo, com diversos nós, pode ser uma ideia, de forma a que consigamos receber informações pertinentes para pôr em funcionamento um sistema. Posto isto, convém perceber as possibilidades de topologia de rede existentes, bem como as suas especificidades.

O sistema pode ser implementado de uma forma centralizada. Sistemas centralizados usam arquitetura cliente/servidor, onde um ou vários clientes estão conectados diretamente a um servidor central. Frequentemente, cliente e servidor comunicam a partir de hardware distintos, mas residem no mesmo sistema. Neste caso, o servidor recebe informação do cliente, executa o processamento e partilha com os clientes o resultado. Contudo, na ocorrência do servidor ficar afetado e deixar de funcionar, todo o sistema deixará de ser processado [33] . Analisando a figura 2.2a, tomando como principio o nosso caso de estudo, o servidor central obtém informações acerca dos sensores e a posição de cada indivíduo no edifício, a partir daí fornece a melhor solução a tomar.

Por outro lado, um sistema poderá ser arquitetado de maneira distribuída. Neste tipo de sistema, cada nó fará o processamento de forma individual, ou seja, recolhe a informação e realiza o processamento ele mesmo. Diferenciando-se do formato de sistema apresentado em cima, neste caso a agregação da informação é processada individualmente por cada nó e o comportamento final da rede ocorre mediante os resultados desse processamento. Normalmente são definidas como redes ad-hoc [12][24] . Observando a figura 2.2b, percebemos que, contrariamente a um sistema centralizado, cada dispositivo móvel receberá informações dos sensores e fará toda a execução neles mesmos, sendo que numa vertente

centralizada será tudo feito no servidor. No final, essa informação é partilhada pela rede, determinando a melhor solução possível.

Posto isto, é necessário que o sistema esteja já estudado e definido antes de ocorrer algum problema. A sua implementação no edifício trará consequências positivas em relação à sua ausência. Para isto, a importância de perceber o layout do edifício é significativa para definir qual dos sistemas trará aspetos mais vantajosos para o pretendido [19] .




(a) Sistema centralizado (b) Sistema descentralizado

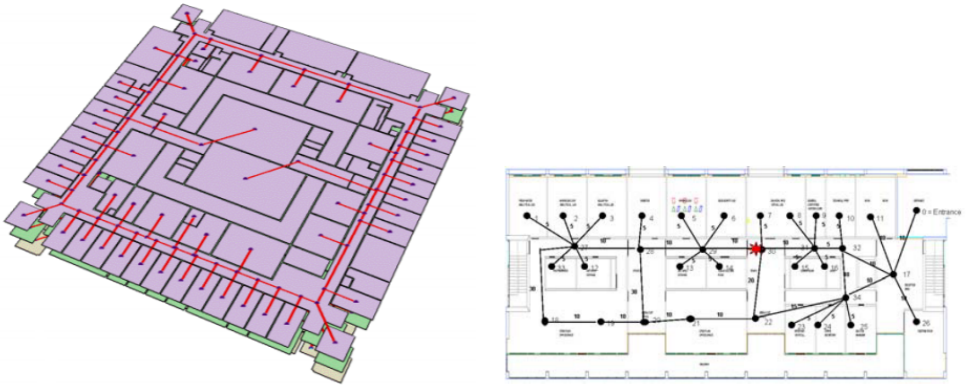
Figura 2.2: Tipos de sistema existentes e a sua representação.

## 2.3 Representação do edifício

Para conseguirmos arranjar uma solução e respetiva orientação para as pessoas a evacuar é necessário transformar a planta geográfica do edifício, numa forma computacional. Para tal, várias representações foram definidas, cada uma com as suas especificidades.

A representação do mapa do edifício num grafo é a ideia usada nos modelos de evacuação existentes. Este permite conseguir uma perspetiva precisa acerca de distâncias, tempos entre cada compartimento ou posições referentes aos diversos compartimentos do edifício [14] . Os nós do grafo normalmente são identificados como salas, corredores, cruzamentos, e as arestas sendo a ligação correspondente a mudanças de compartimentos, dentro de um edifício ou intervalos de espaços num corredor. A representação de portas tanto já se utilizou como nó [21] ou como aresta [20] . Em relação a escadas, uns usam um simples nó a representá-las, outros usam vários pontos na escada a dividir os pisos por onde esta passa [44] . De fazer referência que cada compartimento de um edifício, normalmente apresenta uma anotação para se o poder identificar [29] . Já se definiu a ideia de um edifício estar totalmente representado através de uma grelha de nós, de forma a lidar com vários obstáculos que possam aparecer no edifício [23] . Para uma melhor eficiência a nível de complexidade no processamento das melhores rotas percorrendo um grafo, a introdução de um super nó que obtenha ligações de todas as saídas de um edifício é uma ideia que traz vantagens [26] . Tanto a figura 2.3a, como a 2.3b mostram dois exemplos de representações computacionais já utilizadas.

Vários sistemas baseiam a sua representação no modelo *Wireless Sensor Network*, um sistema centralizado. A rede é composta por sensores, os quais correspondem aos nós,



(a) (b)

Figura 2.3: Exemplo de duas representações computacionais da planta, de formas diferentes. 2.3a foi implementado em [10], sendo um grafo que apresenta um maior número de nós na sua constituição. 2.3b pertence a [40] apresentando uma disposição de nós diferente, mais propriamente apresentando apenas um nó para cada corredor.

estando estes espalhados pelo edifício. Estes podem coletar diferentes tipos de informações, conforme os respetivos tipos de sensor e enviam as suas informações para o servidor central para este gerir a rede a partir destes [35] . Porém, esta disposição sem fios origina mais gastos e é preciso ter em atenção a durabilidade de cada sensor. Caso fiquem sem bateria, na ocorrência de uma falha, podem ocorrer problemas na obtenção dos resultados pretendidos. Para além destes factos, um sensor pode estar inserido dentro de uma região liderada por um *gateway*, sendo este um nó de comando, formando assim várias clusters, que comunicam os resultados entre si [47] .

Por outro lado, os nós podem ser representados por *decision nodes*, sendo que aparecem em sistemas descentralizados. Estes, como o nome indica, processam a informação e tomam decisões, com base nos resultados da sua execução. Na obtenção da informação, entre cada ligação de um par de *decision nodes* está um sensor, que faculta a informação necessária para o processamento [18] .

Após a especificação de como queremos representar o edifício computacionalmente, todas as anotações, posições geográficas de cada parte da planta, existe um sistema que guarda todas estas informações e fornece-as posteriormente. O Geographic Information System (GIS) armazena todos estes dados geográficos, matérias, métodos e trabalha-os, de maneira a fornecer ao utilizador do sistema, informações pertinentes para o mapeamento da rede [10] .

## 2.4 Mapeamento da rede

Após definir-se a representação computacional de uma planta de um edifício como um grafo, várias possibilidades existem de atribuir às suas arestas ou aos seus nós, diferentes tipos de pesos. Consequentemente, cada tipo de valor inserido, dependerá de que métrica ou conjunto de métricas possamos querer usar.

### 2.4.1 Grafo de navegação

Querendo um sistema de evacuação, a obtenção de um caminho mais seguro e ao mesmo tempo mais rápido, encontrar uma rota que nos leve a uma saída do edifício, é comum usar-se a distância física como um valor a ter em conta. Esta, por sua vez, é mapeada nas arestas do grafo, tendo como significado a distância física que leva a percorrer de um nó inicial, a um nó destino, podendo saber-se a distância até ao nó vizinho ou mesmo até a um nó saída [29] .

Por outro lado, falando de distância física, não podemos deixar de parte o fator tempo. O tempo de navegação, permite-nos perceber quanto tempo demoramos a chegar até uma saída. Em condições de incêndio, o valor de tempo, em cada aresta, pode ser usado para medir o tempo que uma consequência de um incêndio demora a chegar de um nó a outro, jogando assim com os valores do grafo. Tendo em conta que na ocorrência de fogo, corresponderá a um menor tempo face à apenas existência de fumo, acabando por ser importante no tempo de saída total [34] . Juntando estes dois medidores de tempo, também podemos usar este fator para medir o tempo de atraso que temos à nossa disposição, ou seja, caso aconteça alguma irregularidade durante o percurso, sabemos que temos um valor de tempo disponível para ultrapassamos uma adversidade, sem que sejamos atacados pelo fogo [40] .

Contudo, o tempo não nos ajuda só a lidar com a propagação do fogo, mas sim com outro fator importante numa evacuação, que é capacidade de pessoas possível, numa determinada via, num edifício. Um sistema que faça a sua gestão, tendo em conta este fator, pensa em evitar congestionamentos que podem levar a propagações desastrosas no objetivo final. O *Capacity constrained route planner* (CCRP) planeia ao longo de iterações, correspondentes a um determinado intervalo de tempo, a ocupação que terá de pessoas, de um determinado nó num grafo [27] . Também já foram elaborados certas fórmulas matemáticas, com o objetivo de ajudar a calcular o fluxo de pessoas a passar num certo espaço [10] .

A maioria das soluções são feitas a partir do uso de filas de espera, do uso de sensores espalhados no edifício, que medem a quantidade de pessoas a passar por um certo ponto ou mesmo, através de cálculos que deduzem a densidade populacional, num determinado local. Através destes cálculos poderão definir-se as rotas de cada pessoa, em diferentes locais de um edifício, de forma a que numa passagem, durante a rota, não tenham que abrandar, ou mesmo parar o seu movimento [7] .

### 2.4.2 Grafo de comunicação

Como visto na secção acima, muitos sistemas utilizam diferentes métricas relacionadas com distância, tempo e perigo. Contudo, podemos efetuar o encaminhamento das pessoas para a evacuação, relacionando com o transporte de dados, numa rede de comunicação.

Numa rede de comunicação, as informações podem ser enviadas através de pacotes, podendo cada ponto receber pacotes de confirmação de entrega e saber o estado dos seus vizinhos. *Gossip Protocol* está presente em algumas ideias investigadas. Basicamente, a informação recebida é passada aos seus vizinhos e devido ao facto da informação ser espalhada por toda a rede, sabe toda a rede, o que se está a passar no sistema inteiro [25] .

Neste caso saber quais os caminhos disponíveis, as distâncias a percorrer, elaboraram-se

sistemas aliando os dois propósitos. Os nós da rede, através do número de saltos efetuados de nó para nó, conseguem saber quanto saltos necessitam até chegar ao nó final e quais os seus vizinhos. Todavia, também permite saber a localização dos sensores espalhados pela rede. Para isto poder acontecer, certos modelos usam um esquema de encaminhamento utilizado numa rede de comunicação, *Destination-Sequenced Distance-Vector Routing* (DSDV). O ajustamento feito a este protocolo fez com que cada nó, contendo um sensor, conseguisse saber o número de saltos até ao final, baseado nos dados de distância recebidos pelos seus vizinhos. Apesar de conseguirem saber o número de saltos até à saída, não necessitam de saber o caminho até esta. Quando se despoleta a ocorrência de algo anormal, as arestas ligadas a esse nó são cortadas e a rede é reprogramada novamente [12] . Por outro lado, há sistemas que no caso de falhas, dividem a rede em sub-redes, para que estas possam trabalhar autonomamente e, consequentemente, apresentar os resultados pretendidos [33] .

Para além de sistemas baseados no DSDV, outros baseiam-se no *Temporally ordered routing algorithm* (TORA). Neste caso, os nós de saída enviam pacotes para toda a rede, para estes poderem identificar o número de saltos até ao final, identificados como altitude. Se chega um pacote com um valor mais alto de saltos, o valor mantém-se e mantém-se também a saída referente a esse número de saltos. Caso algum nó fique sob perigo iminente, o número de saltos aumentará nesse nó. Continuamente, na parte da navegação, cada nó escolhe sempre o que contiver um valor de altitude menor, até chegar ao zero, que será o valor da altitude da saída [41] . Comparando com este caso, pensou-se no processamento do algoritmo de Dijkstra, usando valores de uma distância ficticia. No caso de haver risco num local, a distância ficticia aumentava sobre a real [8] .

Juntando estes meios de transporte de informação, numa rede com a WSN já referida em cima, num edifício poderão estar espalhados vários sensores fixos e os dispositivos móveis espalhados pelas pessoas, formando estes, a rede. As trocas de informações são efetuadas entre dispositivos móveis, onde estes guardam as mensagens e enviam, quando tiverem um bom alcance de comunicação para com outro dispositivo. Para estes saberem a sua posição, os *sensor nodes* servem de postos de referência, enviando estas mensagens aos dispositivos móveis, mais próximos da sua localização [22] .

### 2.4.3 Comportamento das pessoas

Um dos factores importantes, na evacuação das pessoas, são as questões comportamentais destas, na reação aos alarmes de emergência e durante o seu percurso de saída. Porém, é um assunto de difícil estudo, pois cada pessoa tem reações diferentes ao medo. Umas por um lado podem paralisar, enquanto outras podem ficar num descontrolo total, de tal forma que começam a andar desorientados. Elaborar algo com base nestas duas extremidades é muito complexo, não se obtendo uma boa solução, visto que podem existir diversas condicionantes.

No entanto, já foram desenvolvidas bases de dados, com análises a tempos de reação das pessoas, ao alarme, tempos de saída, tempos de subidas de escadas ou descidas para diferentes idades, entre outros detalhes. Isto ajudará na questão de mapeamento da rede, pois poderemos ter valores mais precisos de tempos de saída, de cada individuo, face à sua constituição física. Contudo, é muito difícil de especificar, tal e qual, pois o ambiente do edifício, em situação de incêndio, pode ser alterado constantemente.

Postas estas duas vertentes, o comportamento de uma pessoa face ao alarme ocorrido e os seus tempos de deslocamento, dentro de um edifício, face às suas características, são aspetos a ter em conta, contudo são assuntos que apresentam um valor de precisão baixa, onde podem ser tratados todos os problemas, mas assumem-se algumas especificações deste assunto, no desenvolvimento de algum tipo de metodologia [38] .

### 2.4.4 Algoritmos caminho curto

Para conseguirmos percorrer a rede à procura da melhor rota, necessitamos de encontrar o algoritmo apropriado para o fazer, estabelecer as métricas que pretendemos utilizar, restrições necessárias a ter em conta, como todos os fatores que pretendemos assumir numa ocorrência de um incêndio.

A partir do momento em que temos os critérios, restrições e alternativas definidas, podemos empregar o *Multicriteria Decision Making* (MCDM). Para conseguirmos efetuar este procedimento, usa-se o *Analytic Hierarchy Process* (AHP), o qual tendo vários elementos que podemos utilizar como distância, congestionamento, risco de propagação de fogo, podemos juntá-los e atribuir um valor de perigo geral. Este utiliza uma *Pairwise comparison matrix*, pelo qual são comparadas as importâncias de cada fator para o perigo. Após este procedimento e utilizando *Synthesising matrix* obtemos a importância de cada fator, em forma de percentagem [36] . Posto isto, para mapear a rede, o utilizador, por exemplo pode assumir uma certa intensidade para a propagação do risco, sendo que este é representado graficamente por um *radial buffer* [10] . Porém, já foi idealizado que mediante a propagação do fogo, nós e arestas, inerentes ao fogo, serão eliminados do grafo a percorrer [11] ou o custo de cada aresta, apresentar valores infinitos durante o cálculo do algoritmo de Dijsktra [46] .

Na busca pelas melhores rotas existem os chamados algoritmos de caminho mais curto. Algoritmos que através dos valores em cada aresta de um grafo, de um nó inicial para um final, vai escolhendo o nó a seguir, consoante as métricas a usar, como distância*perigo de uma certa aresta [19] . Por outro lado, há modificações que se podem fazer no grafo, na ocorrência de um foco de incêndio, numa certa região, alterando a bidirecionalidade das arestas, apresentando apenas uma só direção, consoante a propagação do fogo. O bloqueio do nó que estabelece contacto com o fogo, inclusivamente as suas arestas adjacentes é uma das alternativas, com o objetivo de ninguém poder passar por aquele local [37][11] . Existem uns algoritmos padrão usados frequentemente na busca pelo caminho mais curto, como algoritmo Dijkstra (AD), Bellman-Ford, A*, Floyd-Warshall.

O algoritmo de Djikstra soluciona problemas em que se pede o caminho mais curto de um ponto inicial até um final, num grafo dirigido ou não dirigido. O início poderá corresponder às saídas, percorrendo a rede ao contrário, como acontece em alguns casos [11] . Porém, este algoritmo só é eficaz se cada aresta do grafo for positiva pois, ao contrário iria reduzir a distância de um ponto ao outro, o que alterava a veracidade do mesmo. Passando a explicar o funcionamento do algoritmo, no início temos um vértice inicial I, que pertence a um conjunto S de um conjunto de vértices já fechados. Depois de fechar o vértice I, vemos quais os seus vértices adjacentes e calcula-se a distância de I a cada um dos vértices. Se a distância calculada for menor do que a distância presente num desses vértices, já calculada, vinda de outro vértice, o caminho para esse vértice adjacente terá como precedente o I e uma

nova distância, chamando a esta situação, o método de relaxamento das arestas. Posto isto, para escolher o próximo vértice a tratar, temos os vértices que ainda não estão fechados e escolhemos sempre o vértice com menor distância. Consequentemente, juntamos este vértice ao conjunto S e fazemos novamente todo o procedimento já explicado acima para o novo vértice [13] .

O algoritmo de Bellman-Ford é outro algoritmo que permite a possibilidade de obter o caminho mínimo, de um ponto inicial a um final. Relativamente ao algoritmo Djikstra, este é parecido com o Djikstra, na medida em que faz o relaxamento novamente das arestas, com a única diferença que relaxa todas as arestas ao mesmo tempo, num conjunto de n-1 passos, sendo n o número de vértices no grafo. Por outro lado, o Bellman-Ford também permite calcular distâncias com o valor das arestas negativo, fazendo uma verificação no final da existência de loops infinitos caso hajam [13] .

O algoritmo A* usa uma heurística apropriada para atingir um comportamento, as suas escolhas serão com base na heurística selecionada. Explicando o algoritmo, são escolhidos dois nós, como o inicial K e o final. Começando-se no inicial a ir para o conjunto dos visitados, extraem-se os filhos deste, respetivamente L e M, que vão ter cada um os seus custos em função da heurística. Este, por sua vez, vai expandir os seus filhos também e neste ponto é que está a chave do algoritmo. Após esta última expansão, teremos estes dois nós filho, mais o nó K e o que apresentar o menor custo é o que vai ser expandido, caso tenha filhos para apresentar. Termina o algoritmo quando o nó escolhido com menos custo, relativamente aos restantes, é o nó final [13] .

O algoritmo Floyd-Warshall usa uma matriz, onde estão os valores do peso entre dois vértices e é com esta matriz que se lida ao longo do algoritmo. Permite valores negativos nas arestas, como os dois algoritmos acima mencionados. O funcionamento deste método, de encontrar o caminho mais curto, começa por assumir um subconjunto com k elementos, sendo este o número de elementos dos vértices do grafo. Consequentemente, estes valores vão servir ou não de vértices no caminho entre dois, se o caminho entre dois vértices for mais curto, a passar por k. A matriz de distâncias será atualizada ao longo das iterações feitas, chegando a uma matriz final, onde se pode visualizar as distâncias do caminho mais curto entre dois pontos [13] .

## 2.5 Sistemas de orientação

No final da análise destes aspetos importantes para o sistema e após a obtenção da rota de evacuação mais segura, é necessário passar a informação de orientação para o indivíduo, da forma mais adequada, com a devida assertividade, para que a possibilidade de este se enganar na rota seja diminuta e, consequentemente, colocar este em perigo iminente.

A orientação das pessoas pode ser efetuada através de diferentes formas. Por meio de efeitos luminosos [34], podendo estes estarem dispostos nas paredes, ao longo do edifício ou ao longo do chão a percorrer. Este tipo de colocação das direções a seguir, pode facilitar bastante a pessoa a deslocar-se, visto que são sinais colocados em determinados sítios estratégicos, de maneira a que sejam o mais intuitivo possível.

Por outro lado, esta parte do sistema pode ser efetuado através de uma receção auditiva

de indicações. O sistema por voz pode estar pré implementado no edifício, dando dicas da melhor deslocação, através de sons sonoros. Este tipo de ideal funciona em ambientes cuja luminosidade esteja em falta, o que permitirá ser um auxílio vantajoso, no decorrer do caminho de saída [34] .

Para além destas duas hipóteses, podemos ter um sistema de orientação implementado num telemóvel. A vantagem de implementar um sistema de orientação num telemóvel é que permite às pessoas, que não estejam em concordância com o formato do edifício, terem uma noção do mesmo, a partir da interface do telemóvel. Caso o *layout* do edifício não esteja presente na aplicação, poderão estar apenas as indicações úteis a ter durante a rota de evacuação [42] . Para além desta informação para os ocupantes do edifício, uma interface para as equipas de resgate também poderá ser implementada, identificada a localização do foco de incêndio e das respetivas pessoas [31] . Assim, através deste meio, conseguimos ter uma maior preceção da nossa deslocação no edifício, podendo-nos fornecer alguns indicadores do que nos falta percorrer até ao final, de modo a ajudar no entendimento de uma forma rápida do percurso.

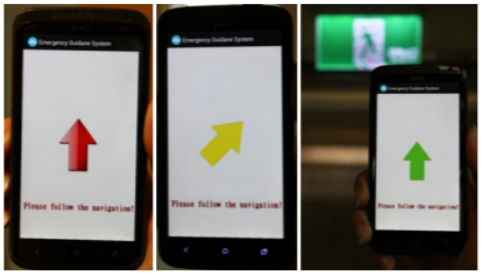


Figura 2.4: Exemplo de um sistema de orientação feito por telemóvel, retirado de [45].

# Capítulo 3

# Análise e especificação

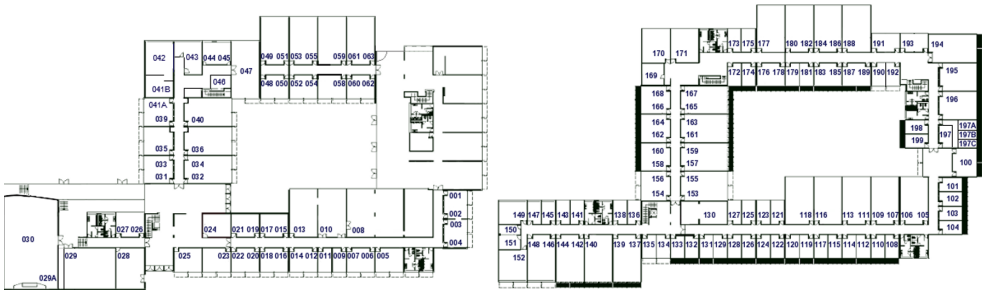
Neste capítulo, serão partilhadas as ideias desenvolvidas ao longo do trabalho efetuado, dividido em diferentes partes. Este projeto foi feito de uma forma sequencial. Até alcançar o objetivo de fornecer uma orientação às pessoas tivemos que passar, primeiramente, pela fase de como representar a planta a um nível computacional, seguindo-se a etapa de investigar quais as melhores métricas, restrições e o melhor algoritmo a utilizar. Por sua vez, estes só se conseguem processar em estruturas devidamente estruturadas e mapeadas. Serão apresentadas partes de pseudocódigo implementado e respetivo output gerado. Explicaremos em detalhe cada uma das opções tomadas em cada categoria, de forma a completarmos o nosso raciocínio.

## 3.1 Representação do edifício computacionalmente

A fim de termos um fio condutor sequencial, para o desenrolar deste nosso projeto, primeiramente tivemos que determinar sobre que características trabalhariamos. A nível do edifício, decidimos então fixarmo-nos no edifício do Departamento de Ciências dos Computadores (DCC), da Faculdade de Ciências da Universidade do Porto (FCUP). Esta decisão advém do facto de ser um edifício, com o qual já estamos ambientados e que contém uma larga banda de especificidades, a serem trabalhadas no nosso projeto. Continuamente, usaremos para a demonstração do protótipo desenvolvido, apenas o piso 0 deste edifício mas utilizaremos, por vezes, a imagem da planta do piso 1, de forma a fomentar e a auxiliar a nossa teoria.

Na representação computacional de um edifício, necessitamos, numa fase inicial, da recolha da sua planta geográfica e desta, para começarmos a estudar todos os pontos possíveis a serem trabalhados. Pontos esses, referentes a compartimentos, corredores, escadas, saídas, espaços amplos interiores ou espaços abertos dentro do edifício e discutir a melhor forma, de passar estes atributos do papel para o sistema. Como podemos visualizar na figura 3.1, o edifício escolhido apresenta dois pisos, compartimentos com diferentes tamanhos, duas saídas, um espaço aberto no interior, dois espaços interiores amplos, os pontos de acesso entre pisos, duas escadas e um elevador.

Analisando cada possibilidade de elaborar este tipo de representação, de forma a facilitar-nos na melhor pesquisa pela similaridade entre a planta real e a virtual, decidimos fazer a transição da planta para o ambiente computacional, utilizando a noção de grafo não



(a) (b)

Figura 3.1: Planta do edifício DCC, a figura 3.1a refere-se ao piso 0, retirado de [2] e 3.1b ao piso 1, de [3].

dirigido, visto que numa aresta, representando esta um troço de um edifício, a pessoa poderá deslocar-se em ambas direções. Tais vantagens associadas à similaridade que um grafo pode trazer, relativamente ao *layout* do edifício, face à respetiva planta após a sua implementação. A possibilidade de num grafo podermos visualizar uma estrutura de uma planta foi o aspeto que considerámos mais relevante para a nossa escolha. Por outro lado, tentámos encontrar outras soluções, não conseguindo chegar a outra que conseguisse fazer o mesmo efeito que a representação através de um grafo. Apesar de um nó de um grafo ser um ponto sem dimensão, conseguimos atribuir, por intermédio de cada nó e suas ligações, posições geográficas de cada nó na planta e respetivas distâncias entre pontos, ao longo do edifício. Este aspeto será importante mais para a frente, na medida que permitirá obter a localização de um foco de incêndio, de uma pessoa a evacuar e, a partir do mapeamento feito no grafo, elaborar a melhor rota até uma saída, escolhendo os melhores nós até ao nó saída.

### 3.1.1 Determinação de cada espaço do edifício

Uma planta de um edifício pode apresentar as várias secções já referidas, de tal modo será necessário conseguir distingui-las e daí poder incluir todos os pontos que vão ser referidos nas próximas subsecções. Através das diferentes regras apresentadas posteriormente, conseguiremos compor a nossa rede do grafo e daí chegar onde pretendemos.

Relativamente à identificação das diferentes partes de um edifício, por vezes visualizá-las através de uma planta é algo que pode carecer de informação, não se percebendo bem a que secção aquele espaço corresponde. Não sendo o objetivo principal desta dissertação, a investigação pela melhor forma de ao visionar uma planta, passá-la para um meio computacional, sabendo distinguir o que é cada espaço, se corresponde a um compartimento ou a um espaço amplo por exemplo, foi um assunto que discutimos, decidindo estabelecer a correspondência para as diferentes zonas da nossa planta de estudo, para continuarmos o nosso trabalho da melhor forma.

A figura 3.2 representa a atribuição que estabelecemos aos espaços do Piso 0, da planta em questão. As zonas sombreadas a laranja correspondem aos espaços amplos e a verde as escadas. Também definimos para os corredores a cor azul e aos compartimentos a cor vermelha. Num trabalho futuro de haver uma plataforma que faça a conversão do papel para um sistema, mediante as regras definidas em baixo, poderá ser uma ideia, anotar o que é cada espaço da planta e daí facilitar a conversão em grafo.

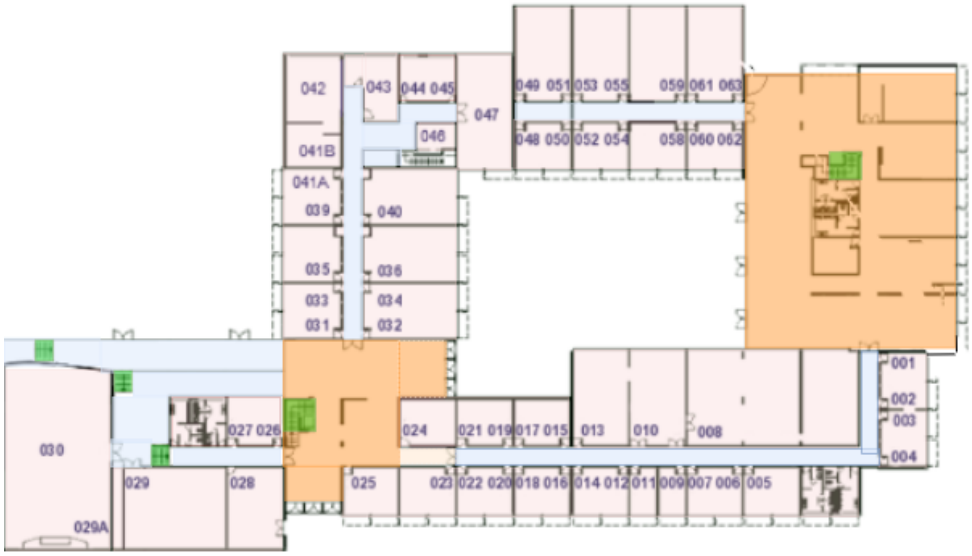



Figura 3.2: Planta exemplo, usada ao longo do nosso trabalho, diferenciando as diferentes secções existentes no Piso 0, através de um sombreado de cores.

### 3.1.2 Espaços amplos no interior do edifício

Normalmente, somos deparados com diversos edifícios, com espaços amplos dentro destes e a forma como os representar computacionalmente é algo que por vezes carece de informação, muito mais quando se trata de o representar, através de um grafo e na possível presença de obstáculos, considerando para este caso obstáculos que possibilitam passagem por ambos os lados. Usaremos para demonstrar as nossas ideias, a parte correspondente à zona do bar, da planta do DCC, como consta na figura 3.3. Consideramos o género de rectângulo que está no meio da figura, a zona do bar, um obstáculo, como também as linhas paralelas nas laterais do espaço e o compartimento no canto superior direito. As figuras 3.4, 3.6, 3.6 são abstrações do espaço amplo que escolhemos para exemplificar as nossas ideias.

Figura 3.3: Espaço amplo do DCC, piso 0, parte referente à zona do bar, considerando este como o obstáculo a ser identificado computacionalmente.

Em primeiro lugar, começamos por aferir que cada entrada/saída deste espaço teria que

ser representada por um nó, pois permitirá definir da melhor forma os caminhos externos a este espaço, no edifício, e delimitá-lo assim num grafo. Pelo facto de não existir nenhum obstáculo, no meio do espaço, consideramos então uma representação como a da figura 3.4, identificando o espaço computacionalmente, com nós em cada entrada/saída deste. Os círculos preenchidos a azul, representam os referidos nós e o número de identificação correspondente a cada um, podendo desta forma determinar a que parte do espaço cada nó corresponde. Atribuímos também as ligações, que mostram a conexão que todos os nós têm entre si, neste caso as diferentes rotas que se pode tomar.

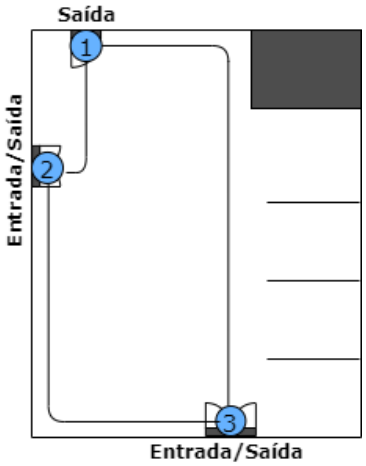



Figura 3.4: Abstração do espaço amplo da zona do bar do DCC, piso 0, mas retirando o obstáculo central e consequente representação computacional do mesmo.

Em segundo lugar e sendo a parte que ocupou a uma maior discussão da nossa parte, como conseguir representar este tipo de secções amplas, na presença de um obstáculo, e consequentemente como representá-lo num grafo, a fim de nos ajudar na fase de escolha do melhor caminho, qual dos lados de passagem do obstáculo será mais aconselhado, surgindo algumas ideias, como as mencionadas abaixo.

Pensámos começar pela tentativa de utilização do Diagrama de Voronoy, para a divisão do espaço. Este corresponde a um conjunto de pontos num plano, relativamente a uma subdivisão desse mesmo plano, em regiões formadas pelos lugares mais próximos a cada um dos pontos [43] . Usando os pontos como os nós de entrada/saída do espaço e a posição centro dos obstáculos, marcando nós auxiliares através das arestas que dividem as secções proporcionadas por estes pontos, não resultaria, pois os nós criados poderiam corresponder, na mesma, a obstáculos e não faríamos a distinção de cada lado do mesmo.

Do diagrama de Voronoy passámos para a triangulação de Delaunay, onde a partir de pelos menos três pontos, se estabelece um triângulo e cada triângulo corresponde uma circunferência [6] . Associando ao nosso estudo, os pontos a utilizar corresponderiam à posição dos sensores no espaço e às entradas/saídas do mesmo, os nós correspondentes a este seriam o centro de cada circunferência. Contudo, chegámos ao mesmo problema do Diagrama de Voronoy, pois esse ponto poderia estar sobre um obstáculo, o que não nos interessaria.

Nesta busca pela melhor solução, pensámos também usar o método *Line Graph*, que transforma as arestas de um grafo em nós e o nós a correspondentes arestas. Porém,

teriamos novamente os mesmo problemas referidos acima, relativamente ao obstáculo, o que não nos levou a conclusão alguma.

Uma outra hipótese colocada foi a possibilidade de, identificado o obstáculo, adicionar nós à volta deste, considerando-os como as possíveis laterais disponíveis a percorrer-se. Todavia, esta hipótese poderia levantar problemas relativamente à forma como seriam feitas as ligações entre cada nó e a posição de cada nó à volta do obstáculo, o que não seria uma precisa representação do espaço, como podemos ver na figura 3.6. Porém, esta situação seria menos esclarecedora na existência de um outro obstáculo, para além do mostrado na figura, entrando numa subjetividade, relativamente às ligações entre os nós.

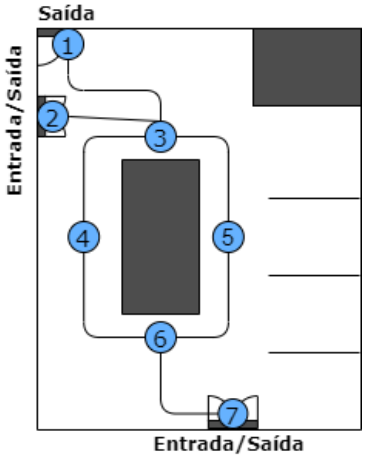



Figura 3.5: Hipótese estudada de representar o espaço amplo com obstáculos, com nós à volta do obstáculo, não conseguindo alcançar uma solução válida.

Por último, pensámos então solucionar este problema através de uma grelha de nós, o que nos pareceu a ideia mais credível e que nos permite definir, com um certo grau de confiança, o espaço. Inicialmente pensamos nesta metodologia a aplicar em todo o edifício, porém iria levar a um excesso enorme de nós, o que ia aumentar a complexidade do sistema e não traria vantagens para outras zonas do edifício. Esta grelha apenas seria referente à representação de um espaço amplo e teria como objetivo especificar a localização de cada obstáculo, e daí orientarmo-nos da melhor forma. Porém, este tipo de atribuição poderá trazer problemas a nível da complexidade do sistema, pois terá que levar a um maior número de nós presentes na rede. Como tal, apresentamos esta nossa ideia, espelhada na figura 3.6. Podemos ver a cor azul os nós de possível passagem pelo espaço, os nós a laranja e tracejado a representarem os obstáculos e a vermelho o fim da área do espaço amplo em estudo. A verde serão os nós da grelha mais próximos de cada saída do espaço amplo, os quais farão ligação entre a grelha e o resto do edifício ou mesmo para o exterior do edifício. A distância de cada aresta da grelha teria o comprimento da largura, da saída mais pequena do espaço.

### 3.1.3 Compartimentos e corredores

Os compartimentos e os corredores são duas zonas que decidimos meter na mesma secção, pois, geralmente, os dois possuem sempre ligação entre ambos numa planta. A disposição dos nós em compartimentos será representada e posicionada no meio destes, de forma a termos uma perspetiva eficaz acerca da mesma. Porém, este posicionamento do nó, no

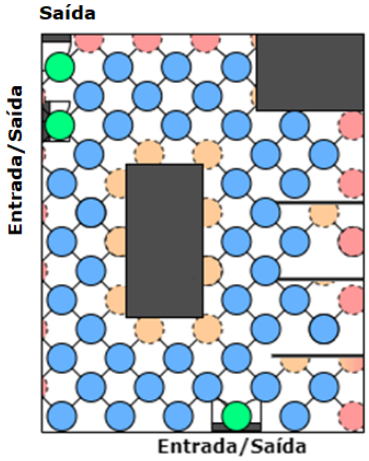



Figura 3.6: Definição de um espaço amplo computacionalmente, com obstáculos, através de uma grelha de nós, disposta por toda a área da secção.

compartimento, pode gerar alguma controvérsia, no caso de, nesse local, existir algum obstáculo, que impossibilita a passagem por esse mesmo nó. Todavia, referimo-nos a um compartimento, na medida geral, por um espaço com área mais reduzida do que um espaço amplo, ao que a presença do mínimo obstáculo que possa existir, não interfirá no caminho a percorrer pela pessoa a evacuar.

Por conseguinte, necessitamos de fazer a ligação do compartimento, ao espaço adjacente a este, em que terá que existir um outro nó, nesse mesmo espaço, a fazer a ligação. Primeiramente, fazendo referência à ligação entre um compartimento e um corredor, o nó representativo do compartimento ligará a outro nó no corredor, estando este posicionado à frente da saída do compartimento e no meio do corredor. A existência dos vários nós a representar o corredor permite-nos precisar, com exatidão, o valor do perigo de cada troço num corredor. Porém, também poderemos ter saídas de compartimentos bastante juntas umas das outras, o que levará a ter vários nós com uma distância bastante diminuta entre eles. Apesar deste aspeto menos positivo, concluímos que seria a melhor forma de realizar esta representação, pois para o nosso objetivo, de sabermos as distâncias entre secções e o correspondente perigo, será a forma mais adequada.

Por outro lado, uma ideia que foi também estudada, a de num corredor haver só um nó auxiliar e ter todas as ligações dos compartimentos ligadas a este, levou-nos a concluir que não seria útil no nosso plano, pois não conseguiríamos obter informações acerca da distância entre compartimentos e dos vários perigos expostos ao longo do corredor, podendo existir um corredor de grande compartimento, não diferenciando os perigos existentes em cada secção, ao longo desse mesmo corredor. A figura 3.7 retrata este mesmo tópico, sendo os nós 1, 2, 3, 4, 5, 6, 7 correspondentes aos compartimentos e o 8 ao corredor.

Um dos aspetos referidos em cima, que remete para o facto de que poderá haver um caminho por entre compartimentos, neste caso, cada compartimento teria o nó identificado no centro do local na mesma, e haveria a ligação entre os nós. Assumimos, no nosso trabalho, que estas passagens entre compartimentos, que podem levar a um melhor caminho, estarão sempre disponíveis para circulação.

Posto estas definições, é importante referir a ligação que existe, quando na saída de um

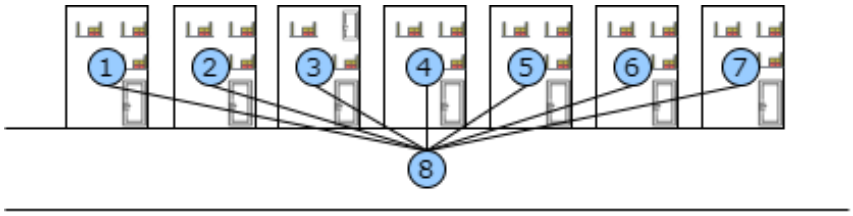



Figura 3.7: Ilustração da ligação entre compartimentos e corredor, no caso de existir apenas um nó a representar o corredor.

compartimento, está um espaço amplo e não um corredor. Como a nossa caracterização de espaço amplo será uma grelha de nós, será feita a ligação do nó do compartimento ao nó mais perto da porta do mesmo e, consequentemente, fazer essa conexão.

Relativamente às ligações que serão representadas de/para corredor, a nível dos compartimentos, já foi esclarecida, mas temos de ter em consideração quando um corredor desagua num espaço amplo. Nesta situação, o nó presente no corredor, com uma distância menor para os nós da grelha do espaço amplo, fará a sua conexão entre o corredor e o espaço.

Com o objetivo de fazer um breve resumo do que foi dito nesta secção, elaborámos a figura 3.8, representando a ligação de compartimentos aos corredores e a transição entre os mesmos. A relação existente entre este tema e os espaços amplos, representámos de forma abstrata na grelha de nós com a cor verde água e fizemos então a transição do corredor para o espaço amplo e de um compartimento para o espaço amplo. Os nós identificados pelos números 2, 4, 6, 8, 10, 12, 14, 16 correspondem aos nós do corredor e os 1, 3, 5 ,7, 9, 11, 13, 15 e 17, aos compartimentos. O nó representado a verde fluorescente, será o nó da grelha que fará a ligação com o nó mais próximo do corredor, visto que é dos nós da grelha mais juntos àquela saída e posicionado o mais próximo do meio da largura do corredor.

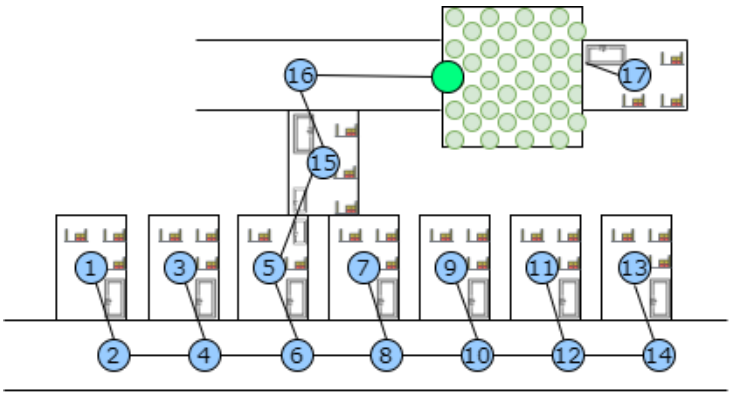



Figura 3.8: Elaboração de uma forma abstrata, da ligação entre dois compartimentos, entre compartimento e corredor ou com um espaço amplo. Também é mostrada a ligação entre corredor e espaço amplo.

### 3.1.4 Vias verticais

Relativamente às vias verticais num edifício, teremos em consideração rampas ou escadas que ligam cada piso um ao outro. Neste caso, sendo estas duas hipóteses para que

uma pessoa se possa deslocar entre pisos, o que terá sempre que existir, visto que não consideramos os elevadores como meio de deslocação e admitindo que estas não apresentam saídas para o exterior, assume-se a existência de um nó, no início de cada escada ou rampa a representar cada piso, tendo a aresta que liga estes nós, a importância de simbolizar a mudança de piso. No caso de existirem vias verticais no exterior do edifício, este será tratado na mesma forma do que as interiores, existindo um nó referente à saída para o exterior, fazendo a transição entre o interior e o exterior. Através deste formato, a rota a tomar por escadas auxiliares, por exemplo, poderá apresentar uma menor distância e assim ajudar no objetivo proposto.

A figura 3.9 remete para as duas escadas que existem no objeto de estudo, ligando o piso 0 e o 1 do DCC, estando localizadas uma em cada extremidade do edifício. Cada secção de escadas, em diferente locais do edifício, está sombreada a cor diferente, umas a amarelo e outras a vermelho. É de notar, que devido à resolução da imagem proporciona-nos a visualização das escadas, na planta, com um tamanho pequeno, sendo que a zona a sombreado acaba por englobar outras zonas do edifício. Neste caso, os nós numerados a 1 e 2 correspondem ao início das escadas no piso 0 e os nós identificados com 3 e 4 correspondem ao piso 1, as duas ligações representam então as ligações entre pisos, respetivamente.

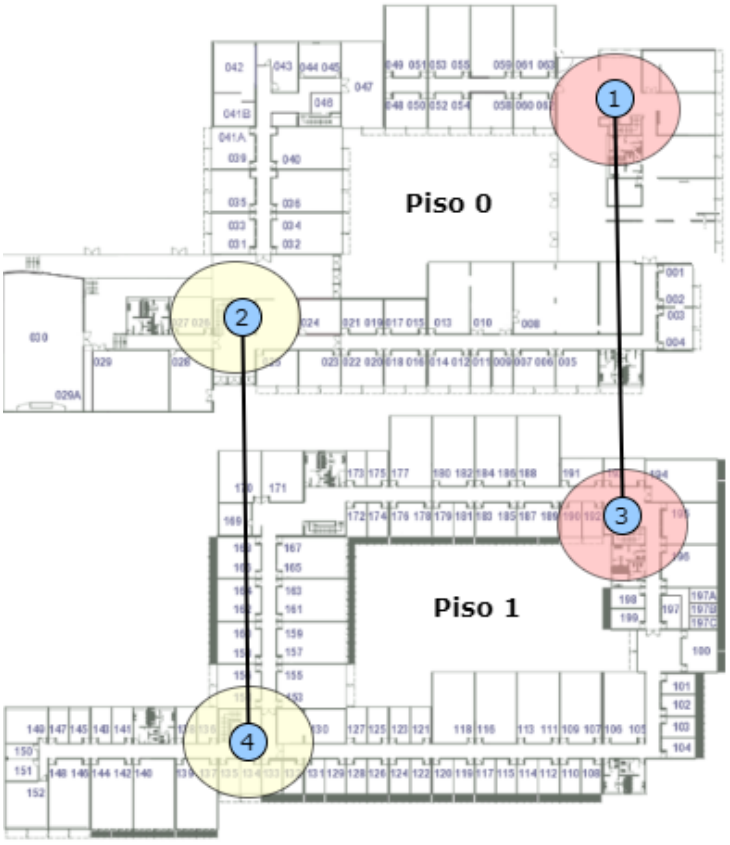



Figura 3.9: Identificação dos nós a representar cada sequência de escadas entre pisos. As zonas a sombreado remetem para o mesmo lanço de escadas, fazendo a ligação entre os dois pisos.

### 3.1.5 Elevadores

Neste contexto de evacuação de pessoas de edifícios, falar de elevadores como um pedaço de caminho a poder ser validado, é um aspeto bastante particular, pois engloba vários assuntos relacionados. Como este assunto faria sentido ser tratado num estudo de edifícios de uma altura enorme e como não é o nosso principal objetivo, assumiremos que não será uma rota possível. Pois, para conseguirmos validar esta possibilidade, teríamos que tratar de aspetos, relativamente à energia dispendida do elevador e o controlo do mesmo, sendo que em alturas de incêndio, são fatores que podem apresentar várias falhas e modificações de resultados. Aspetos relacionados com as entradas de fumos que possam perturbar a saúde das pessoas, e também podendo relacionar com os aspetos comportamentais das mesmas, em casos de pânico, o que poderia levar ao mau funcionamento do elevador [30], são outros aspetos que nos levaram a não incluir o funcionamento deste no nosso projeto.

Contudo, uma possibilidade que nos surgiu, enquanto discutíamos acerca deste assunto, foi a hipótese de utilizar o elevador para pessoas, em casos especiais e em edifícios com um número elevado de pisos, faseando o movimento do elevador. Ou seja, para haver uma precaução face aos problemas referidos em cima, poderíamos impor no sistema, a possibilidade de o elevador descer, apenas até um determinado piso e daí as pessoas fazerem a pé o restante percurso. Nesta situação, cada piso onde o elevador parasse, teria que ter um nó a fazer essa identificação. Também se poderia adaptar o uso de elevador, no caso de a deslocação entre pisos por escadas ou rampas estar afetada e usar este como uma última opção de recurso.

### 3.1.6 Janelas e portas para o exterior

Este é um dos aspetos relevantes a ter em consideração. Ao longo de um percurso, caso não consideremos os elevadores como uma possibilidade, ou não tenhamos possibilidade de sair do edifício, é preciso ter estes dois factores em conta porque, ao fim ao cabo, serão a nossa última hipótese de sobrevivência.

No caso de podemos utilizar a janela como meio de evacuação, teremos que contar com o risco associado à queda no solo, o que levará a imensas investigações e análises, a serem efetuadas noutro caso de estudo. A distância até ao solo, a composição do local de embate, o peso, o tamanho, a posição de queda da pessoa são parâmetros que levam a imensa teoria a ser estudada e fogem um pouco ao objetivo do nosso trabalho. Todavia, pensámos no proposto e apresentámos um método vantajoso na nossa opinião, que seria descer até ao menor piso possível e daí usar esta como meio de saída. As janelas que tivessem um tamanho adequado para a saída de um indivíduo, apresentariam um nó referente a uma provável saída do edifício, mas com um risco elevado e apenas utilizado nos casos já referidos.

Por outro lado, havendo uma porta que liga ao exterior do edifício, este pode ser um local importante também, numa eventual restrição às saídas principais, o que pode permitir levar as pessoas para um local, onde poderão permanecer à espera de ajuda. Este pode estar também identificado como um nó de saída do edifício, apresentando outros valores até chegar a este.

### 3.1.7 Espaços exteriores dentro do edifício

Muitos edifícios e nomeadamente no edifício em estudo, apresentam secções de passagem ao ar livre, dentro do mesmo. Estas podem ter passagem de um lado ao outro do edifício ou apenas espaços abertos, o que nos pode ajudar imenso na deslocação, precisando ter em conta esta possível existência que poderá ser um dos caminhos menos perigosos a efetuar durante um incêndio.

No exemplo em estudo, o espaço aberto liga uma ponta à outra do DCC, o que nos permite fazer desse, um possível caminho de ligação na nossa rota a tomar. Para locais como este, onde a propagação de incêndio é igual, tendo ou não tendo obstáculos no meio, não necessitamos de incrementar nós auxiliares, o que basta fazer uma ligação direta entre as portas do espaço. Como tal, basta considerar um nó em cada entrada/saída e fazer a respetiva ligação.

Porém, se na eventualidade de apenas existir o espaço com uma entrada para o mesmo, sendo também a única saída, adiciona-se um único nó, como demonstrado na subsecção 3.1.6, relacionada com o assunto das portas para o exterior, de forma a representar este espaço, podendo ser necessário utilizá-lo em caso de refúgio ou de escapatória.

Na figura 3.10 conseguimos perceber a existência do espaço em estudo na planta do DCC, sendo uma zona de passagem ao ar livre, representado na planta, com a existência de entradas/saídas. Apresentamos na imagem a adição dos nós a representá-lo, considerando apenas a existência de duas portas abertas nas extremidades do local. Convém referir, como dito na figura 3.9, a zona a sombreada acaba com englobar outras zonas do edifício querendo apenas, neste caso, fazer referência ao espaço onde estão presente os dois nós.

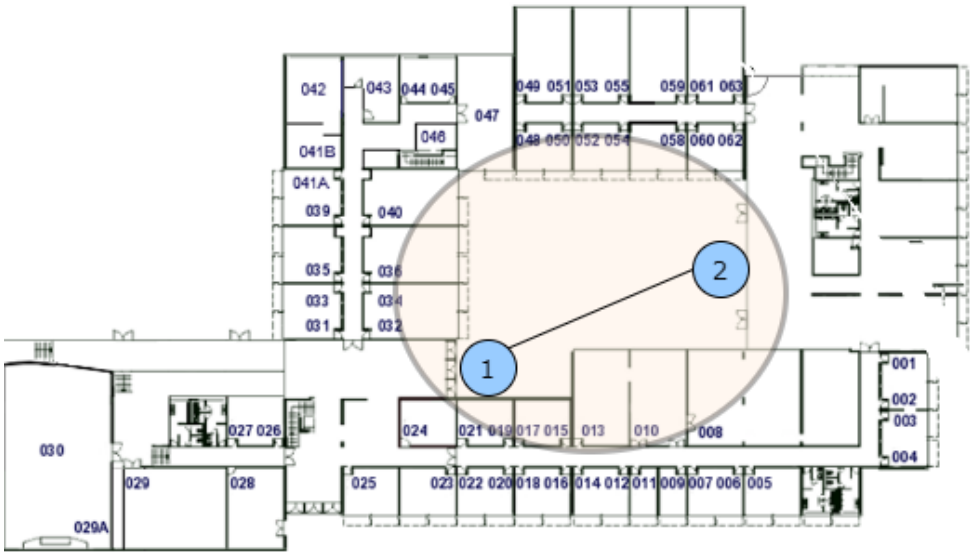



Figura 3.10: Representação do espaço exterior, dentro do edifício em estudo, com passagem entre pontas do edifício. Zona a sombreado corresponde a esse espaço, com a ligação entre dois nós, das duas entradas/saídas possíveis.

## 3.2 Algoritmo, métricas, restrições desenvolvidas

Após estabelecermos algumas regras, na primeira fase do nosso projeto, de como realizar a representação computacional de uma planta, passamos então, ao momento de começarmos a pensar no melhor algoritmo a utilizar, métricas e restrições a este. Em consonância com o propósito do nosso objetivo, procurar a melhor implementação para descobrir qual o caminho mais curto, consoante determinadas especificidades.

### 3.2.1 Algoritmo

Como mostrado no capítulo 2, mais propriamente na subsecção 2.4.4, existem já algoritmos que permitem fazer a busca pelo caminho mais curto entre dois pontos. Contudo, o nosso principal objetivo, é o de implementar um algoritmo que seja de acordo com as nossas ideias e que permita realizar todo o processamento, de uma forma rápida e eficaz. Posto isto, analisando os algoritmos presentes, decidimos começar a preencher a nossa base pelos fundamentos do melhor algoritmo encontrado e começando aí o nosso processo de busca.

Quando falamos de rapidez de um algoritmo, podemos olhar à complexidade que este apresenta, aquando da sua execução. Existindo uma maior complexidade, existirá um maior tempo de processamento. Na escolha de um algoritmo, também é necessário estar atento às suas estruturas de dados, pois no caso destas serem bastante complexas e a sua rapidez não fazer grande diferença perantes outros algoritmos, optar por uma estrutura de dados mais intuitiva e simplista será a melhor opção. Outro dos critérios importantes, também será a confiabilidade que esse algoritmo dá, consequentemente com a sua eficácia. Sendo que procuramos o melhor caminho, a partir de apenas uma fonte, percebemos que excluirmos o algoritmo Floyd-Warshall das nossas opções, seria o mais adequado, visto que este realiza a busca do melhor caminho para todos os pares de vértices no grafo, apresentando este uma complexidade grande, no pior caso de $O(V^3)$, sendo V o número de vértices do grafo. O algoritmo A* funciona de acordo com um custo presente em cada aresta, associado a uma heurística, o que no continuar do nosso trabalho não fará muito sentido, pois não fixaremos nenhum custo às arestas, através de nenhuma heurística. Comparando os último dois algoritmos apresentados no estado da arte, Bellman-Ford e o AD, o primeiro seria bastante útil no caso, de na nossa rede, existirem valores negativos, o que não irá acontecer, e também pelo facto de a sua complexidade ser mais elevada que a do AD no nosso caso, $O(V * E)$ para o primeiro e $O((E + V)logV)$ para o segundo, sendo V o número de vértices e E o número de arestas. Optamos então por seguir o trabalho, tomando como base o AD, visto que é um algoritmo já com provas de eficácia reconhecidas e que apresenta um tempo de execução bastante rápido, o que será importante na determinação da melhor rota, em condições de urgência [13] .

### 3.2.2 Métricas e restrições

Após uma investigação e discutirmos acerca da forma base com que avançávamos no tópico do algoritmo, seguiu-se de seguida a procura pelas métricas a utilizar, possíveis restrições a haver, adaptando este caso ao algoritmo pretendido.

O nosso caso de estudo engloba lidar com a rapidez na saída, mas também com a sua segurança. Como tal, definimos utilizar duas métricas e circundar o problema à volta delas, usando estas, como fio condutor no desenvolvimento do nosso algoritmo. Não podendo ter em conta, todas as métricas possíveis, como por exemplo, valores da velocidade da propagação do incêndio, velocidade de movimento das pessoas, capacidade de pessoas, em cada zona do edifício, sendo estas algumas métricas possíveis neste tipo de investigação, porém cada uma apresenta outros estudos que fogem ao nosso assunto. Contudo, chegámos a discutir algumas ideias possíveis para estas métricas complexas. Pegando no caso da velocidade de movimento da pessoa, o sistema poderia analisar um intervalo de tempo percorrido pela mesma e daí apresentar um resultado para a velocidade. Todavia, este não conseguiria ser altamente preciso, uma vez que a velocidade de uma pessoa, devido aos problemas encontrados durante um incêndio, poderá sofrer alterações. Por outro lado, conseguindo saber a velocidade das pessoas e as suas respetivas localizações, conseguiríamos desenvolver o algoritmo, de maneira a que não existisse nenhum congestionamento, em nenhuma parte do edifício. Após esta secção, de averiguação acerca de métricas que levantavam imensas questões, decidimos então jogar com o valor da distância e generalizar um nível de perigo.

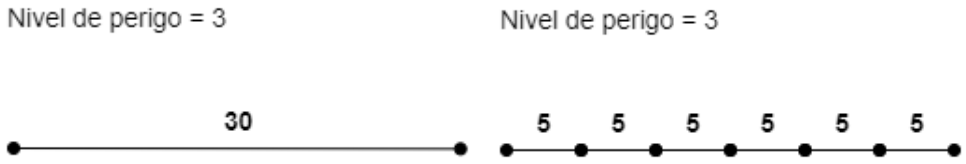
Começando pela escolha da distância como uma métrica, tal como o nome indica, esta será a distância física de um ponto ao outro do edifício. Neste caso, o mapeamento será realizado nas arestas entre dois nós, sendo que cada nó representa uma certa posição geográfica no edifício. O valor da aresta corresponderá ao intervalo entre esses dois nós. Esta métrica apresenta uma grande importância, visto que como queremos sempre reduzir o valor percorrido ao longo do edifício, será necessário ter em consideração a distância, pois será a nossa grande referência na distância da pessoa a percorrer até uma saída.

Mapear o grafo apenas com a distância, neste contexto, não serviria, pois não iríamos estar a incluir o risco que poderia existir em qualquer parte do edifício, resultando a busca apenas no caminho mais rápido, mas não no mais seguro. Consequentemente, decidimos incluir uma métrica de perigo. Esta métrica será representada através de um valor de perigo e por sua vez, este valor será baseado num respetivo nível de perigo. Este valor de perigo, basear-se-á em outros estudos que englobem todos os riscos já referidos e com base em mais ou menos riscos, existentes em cada parte do percurso, esse nível de perigo aumentará ou diminuirá. No nosso trabalho decidimos não adicionar a parte da pesquisa relativa ao valor de perigo em cada troço, mediante certos riscos, visto que seria algo demorado e material específico para outra investigação.

Após a escolha destas duas métricas, sobre qual o algoritmo que irá ser processado, procedemos então à busca por uma ideia de como as juntar e de fazer o relaxamento de cada aresta, da forma mais eficaz possível. Começamos apenas por utilizar a métrica do máximo de cada nível de perigo num caminho. Nesta linha de pensamento, o suposto seria escolher sempre a melhor aresta, na resolução do algoritmo, com base na aresta que no caminho até chegar a ela, apresentasse em cada nível de perigo, o valor mais baixo de ocorrências em cada uma. O critério de desempate, seria começar por ver o nível de ocorrências do maior nível de perigo, até ao menos perigoso. Contudo, não conseguimos validar esta ideia, pois uma aresta com um grande tamanho de nível 5, tem apenas uma ocorrência, mas uma outra, mais curta, terá na mesma uma ocorrência.

A figura 3.11 é um exemplo para esclarecer o porquê desta ideia não resultar. Assumindo que ambos os troços têm de nível de perigo 3, um caminho com a aresta de 3.11a apresenta

no número de ocorrências do nível de perigo 3, apenas uma, enquanto que a outra, 3.11b , visto que está dividida em 6 partes, já vai aparecer 6. Contudo, na teoria esses dois troços têm distâncias iguais, o que não resultaria numa boa escolha para o nosso melhor caminho a escolher.




(a) (b)

Figura 3.11: Exemplo de dois caminhos possíveis num grafo, onde em 3.11a está representada uma aresta de distância 30 e nível de perigo 3, e em 3.11b, seis arestas com distância 5, cada uma e todas com nível 3. Mostra-se aqui, que a ideia de ir por uma métrica, que fizesse escolhas através do número de vezes que um risco aparece num caminho, não é válida.

A seguinte ideia, remete para a teoria de se fazer *pruning* numa rede. *Pruning* consiste na eliminação de todas arestas com valores não necessários para o processamento do sistema e, daí fazer o processamento com as arestas resultantes [15] . A aplicação que fizemos desta ideia ao nosso trabalho, remete para o objetivo de numa rede mapeada com diferentes níveis de perigo, começarmos a correr o nosso algoritmo, restringindo de forma a utilizar o primeiro nível de perigo, e visualizar se existe algum caminho até uma saída. Na inexistência de um caminho, aumenta-se um nível de perigo e corre-se utilizando arestas com esses dois valores apenas, e continuando assim até ao último nível de perigo a considerar, consequentemente o mais perigoso.

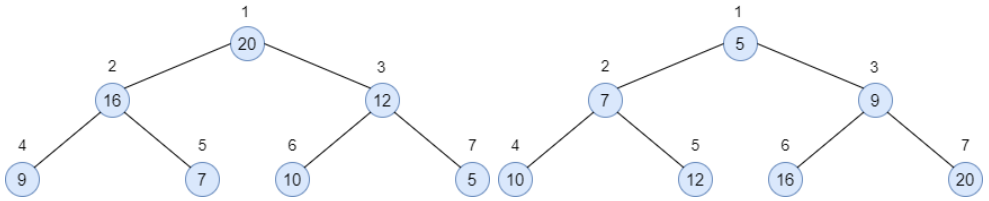
Estas duas ideias foram importantes no complemento da ideia final. Pegando na primeira ideia, pensamos que, se não conseguimos ver o máximo das ocorrências de cada perigo, em cada troço, conseguiríamos sim, ver a distância percorrida em cada caminho, para cada valor de perigo. Por um lado, este conjunto de distâncias, permite-nos reduzi-la e, em cada risco, favorecer o aspeto da rapidez e, ao mesmo tempo, a questão da segurança. Pensando nesta possível ideia das distâncias em cada risco, podemos começar por incluir o pruning neste processo e estaríamos a otimizar o problema, o que nos será muito vantajoso no processamento do algoritmo e na eficácia de valores que este pode apresentar.

### 3.2.3 Arquitetura do algoritmo

Depois da escolha do melhor algoritmo e das melhores métricas a utilizar, tivemos que proceder ao desenho do algoritmo e à sua estruturação. As etapas base que o AD executa foram já explicadas na subsecção 2.4.4. Porém, desenvolvemos o nosso algoritmo, baseando-nos no AD, juntamente com as nossas ideias.

De forma a elaborarmos o nosso algoritmo, de uma maneira que seja confortável para nós e que apresente um bom grau de fiabilidade, a nível do tempo de processamento, decidimos então implementá-lo com o auxílio de uma estrutura de dados heap binária. Este mecanismo corresponde a um array de objetos que, basicamente, é visto como uma árvore binária. A árvore é criada, de forma que para um pai, existam sempre dois filhos, permitindo que exista

uma raiz com apenas um elemento e que possamos fazer a pesquisa por um nó, através dos seus filhos, da esquerda ou direita. Podem existir dois géneros de heap, heap mínima ou heap máxima, havendo sempre nas duas, propriedades de escolha de qual o nó a ficar na raiz. Na heap mínima, a escolha pelo elemento da raiz será o que tiver o valor da propriedade de escolha menor e assim continuamente para o resto da heap, ou seja, ao longo da árvore, o valor do pai será sempre menor que o valor dos filhos. Na heap máxima, acontece o contrário da heap mínima, o valor do pai será sempre maior do que o dos filhos [13] . A figura 3.12 mostra o que acabámos de dizer de uma forma ilustrativa, sendo o identificador do nó, o número fora de cada círculo azul, e o número dentro dos círculos, o valor dos respetivos nós. Sendo que em 3.12a o nó pai de cada nó terá sempre um valor superior aos restantes filhos e em 3.12b representa-se o contrário.




(a) (b)

Figura 3.12: Representação exemplificativa do conceito de heap máxima em 3.12a e heap mínima em 3.12b. Adaptado de [13].

Representamos no algoritmo 1, em 1, o pseudocódigo do algoritmo implementado no nosso sistema desenvolvido. Como pretendemos trabalhar com a distância percorrida em cada risco, sendo esta propriedade, o nosso critério de escolha do nó a tirar no algoritmo, decidimos trabalhar com arrays, em que cada um deles corresponderia a um certo nível de perigo e que cada posição destes, representaria um nó do grafo e o seu valor a distância percorrida naquele nível de perigo. Na linha 2 serão inicializadas todas as posições dos arrays, mediante o número de nós existentes no grafo, com o valor de $\infty$ e o valor da origem em cada array será inicializado com 0, sendo a origem. Continuamente, na linha 3, a heap mínima a utilizar será também inicializada. Para o nosso sistema, decidimos considerar apenas 4 níveis de perigo, o que iremos trabalhar apenas com 4 arrays de distância em cada perigo.

Após as duas primeiras linhas, teremos na linha 4 uma variável *flag* que fará a implementação da ideia do *pruning* já referida. Esta é importante na medida em que controla se é necessário aumentar o limite do nível de perigo, no caso da inexistência de caminho até uma saída.

Visualizando o algoritmo, as linhas compreendidas entre 5-24, será a parte onde a aplicação de todas as métricas serão efetuadas e se desenrola a procura pelo melhor caminho até a uma saída, para cada nó do grafo, enquanto a heap não estiver sem nós. Na linha 6 o método EXTRACT_MIN pertence à heap e será retirado o nó que estiver na raiz. Considerando apenas 4 níveis no nosso sistema, o nó da raiz será o que possuir no seu melhor caminho até ele, a menor distância em arestas com risco 4, comparado com os outros. No caso de ter outro nó com a mesma distância, para arestas do risco 4 até ele, será feito o critério de desempate com a distância em arestas de risco 3. Seguindo esta lógica até ao primeiro nível, obtemos então o nó presente na raiz. Da linha 7-11, vemos se o valor da posição do nó que retirámos, nos quatro arrays de cada risco, está ainda com uma distância

**Algorithm 1** *bestPath*: Algoritmo para obtenção da melhor rota de evacuação de cada nó, do grafo $G$, para cada saída do edifício.

```
1: procedure BEST_PATH(G, s, f)
2:     INITIALIZE_NODES(G,s)
3:     Q ←INITIALIZE_HEAPMIN(G,s)
4:     flag ← 0
5:     while Q.IS_EMPTY ≠ true do
6:         v ← EXTRACT_MIN(Q)
7:         if distance_each_risk of v = ∞ then
8:             flag ← 1
9:             INCREMENTAL(danger_value)
10:            break
11:        end if
12:        if v ≠ supernode then
13:            CHANGE_DIRECTIONS(v,pai[v])
14:        end if
15:        for all edge u of G.Adj[v] do
16:            w ←VERTEX_ADJ(u)
17:            risk ←RISK_VALUE(u)
18:            valor_dist ←DIST_VALUE(u)
19:            if risk ≤ danger_level then
20:                RELAX(w,v,risk,valor_dist)
21:            end if
22:        end for
23:    end while
24:    if flag = 0 then
25:        SEE_ALL_DIRECTIONS
26:    end if
27:    if flag = 1 then
28:        INCREMENTAL_DANGER_VALUE
29:        BEST_PATH
30:    end if
31: end procedure
```

de $\infty$ . Caso essa situação se verifique, percebemos que não haverá caminho para o limite de perigo a utilizar, sendo aumentado esse mesmo, originando um aumento nas arestas a procurar, parando o ciclo e colocando o valor da *flag* a 1.

No caso de passar a condição da linha 7 e o nó a extrair, não corresponder ao nó final, será atribuída a direção entre o nó que se retirou e o seu nó pai. Criámos um array, neste caso com o nome pai, cuja posição corresponde a um nó e cujo valor corresponde ao seu nó pai. Concluímos o ciclo com as linhas 16-24. Nestas, está presente um ciclo, no qual são visualizadas todas as ligações vizinhas do nó, que extraímos da raiz da heap. Nas linhas 17, 18, 19, procuramos pelo nó vizinho, o valor de risco da aresta e o valor da distância daquela aresta, respetivamente. Na condição presente entre as linhas 20-22, será novamente feita a condição de *pruning* que decidimos utilizar, caso o risco da aresta seja maior que o valor do limite de risco aceitável para cada aresta, não fará o relaxamento dos valores. Caso contrário, procederemos ao relaxamento do nó. No nosso caso, este relaxamento, será feito considerando o risco associado à aresta e à sua distância. Os novos valores da distância em cada risco, do nó vizinho, alteram-se para os do nó que extraímos na linha 6, comparando as distâncias a partir do último nível de risco, este apresenta uma distância superior ao do nó

extraído. A nossa ideia pretende sempre preferir que exista menor distância em níveis de perigo mais alto e muita em níveis de risco mais baixo, do que o contrário.

Nas linhas 24-30 será tida em condição a *flag* já identificada anteriormente. Caso esta seja igual a 1, aumentamos um nível no limite de nível de perigo, a pesquisar em cada aresta e voltamos a correr novamente o algoritmo. Porém, se for igual a 0, ou seja, encontrou um caminho, será feita uma revisão se todas as arestas do grafo, para ver se ficaram todas com direção. Este é um aspeto importante a tratar, visto que após o processamento do AD existem arestas, entre nós, que não apresentam direção, sendo que no nosso caso, precisamos que todas as arestas tenham direção, pois podemos ter várias pessoas espalhadas pelo edifício, sendo necessário que todo o grafo seja mapeado. Para este caso específico, decidimos então na linha 25 visualizar a condição se todos os nós do grafo têm direção para cada vizinho. No caso de não existir, serão criados nós auxiliares, no meio dessas arestas e segue-se novamente o processamento do nosso algoritmo com os nós auxiliares. No final, eliminam-se esses nós e atribui-se a direção entre nós, consoante a direção feita no nó auxiliar.

No conjunto de subfiguras da figura 3.13 visualizamos a ideia que foi apresentada anteriormente. Em 3.13a foi criado um grafo para explicar o que foi dito, apresentando a verde o nível de perigo da aresta, o número ao lado da aresta a preto, a distância da aresta e dentro de cada círculo, um número a identificar o respetivo nó. Após a resolução do nosso algoritmo, atribuímos as direções de cada aresta como visto em 3.13b. Não tendo todas as arestas direção, procedemos então ao recurso de nós auxiliares e à execução do algoritmo novamente, dando como resultado a sub figura 3.13c. Continuamente, faz-se a associação dos nós auxiliares, aos nós principais do grafo e as suas correspondentes direções, resultando na apresentação da sub figura 3.13d.

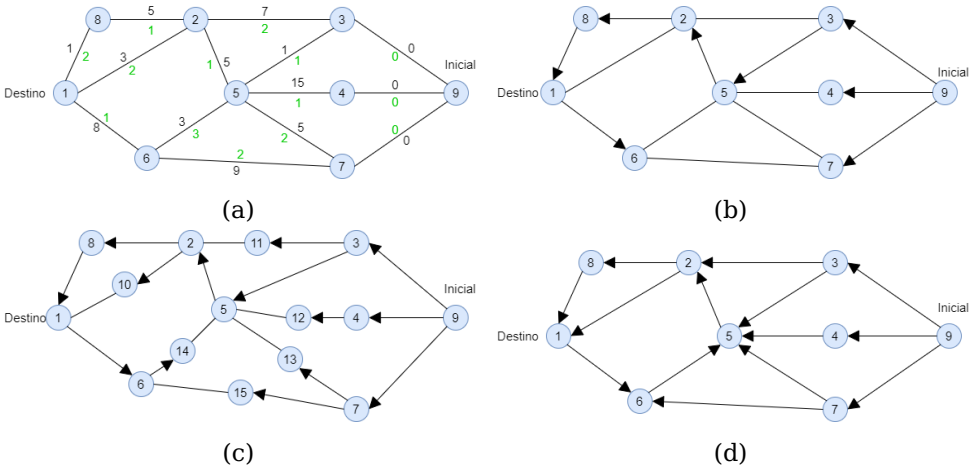



Figura 3.13: Ilustração exemplo, da forma a utilizar na resolução do problema de arestas sem direção. A solução é a inserção de nós auxiliares e proceder novamente à execução do algoritmo.

Para além destas ideias já relatadas, começámos a pensar como seria a melhor forma de agrupar todas as saídas, de forma a correr o algoritmo, tendo um nó fonte e um nó destino. Assim, decidimos implementar um super nó. Este nó representará todas as saídas, ou seja, fará ligação com cada saída de forma direcionada. As propriedades da aresta terão distância 0 e o menor nível de perigo, porque desta maneira, conseguimos continuar com a fiabilidade da rede, visto que é um nó auxiliar, que ajuda na escolha do caminho até à

melhor saída apenas. O supernó também nos permite realizar outro pormenor que decidimos implementar, o processamento por inversão. Queremos falar acerca de inversão, na medida em que necessitamos de ter caminho para uma saída, a partir de qualquer nó da rede. Posto isto calcularemos o algoritmo, atribuindo ao nó inicial o super nó e ao nó final, um qualquer nó do grafo, de forma a termos todos os nós com o caminho para uma determinada saída. Na ilustração representada em 3.14 está um exemplo de um grafo criado com três saídas, consequentemente um nó identificado como super nó, com as arestas que ligam este a cada saída com uma única direção, estando direcionadas para cada saída.

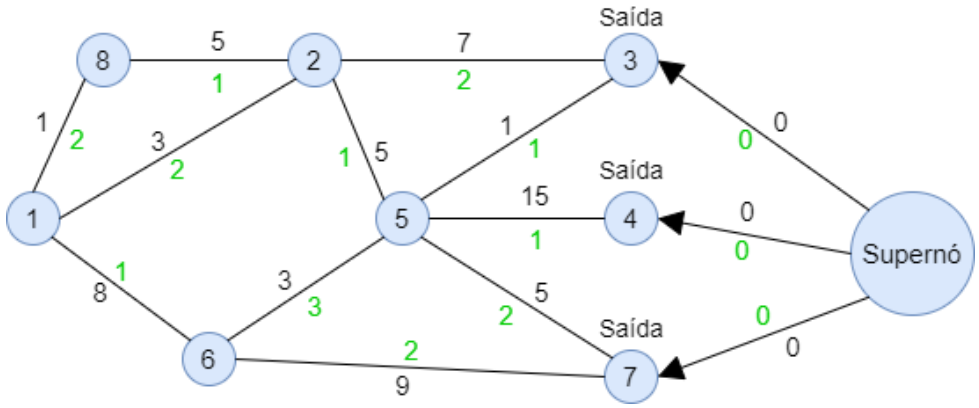



Figura 3.14: Ilustração exemplificativa da inclusão de um super nó num grafo e a direção das suas arestas no mesmo, com os respetivos valores conforme o nosso projeto.

# Capítulo 4

# Implementação do sistema

Após pensarmos e discutirmos em novas ideias, alternativas para o problema que queremos ajudar a resolver, passamos à parte de colocar a teoria em prática e aplicarmos a um sistema. Neste capítulo, iremos apresentar as nossas ideias, de forma a mostrar, como forma poderíamos implementá-las, através de um sistema. Nele serão mostrados parâmetros referentes à parte de *back-end* (BE), *front-end* (FE), tipos de arquitetura de software a utilizar, base de dados e todos os dados inseridos no nosso objeto de estudo, necessários para fomentar as nossas ideias na prática.

Antes de iniciar a discussão acerca de pormenores, mecanismos e *frameworks* a usar, foi importante preencher a planta, atribuindo as definições que determinamos na secção 3.1. Como tal, na figura 4.1 estão representados todos os nós criados no nosso trabalho a vermelho, e todas as arestas entre nós a verde.

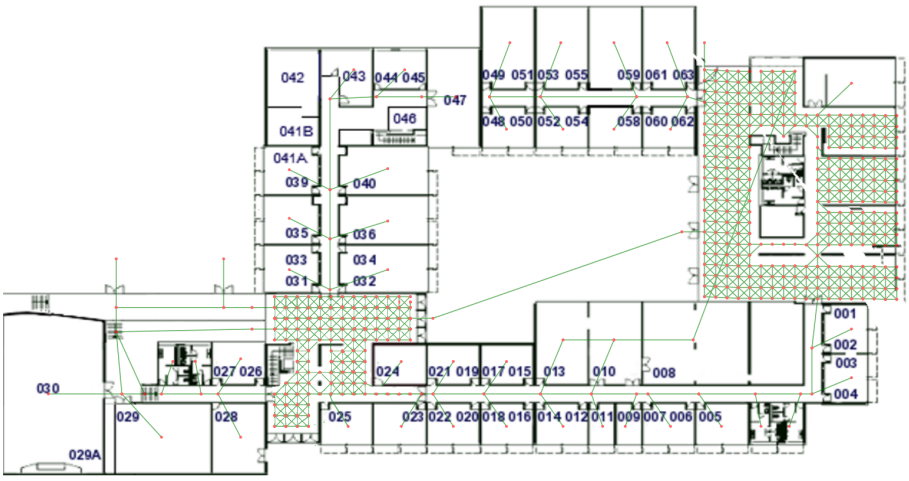



Figura 4.1: Representação dos nós criados na planta em estudo, mediante as regras definidas na secção 3.1.

Por outro lado, também tivemos que ter em conta a respetiva disposição dos sensores, que tipos de sensor a utilizar e que valores nos fornecem estes. Tendo referido anteriormente estes três parâmetros foram efetuados de uma forma um pouco arbitrária, visto que não conseguimos ter acesso a sensores em específico. Como tal, decidimos trabalhar apenas com sensores de incêndio e de fumo, abrangendo cada um deles, uma zona à sua volta por nossa

escolha. Consideramos que um sensor de fumo nos daria valores binários, respetivamente 1, caso houvesse fumo no local e 0, caso estivesse tudo normalizado. Relativamente aos sensores de temperatura, estabelecemos três tipos de intervalos de temperatura, que em cada um deles fizesse o sensor enviar informação para o servidor, a avisar das mudanças dessa mesma temperatura. Atribuímos estes dois tipos de sensor ao nosso trabalho, visto que se trata de uma simulação e que são dois tipos de sensores que certamente irão ser usados no futuro, no interior de um edifício. A posição que estes ocupam e as zonas que definimos que controlariam, estão representadas na figura 4.2. Definimos de forma arbitrária que os compartimentos teriam sensores de fumo, estando identificados com F e os de temperatura correspondendo aos corredores e espaços amplos a T. Cada sensor terá responsabilidade por mapear arestas, num raio definido por nós.

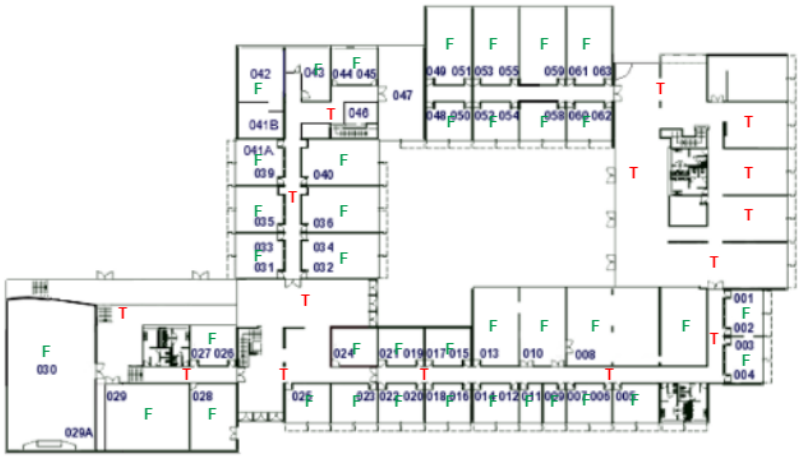


Figura 4.2: Representação da posição escolhida para cada sensor na planta, concretamente sensores de fumo e temperatura.

## 4.1 Arquiteturas do sistema e software

Um aspeto importante que tivemos de decidir foi acerca de como os módulos do nosso sistema interagiam uns com os outros. Como seriam feitas as trocas de informações, o seu processamento, o armazenamento, são aspetos a decidir antes de começarmos a aplicar o trabalho na prática propriamente.

Como visto em 2.2 poderemos proceder à estruturação do nosso sistema através de várias opções. Porém, decidimos elaborará-lo, de forma centralizada, ou seja, o servidor recebe informação, trata-a e fornece as informações. Neste nosso caso de trabalho será o mais vantajoso, dado que terá sempre em sua posse a composição do grafo correspondente à planta, recebe a informação vinda dos sensores, aumenta ou diminui o nível de perigo presente em cada aresta da rede, corre o algoritmo e envia o melhor caminho para cada pessoa a evacuar. A fase de chegada do caminho mais curto, para o utilizador, será feita diretamente através de uma interface desenvolvida para cada dispositivo móvel, que pertence à pessoa a evacuar.

Decidida a centralização do nosso sistema, como arquiteturar as transferências de

informação será um dos pontos fulcrais do sistema, pois trata da interação entre as suas componentes, mexe com estruturas de dados, processamentos de dados, armazenamento, daí a sua importância.[28] O objetivo foi encaixar arquiteturas de software, que nos permitissem garantir principalmente rapidez nas trocas de informação e que nos garantissem fiabilidade no envio e receção da mesma.

A comunicação entre aplicações, no nosso sistema, foi pensada seguindo o conceito de *Web Service* (WS). Considera-se este como uma peça de software disponível na internet, tendo uma coleção de protocolos e padrões para a troca de dados entre aplicações ou sistemas. De referir, que o principal objetivo de uma WS é fornecer um serviço, não propriamente uma interface para o utilizador. Os protocolos mais comuns a serem utilizados nos dias de hoje são o *Simple Object Access Protocol* (SOAP) e *Representational State Transfer* (REST). Tanto SOAP como REST, providenciam um caminho para comunicar entre aplicações correndo em diferentes sistemas operativos, com diferentes tecnologias e linguagens de programação, seguindo assim o conceito de WS em que estão inseridos. [5]

A arquitetura SOAP pode trabalhar com vários protocolos de Internet, como HTTP, sendo este o protocolo base de comunicações de dados, entre sistemas, na Internet. SOAP apenas utiliza o formato XML no formato das suas mensagens. Basicamente, uma mensagem sua contém um elemento envelope, digamos assim, que identifica o ficheiro XML como uma mensagem SOAP. Também apresenta a informação de um cabeçalho e corpo de mensagem contendo a informação de pedido e de resposta, podendo ter mais campos, não sendo estes obrigatórios. Porém, devido a tratar apenas de ficheiros XML e usar o conceito de envelope nas trocas de mensagens, a sua performance pode ser afetada e, consequentemente, apresenta uma maior complexidade na sua construção face ao REST. [9]

A arquitetura REST é conhecida por ser *stateless* e de separar as preocupações do cliente e do servidor. Um sistema que segue um paradigma REST é *stateless*, visto que o servidor não precisa saber nada sobre o estado do cliente e vice-versa. Desta forma, tanto o servidor quanto o cliente, podem entender qualquer mensagem recebida, mesmo sem ver as mensagens anteriores. Esta característica é reforçada, por meio do uso de recursos, em vez de comandos. Os recursos, basicamente, descrevem qualquer objeto, documento ou outra coisa que seja necessário armazenar ou enviar para outro sistema. Estes têm como identificador um *Uniform Resource Identifier* (URI), que serve de apoio quando queremos identificar um recurso para termos uma resposta de volta, do servidor. Esta resposta pode vir em HTML, XML ou JSON, sendo mais utilizada nos dias de hoje, devido à sua forma simplista e rápida na troca de dados, o JSON. Este pedido do cliente para o servidor, no REST, é realizado através do protocolo HTTP, de forma a obter a interação com os recursos. Consequentemente, nesta interação utilizam-se os chamados métodos HTTP, tal como os mais usados GET, POST, PUT e DELETE. Relativamente à interação entre cliente e servidor, podemos perceber que a implementação do cliente e a implementação do servidor podem ser feitas de forma independente, sem que cada um conheça o outro. Isso significa que o código no caso do cliente, pode ser alterado, a qualquer momento, sem perturbar a operação do servidor, e o código no servidor, pode ser alterado sem afetar a operação do cliente. Desta forma, permite escalar cada sistema, sem que nenhum seja afetado. [39]

Após estas duas arquiteturas possíveis a utilizar, optamos por usar a arquitetura REST para realizar a troca de informação entre o servidor e cada dispositivo móvel de cada pessoa, ou seja, a parte de FE. Esta escolha, advém dos factos diferenciados que referimos em cima,

como a escalabilidade permitida pela independência entre o servidor e o FE, também não é necessário um reconhecimento acerca de um estado da informação passada, na troca de dados e o facto de a informação poder ser tratada num ficheiro JSON, o que aumenta a performance e apresenta fiabilidade nesta arquitetura.

Por outro lado, também precisamos de ter em atenção a outra parte do nosso trabalho, que é a forma como se faz a transmissão de dados entre os sensores e o servidor. Para este já não nos serve pensar nem numa arquitetura SOAP, nem REST, visto que não haverá respostas, apenas valores a serem recebidos no servidor frequentemente, sempre que haja mudanças nos sensores. Para este caso, nós pensamos na troca de dados, utilizando o padrão de *Publish-Subscribe* (PS). PS permite que as mensagens, sejam transmitidas para diferentes partes de um sistema, de forma assíncrona. Este padrão utiliza uma fila de mensagens, cada uma correspondente a um tópico, com um nome a identificá-lo. Para transmitir uma mensagem, um componente chamado *publisher*, simplesmente envia uma mensagem para um respetivo tópico. Ao contrário das filas de mensagens, que as enviam em lote até serem recuperadas, os tópicos das mensagens transferem-nas sem nenhuma ou muito poucas filas e enviam-nas imediatamente para todos os subscritores. Todos os subscritores que subscrevem o tópico, receberão todas as mensagens transmitidas, a menos que uma política de filtragem de mensagens seja definida pelo subscritor. Por outro lado, um tópico pode apresentar vários subscritores e vários *publishers*. Além disso, ainda se pode configurar o tópico, de forma a que a fila de mensagens que ele apresenta, seja eliminada num intervalo de tempo específico. [17] Visto que pretendemos estar sempre a receber informações dos sensores, não sabendo quando é que elas chegam, a possibilidade de ter guardado em cache do tópico os vários dados, leva que este paradigma vá de encontro a todo o nosso pensamento, nesta parte do nosso trabalho. A figura 4.3 apresenta o que foi dito aqui, como é feita a troca de dados num método PS. Apresenta um tópico central, dois *publishers* e três subscritores, os quais recebem a informação retida no tópico enviada pelos dois *publishers*.

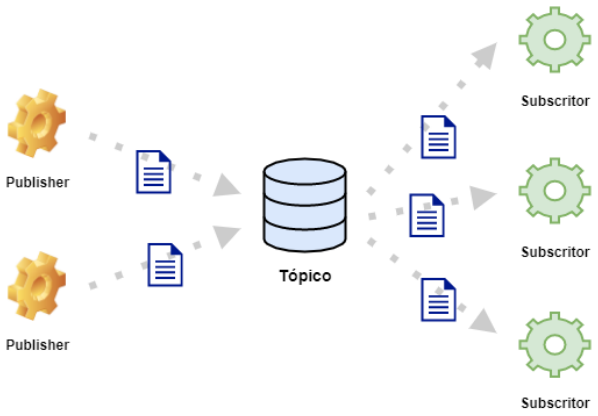



Figura 4.3: Ilustração do paradigma PS, apresentando dois *publishers* e três subscritores ao tópico.

## 4.2 Implementação do back-end

Na implementação do nosso sistema, tivemos que pensar em todas as partes do mesmo, visto que foi tudo feito de raiz. Na parte do servidor, optámos por utilizar a *framework* Spring. Esta utiliza a linguagem JAVA, trabalha com um conjunto de dependências que são automaticamente configuradas pela aplicação, o que permite fornecer ao utilizador uma simplicidade enorme na hora de executar a mesma. Em termos das arquiteturas que vamos usar, Spring dá um grande apoio a quem desenvolve o sistema, visto que possui mecanismos já predefinidos para arquiteturas REST e PS. A escolha recaiu no Spring, visto que também é um ambiente do qual já tínhamos uma ideia prévia acerca do seu funcionamento e da sua simplicidade na implementação de aplicações e na consequente redução de tempo de desenvolvimento do sistema.

Em relação à troca de informação utilizando o paradigam PS, decidimos utilizar o Kafka. O Kafka é uma plataforma de transmissão de dados que utiliza o PS no seu funcionamento. A transmissão é feita através de um produtor (publisher) e consumidor (subscritor), existindo um *broker*, com o respetivo nome do tópico, na intermediação dos dados. Este pode apresentar várias partições e destinar a publicação de mensagens para diferentes partições no *broker*. Os consumidores estão associados às partições de cada tópico. Satisfazendo os nossos propósitos através do seu funcionamento, o Spring também faz ligação com o Kafka, o que nos permite uma maior facilidade no seu manuseamento. Como tal, esta interação de ambos foi o ponto fundamental para a nossa escolha.

Para além destas duas *frameworks*, tivemos de discutir acerca de qual seria a base de dados preferencial a usar. Uma base de dados é a parte do nosso sistema que serve para guardar todos os dados de forma permanente, para o servidor ser monitorizado face aos dados lá guardados. É importante pensar neste assunto, quando queremos desenvolver um sistema, visto que serão estes dados armazenados, que serão usados para executar processamentos e manter o correto funcionamento do nosso sistema. Devido a podermos ter falhas no sistema e ser necessário o reinicio do mesmo, decidimos optar por uma base de dados que mantenha os dados inseridos, de uma forma permanente. Imaginemos um reinício do sistema, estávamos em situação de alarme, se não tivessemos a informação acerca do mapeamento feito em cada aresta, o nosso sistema estaria a dar informações erradas. Da mesma forma que, também para os dados, relativamente aos nós e arestas existentes do grafo, seria necessário fazer novamente o processamento para a base de dados, dessas mesmas estruturas.

De acordo com a tabela anterior, decidimos apoiar o nosso trabalho numa base de dados não relacional, visto que não pretendemos fazer relacionamentos entre tabelas, apenas ter os objetos criados do sistema guardados, que nos permita guardar vários dados e que seja eficaz e rápido. De entre o conjunto de softwares de base de dados, decidimos continuar o nosso trabalho com a ajuda do MongoDb, pois apertados relativamente ao tempo de realização do projeto, este apresentou uma maior simplicidade na sua aprendizagem. Os objetos são guardados em formato BSON, não precisando de apresentar a mesma estrutura ou campos, podendo este ser de vários tipos. A sua simplicidade na compreensão do software, a interface gráfica que apoia na visualização e mesmo na possível mudança de campos da base de dados, foram dois aspetos que vieram ajudar a completar a nossa decisão. Por outro lado, o Spring, como já tinha acontecido com o Kafka, apresenta também uma ligação com o

MongoDb, de forma a facilitar toda a parte de desenvolvimento e acesso à base de dados. Esta ajuda, advém de o spring possuir métodos pré definidir para pesquisar na BD, não sendo necessário escrever todas as *querys* de raiz. Com isto, para cada ED a ser guardada na BD será criado um documento novo, tendo cada um repositórios diferentes no nosso projeto de programação.

### 4.2.1 Estruturas de dados utilizadas para representação do grafo

Para implementar todas estas ideias e visto que trabalhamos na parte de BE com a linguagem JAVA tivemos de, primeiramente, perceber quais as estruturas de dados a utilizar e quais seriam as mais indicadas para o nosso trabalho, de forma a guardar na base de dados e não utilizando pesquisas duradouras entre o servidor e essa mesmo.

Seguindo o caminho por etapas, primeiramente começámos por pensar na melhor maneira de estruturar os dados respetivos aos nós e arestas no sistema. Decidimos que cada objeto nó tem um nome e uma lista com o objeto de cada vizinho, sendo estes vizinhos, as ligações adjacentes ao nó em questão. Continuamente, também tivemos de implementar um id, sendo que este representa o objeto na base de dados. O objeto vizinho é identificado por o nome do nó de um lado da ligação e o nome do nó do lado oposto. Cada ligação vizinho apresenta um valor de distância entre nós e o respetivo valor do nível de perigo presente naquela aresta. Por fim, sendo útil para a resolução do algoritmo, incorporamos o campo da direção, que refere a direção que essa aresta tomou após a resolução do mesmo. As tabelas 4.1 e 4.2 mostram as variáveis usadas para estruturar cada nó e cada aresta do sistema, presentes depois durante a execução do servidor na BD.

Tabela 4.1: Estrutura de dados (ED) referente a cada nó.

| **Tipo** | **Nome da variável** | **Função** |
|---|---|---|
| String | Id | Define o objeto no conjunto de dados da BD. |
| Inteiro | no | Define o nome de cada nó. |
| Lista | vizinhos | Lista com todas as arestas adjacentes. |

Tabela 4.2: ED referente a cada aresta.

| **Tipo** | **Nome da variável** | **Função** |
|---|---|---|
| String | nome_vizinho | Define o nome de cada aresta. |
| Inteiro | primeiro_no | Identifica o nome de um nó das extremidades. |
| Inteiro | segundo_no | Identifica o nome do nó do lado oposto. |
| Float | distancia | Define a distância presente da aresta. |
| Inteiro | nivel_perigo | Define o nível de perigo existente na aresta. |
| String | direcao | Define a direção de que nó para que nó está direcionada a aresta. |

O mongoDB, como dito anteriormente, apresenta uma interface gráfica que permite ao utilizador estar por dentro do conteúdo da BD. Essa interface gráfica é designada por MongoDB Compass Comunity. A figura retrata a nossa ED inserida nesta plataforma, com um exemplo de um nó do grafo, com apenas uma aresta adjacente.

Após a fase de processar o grafo, relativamente a uma planta, esses dados têm que ser gravados para depois, ao correr o servidor, este os poder inserir na base de dados,


Figura 4.4: Exemplo da representação da ED de um nó na interface gráfica do MongoDb, retirado de *MongoDB Compass Comunity*.

atribuindo-os à ED. Como tal, pensámos na hipótese de guardar os dados num ficheiro JSON, visto que este apresenta um formato simples e rápido na sua leitura.

No bloco de código 4.1 está representado um exemplo de um grafo com 4 nós, assumindo a mesma estrutura para o grafo da nossa planta. Ao processar todos os dados, deste ficheiro para o sistema, consequentemente para a BD, processámos cada elemento do array *nos*, representando este, todos os nós presentes no grafo. Para cada chave *no*, procura-se ao longo do array *arestas*, todos os elementos que contêm dentro de cada array *nos*, o nome da chave *no* em questão. Caso esteja presente, adicionamos essa aresta à lista vizinhos. Após a inexistência de mais opções, criamos o objeto com a respetiva lista de vizinhos. De referir, na ED de cada aresta, inicialmente a variável *nome_vizinho* será definida através dos dois números presentes no array,(primeiro numero)_(segundo numero). As variáveis *nivel_perigo* e direção, terão o nível de perigo inicial mais baixo e sem indicação, respetivamente.

```
{
  "nos": [
    { "no": 1 },
    { "no": 2 },
    { "no": 3 },
    { "no": 4}
  ],
  "arestas": [
    {"nos": [1, 2], "distancia": 2},
    {"nos": [1, 4], "distancia": 4},
    {"nos": [2, 3], "distancia": 1},
    {"nos": [4, 3], "distancia": 5},
  ]
}
```

Bloco de Código 4.1: Exemplo ficheiro JSON, de como são guardado os dados após o processamento do grafo face à planta.

### 4.2.2 API para trocas de informação entre os sensores e servidor

Para além de estruturar a parte do grafo do nosso sistema, tivemos que pensar na estruturação que existiria na troca de informação dos sensores e respetiva mudança na BD, dos valores de perigo de cada aresta.

Decidimos dividir o processamento em 3 partes diferentes, utilizando módulos auxiliares ao servidor, para tratamento da informação, de forma a facilitar a compreensão do sistema num futuro. As 3 partes estão interligadas. De referir novamente que toda esta troca de informação será efetuada com o auxílio do Kafka, em projetos Spring.

Primeiramente, criámos um módulo que está em contacto com cada sensor. Após receber informação de um sensor, este possui uma ED, como vista na figura 4.3, que envia o objeto para o tópico do Kafka com o nome *primeira_interacção*.

Tabela 4.3: ED referente ao sensor do primeiro módulo, no processamento de informação vinda dos sensores

| **Tipo** | **Nome da variável** | **Função** |
|---|---|---|
| Inteiro | id_sensor | Define o nome do sensor que enviou informação. |
| Inteiro | valor_enviado | Representa o valor enviado por o sensor |

A segunda parte do processamento da informação dos sensores, trata de pegar nos dados que foram publicados no tópico *primeira_interacao*, processá-los e enviá-los a outro tópico que faça ligação com a terceira parte. Nesta parte tivemos que elaborar 3 estruturas de dados, de forma a dividir a receção de dados do tópico, a fase de processamento da informação e envio para um segundo tópico.

Numa primeira fase neste módulo, o objeto recebido do primeiro tópico, será guardado numa ED, igual à representada na tabela 4.3.

Continuamente, numa segunda etapa, os objetos seguindo a ED representada na tabela 4.4, serão guardados na BD. Optámos por guardar na BD, pois estes farão o controlo dos valores de perigo presentes em cada aresta, de forma que com os valores recebidos, se analise se aumentaram ou diminuiram. Por outro lado, referem o tipo de sensor a que corresponde, se são sensores de fumo ou de temperatura. Já foi referido o input recebido pelo sensores de fumo. Pelos sensores de temperatura será um inteiro, a indicar a temperatura correspondente que o sensor detetou. No caso de recebermos o valor de um sensor de fumo, se for 1 e se o valor de nível perigo presente na ED do sensor correspondente, demonstrar que não tinha recebido o valor referente à existência de fumo ainda, aumenta-se um nível de perigo. Por outro lado, no caso da temperatura, os intervalos de valores que decidimos trabalhar são 3, uma vez que serviram apenas para todo o sistema trabalhar, visto que entra num tema fora do nosso estudo. Atribuímos o primeiro intervalo de valores para valores menores que 51, o segundo entre os valores 51-70, inclusive e por fim, valores maiores que 70. Neste caso, cada intervalo de valores controlará o nível de perigo a aumentar, ou diminuir em cada aresta. O primeiro intervalo terá nível de perigo 0, o segundo corresponderá ao nível de perigo 1 e o terceiro ao nível de perigo 2.

Tendo os sensores divididos por tipos, antes de passar à fase de processamento do nível de perigo neste segundo módulo, é importante ter algum ficheiro que guarde a que tipo

Tabela 4.4: ED referente à fase de processamento de informação do tópico *primeira_interacao*, do segundo módulo.

| Tipo | Nome da variável | Função |
|---|---|---|
| String | Id | Define o objeto representativo de cada sensor, na BD. |
| Inteiro | nome_sensor | Representa o nome do sensor |
| String | tipo_sensor | Representa o tipo do sensor |
| Inteiro | nível_perigo | Representa o valor de perigo atual das arestas do sensor. |

corresponde cada sensor. Este armazenamento será então enviado para a BD. Decidimos guardar num ficheiro JSON, como demonstrado no bloco de código 4.2. Para cada categoria de tipo de sensor, neste caso apenas temos o array *fumo* e *temperatura*, em que estão associados os nomes de cada sensor no sistema, que têm a função de verificar a existência de fumo ou mudanças de temperatura. Consideramos neste exemplo, a ocorrência apenas de 4 sensores, sensores 1 e 3 do tipo fumo e 2 e 4 analisando a temperatura.

```
{
  "fumo": [
    { "sensor": 1 },
    { "sensor": 3 }
  ],
  "temperatura": [
    { "sensor": 2 },
    { "sensor": 4 }
  ]
}
```

Bloco de Código 4.2: Exemplo ficheiro JSON, que define a que tipo de sensor, os sensores do edifício pertencem.

Por último, nesta terceira etapa dentro do segundo módulo, realizámos a parte de enviar a informação para o tópico *segunda_interacao*. O objeto que enviaremos para este tópico terá a estrutura da tabela 4.6. A variável *nome_sensor* será então o nome do sensor que enviou informação. A variável *adicionar_valor*, será o valor da diferença, correspondente à mudança da variável *nivel_perigo* na segunda etapa deste segundo módulo. No caso de ser um valor negativo, quer dizer que diminui o valor de perigo, se for positivo aumentou. Na possibilidade de não haver diferença, não será enviado nada para o tópico, visto que não existiram alterações nos valores de perigo.

Tabela 4.5: ED referente à fase de envio de informação para o tópico *segunda_interacao*, no segundo módulo.

| Tipo | Nome da variável | Função |
|---|---|---|
| Inteiro | nome_sensor | Define o nome do sensor que enviou informação. |
| Inteiro | adicionar_valor | Representa o valor a adicionar às arestas dos sensores. |

A última parte no processamento de informação vinda dos sensores, até à mudança do valor de perigo, em cada aresta do grafo, é já efetuada dentro do servidor. Temos guardada a informação, a que sensor pertence cada aresta, num repositório da BD. Esta informação é obtida através de um outro ficheiro JSON, procede-se à sua leitura e envia-se a informação para a BD. A estrutura do ficheiro é identificada no bloco de código 4.3. Neste, temos o exemplo de dois sensores, com os nomes 1 e 2, sendo responsáveis por duas arestas cada um.

```
{
  "sensores": [
    {
        "sensor": 1,
        "arestas": [
            { "aresta": [1, 2]},
            { "aresta": [1, 3]}
        ]
    }
    {
        "sensor": 2,
        "arestas": [
            { "aresta": [2, 3]},
            { "aresta": [3, 4]}
        ]
    }
  ]
}
```

Bloco de Código 4.3: Exemplo ficheiro JSON, no qual são definidas as arestas, pelo qual cada sensor é responsável.

Estes dados vão ser armazenados em objetos, com base na ED da tabela 4.6. Terão um a identificar o objeto na BD, outro o nome de sensor, e uma lista com objetos, representando as arestas. Este objeto dentro da lista referenciada terá a estrutura da tabela 4.7.

Tabela 4.6: ED que representa cada sensor e quais arestas este é responsável, no servidor.

| **Tipo** | **Nome da variável** | **Função** |
|---|---|---|
| String | Id | Define o objeto na BD. |
| Inteiro | nome_sensor | Identifica o nome de um sensor do edifício. |
| Lista | arestas | Lista de arestas, cujo sensor é responsável. |

Tabela 4.7: ED que representa cada aresta, da lista de arestas de cada sensor.

| **Tipo** | **Nome da variável** | **Função** |
|---|---|---|
| Inteiro | primeiro_no | Define um nó da extremidade da aresta. |
| Inteiro | segundo_no | Define o outro nó da extremidade da aresta. |

A mudança de valor, em cada aresta do grafo, é feita através da informação contida no tópico *segunda_interacao*. O servidor tem subscrito este tópico. Tendo o nome do sensor e o valor da diferença, vai-se à BD pesquisar por o respetivo sensor, e as listas de arestas. Para cada aresta, utiliza-se os dois nós. Para cada um, prossegue-se na pesquisa pelas suas arestas adjacentes. Uma vez encontrada esta aresta, adiciona-se o valor que chegou do segundo módulo, ao nível de perigo.

Decidimos elaborar o esquema representado na figura 4.5, de forma a esquematizarmos os vários passos já explicados na parte do BE, desde o início do alerta por parte de um sensor, passando pelas diferentes etapas modulares no processamento desta informação dos sensores, usando nesta o padrão Kafka. Após estes procedimentos, o servidor terá o algoritmo a correr periodicamente. Estabelecemos um intervalo de tempo para o algoritmo, pois os sensores podem estar a enviar informações frequentemente, o que leva a mudanças de rota constantes. Na parte da pessoa a evacuar isto não será bom, pois ficará confusa ao ver diferentes rotas a tomar, constantemente. Posto isto, optámos por esta opção, sendo na nossa ideia uma resposta válida a este problema. Recebendo pedidos GET de informação, por parte do FE, o servidor calcula a melhor rota, transforma os dados e comunica com o FE, apoiando-se na arquitetura REST.

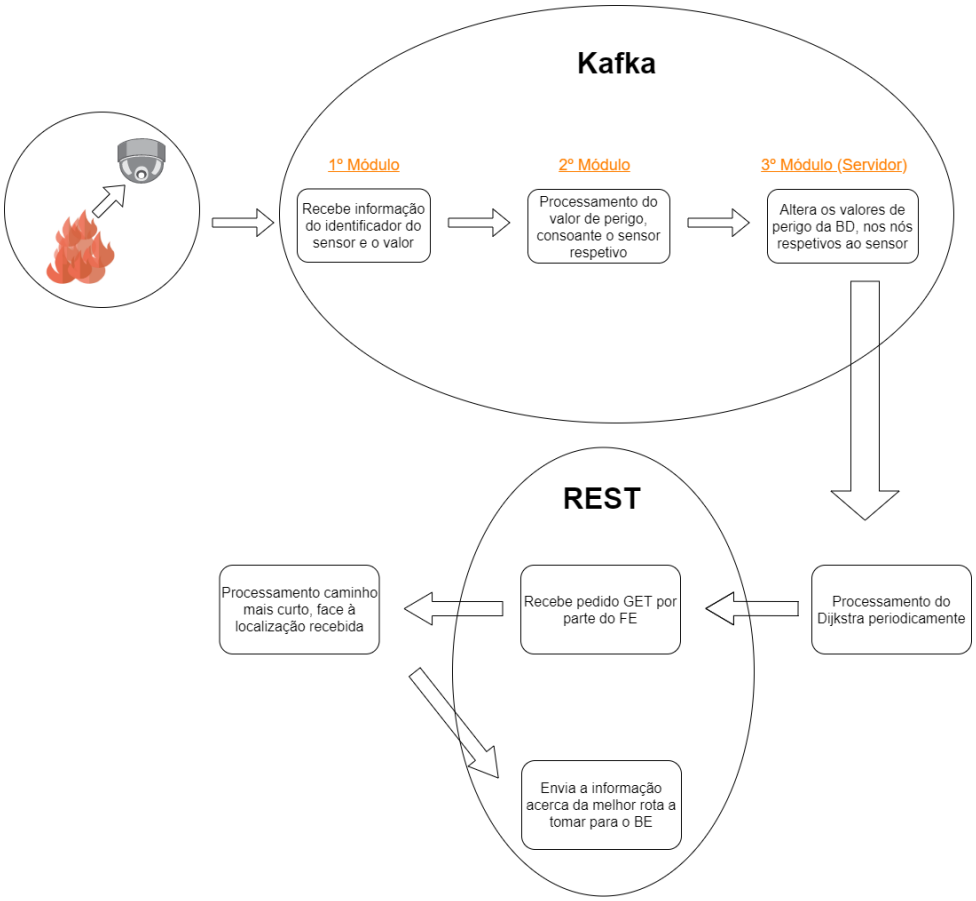



Figura 4.5: Resumo esquemático dos passos efetuados na parte do BE do nosso sistema desenvolvido.

## 4.3 Implementação do front-end

Após pensarmos, discutirmos e implementarmos as ideias de como estruturar a parte do BE, passámos então à parte de desenvolver o FE. Nesta fase do projeto, não nos focamos no design da aplicação, mas sim no seu funcionamento. A sua estruturação e funcionamento foi o nosso principal objetivo.

Na medida em que queremos desenvolver um sistema para um dispositivo móvel, optámos por utilizar a *framework* ionic para elaborarmos esta parte. Fizemos esta escolha, visto que já tínhamos umas noções base acerca do seu funcionamento e com mais algum estudo desta, conseguiríamos atingir com sucesso o pretendido. Pretendíamos expôr o funcionamento do nosso sistema, tanto em ambiente IOS como android. O ionic também nos permite esta possibilidade de *cross-plataform*, pois admite a incorporação da ferramenta capacitor, no seu projeto, e com esta, conseguimos fazer a conversão para qual o ambiente nativo que pretendemos, ou android ou IOS.

Nesta fase avançada do nosso trabalho, precisávamos de saber quais os dados a receber do BE e de que forma os tratar. O ionic permite-nos utilizar uma maneira de subscrever pedidos HTTP. Foi criado um serviço no nosso projeto do ionic, com o pedido GET para obter os dados do melhor caminho. Este pedido tem presente no seu *path* a localização do mesmo e envia-a para o servidor.

O servidor, por sua vez, analisando a localização da pessoa, envia essa mesma localização do nó, mais perto de onde a pessoa se encontra. Por sua vez, a informação acerca de que área do ecrã gráfico, do dispositivo móvel, abrange cada sensor, foi estruturada para ser gravada na parte do BE, presente num repositório da BD. Estes dados foram lidos primeiramente de um ficheiro JSON, com o formato exemplificado em 4.4. Na chave *id* temos o identificador do nó correspondente. O array *x*, corresponde ao intervalo de valores posicionais na planta, atribuídos ao respetivo nó, para o eixo do x. O *y* é o mesmo procedimento do que o *x*, mas para o eixo do y.

```
{
    "nodes": [
        {
            "id": 1,
            "x": [10, 19],
            "y": [60, 89]
        },
        {
            "id": 2,
            "x": [20, 28],
            "y": [76.5, 89]
        }
    ]
}
```

Bloco de Código 4.4: Exemplo ficheiro JSON, onde estão definidos intervalos posicionais, no qual posições de pessoas dentro desses mesmo, correspondem ao respetivo nó.

O envio de informação do servidor será efetuado através de um array de arrays. Cada elemento array será constituído por duas posições, correspondendo a cada uma, o nome de cada nó. Por outro lado, a direção de entre os dois nós, será sempre do nó da primeira posição do array, ao da segunda. O esquema representado na figura 4.6 exemplifica a situação explicada. A parte de baixo da figura corresponde a um possível array a receber. Cada número corresponde ao identificador do nó, cuja representação no grafo toma as direções representadas na parte de cima da figura.

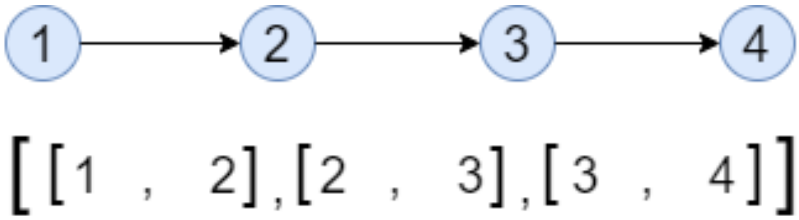



Figura 4.6: Formato da ED dos dados transmitidos entre FE e BE.

Pensámos inicialmente, visto que teríamos disponíveis numa primeira fase, material que nos pudesse indicar a localização de uma pessoa, num espaço no interior do edifício, utilizar a latitude e longitude como orientação neste caso da localização. Porém, não foram disponibilizados materiais e ferramentas para tal e tivemos de elaborar outra alternativa para esta parte. Decidimos então fazer o nosso sistema de coordenadas através de uma grelha de quadrados, feitos com o elemento div, possibilitado pelo HTML. Cada posição de localização seria trabalhada em forma de percentagem face ao ecrã, de maneira a que continuasse tudo bem posicionado na mudança de dispositivos móveis. Com esta grelha, em volta de toda a planta na interface gráfica, conseguimos através da posição de cada quadrado, saber a posição de cada nó. Por causa desta, conseguimos também simular a posição de uma pessoa e perceber a que nó mais próximo corresponde a opção. A grelha que utilizámos está representada na figura 4.7, fornecendo-nos uma orientação relativamente a posições no mapa da planta. As duas cores utilizadas, vermelho e azul, estão em evidencia para nos facilitarem na pesquisa da localização na grelha.

Precisávamos de pensar numa solução no design da direção das setas, após a recolha de dados do BE, do melhor caminho entre nós. Auxiliamo-nos da ferramenta *Scalable Vector Graphics* (SVG). Esta usa gráficos vetoriais, de forma a ilustrar algum desenhos e esquemas. Foi a única hipótese encontrada para a representação do caminho até uma saída, pois possibilita dinamismo na eliminação e criação de elementos constantemente. Contudo, também apresenta os seus elementos com boa qualidade e com ampliação ou redução da imagem, esta não perde qualidade.

O desenho da ligação será apresentada de nó em nó, através de linhas criadas no SVG. Estas linhas precisam das coordenadas bidimensionais, x e y, de cada nó a ligar. Como tal, necessitamos de averiguar a posição de cada nó na planta e inseri-la num ficheiro JSON, dentro do projeto onde está a parte do FE. Este ficheiro guardará a posição de cada nó, para que quando o FE recebe a comunicação dos nós do caminho, por parte do BE, vai-se

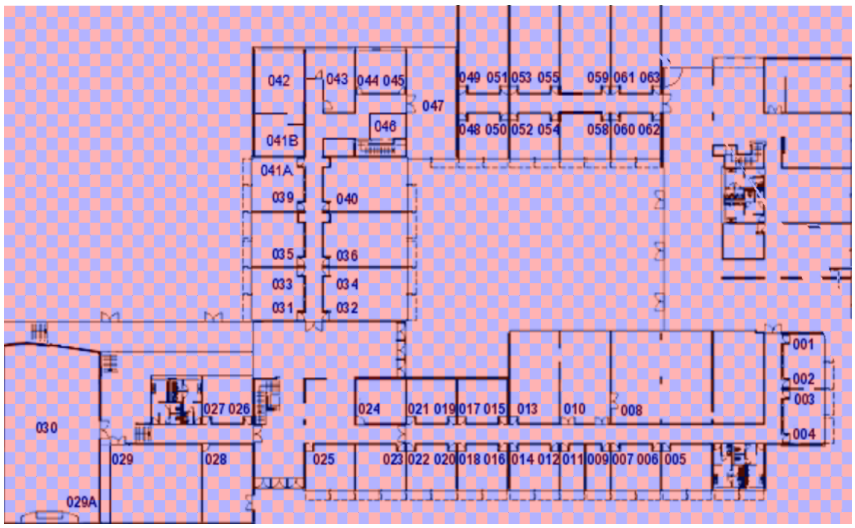



Figura 4.7: Representação da grelha elaborada, de forma a esclarecer-nos a localização de cada sítio da planta, no ecrã de uma interface gráfica.

verificar qual a posição do nó respetivo. O bloco de código 4.5 é um exemplo de como foi estruturado o ficheiro JSON. Tem dentro do array *position* todos os nós do grafo. Cada nó o *id*, sendo o nome do respetivo nó e depois a posição do mesmo, relativamente ao eixo dos x e y, em percentagem, em relação ao tamanho total do ecrã.

```
{
    "position":[
        {
            "id" : 1,
            "x" : 85,
            "y" : 63
        },
        {
            "id" : 2,
            "x" : 80,
            "y" : 63
        }
    ]
}
```

Bloco de Código 4.5: Exemplo ficheiro JSON, onde são guardadas as posições de cada nó na interface gráfica, em valores de percentagem.

Relativamente à interface, optámos por inserir apenas a imagem da planta em estudo e ir fixando as linhas consoante os caminhos obtidos. Uma interface simples, mas que dá para testarmos todo o nosso sistema. Neste caso, a atualização dos valores será feita constantemente, de forma a manter sempre a interface atualizada.

Juntamente com o esquema feito para a parte de BE em 4.5, decidimos também resumir as

etapas no FE, com a ajuda do mesmo. Primeiramente, a parte do FE estará constantemente subscrita com o pedido GET, juntamente com a localização de onde se encontra a pessoa a evacuar. Recebe a informação do servidor, faz o processamento do resultado obtido, transforma as ligações com arestas gráficas utilizando SVG e mostra ao utilizador o produto final.

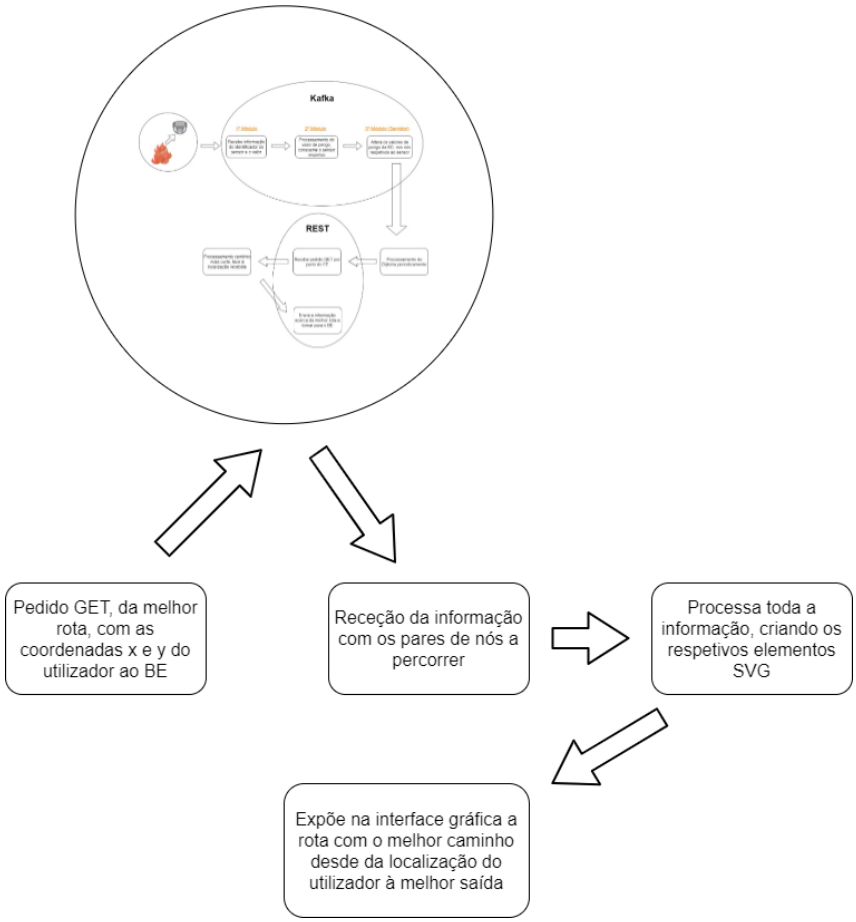



Figura 4.8: Resumo esquemático dos passos efetuados na parte do FE do sistema.

# Capítulo 5

# Resultados e análise

Neste capítulo, apresentamos o resultado obtido de uma simulação que fizemos ao nosso sistema. Alterámos os valores recebidos de cada sensor, calculámos tempos de processamento dos vários procedimentos a efetuar e mostrámos a variação de rotas, consoante as medidas vindas dos sensores.

## 5.1 Demonstração da simulação do sistema

Na nossa demonstração, tivemos de realizar tudo de forma simulada, visto que seria complicado ativar um sistema destes, num ambiente real. Para além da falta de componentes para o fazer, também não conseguiríamos devido ao perigo que todo este processo implica. O sistema apenas foi testado em ambiente android, visto que não conseguimos em ambiente IOS, devido a indisponibilidade de material para testar nesse mesmo.

Como tal, decidimos fazer nós as nossas próprias atribuições durante toda a simulação. Começámos por atribuir a localização da pessoa, face ao nosso mecanismo de localização criado, no eixo do $x = 11$ e $y = 70$, colocando-se esta, na posição mostrada na figura 5.1, onde o símbolo a vermelho representa-a.

Começando por um ambiente onde não se passa nada, no qual nenhum sensor disparou ainda, no caso da pessoa pretender saber qual o melhor caminho até à saída mais próxima, terá o output representado na figura 5.2, onde as linhas a vermelho são arestas entre os nós, representados nas setas verdes.

Após este passo, decidimos então começar com a simulação da parte de disparar os sensores. Visto que não temos nenhum sensor, onde possamos testar, usámos a aplicação Postman, o que nos permitiu simular o envio de dados para o primeiro módulo da troca de informações com o sensor. Este, por sua vez, permite ao programador desenvolver a parte de BE, sem ter a parte de FE feita. Permite criar pedidos HTTP e receber a resposta. No nosso caso, desenvolvemos um pedido HTTP que envia o nome do sensor e um valor para o primeiro módulo. Disparámos primeiramente o sensor verde e a seguir o vermelho, que podemos ver representados na figura 5.3. A sombreado, um abstrato da área que eles controlam. Na figura 5.4, vemos o resultado do caminho que todo o sistema nos fornece apenas com o sensor verde disparado. Por outro lado, a imagem 5.5 mostra-nos o caminho

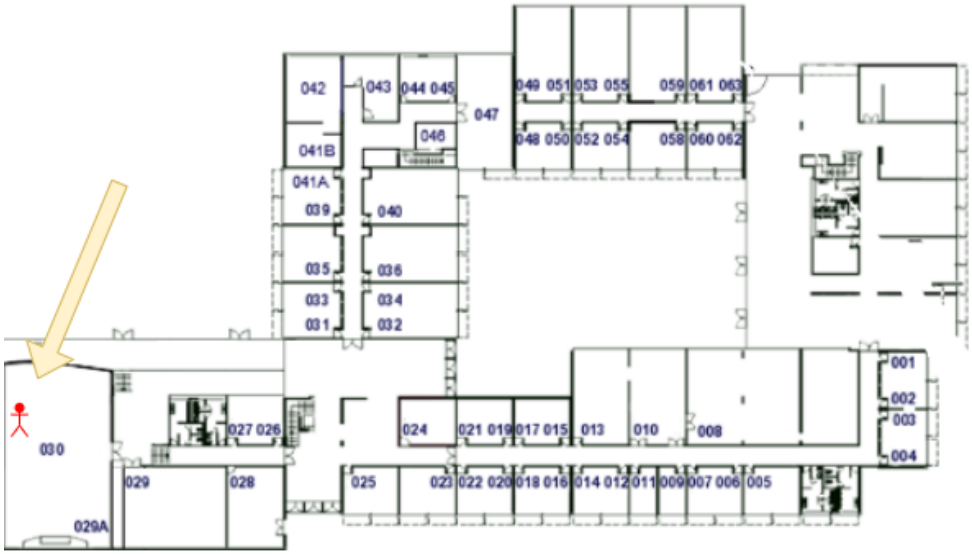



Figura 5.1: Posição da pessoa a evacuar, no decorrer da nossa simulação.

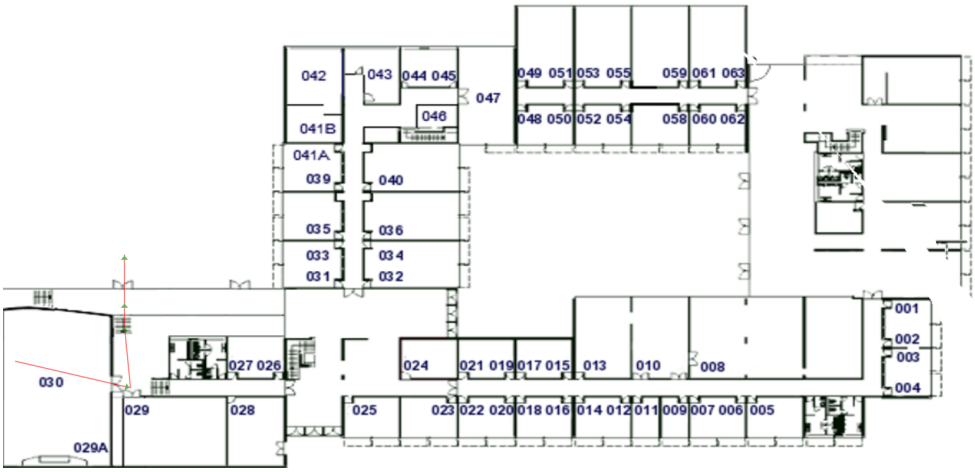



Figura 5.2: Caminho fornecido à pessoa a evacuar, no caso de não existir qualquer problema no edifício.

fornecido no caso dos dois sensores estarem em alerta.

## 5.2 Resultados estatísticos

Na medida que pretendemos estudar o sistema que desenvolvemos, decidimos calcular os tempos de processamento que cada processo leva a realizar. Os tempos, que fizemos para cada parâmetro, correspondem a uma média feita sobre 50 testes feitos, em cada assunto a tratar. Para esta apresentação estatística, convém referir que estamos a tratar de um grafo com 609 nós e 1773 ligações entre nós. Todos estes testes foram realizados num ambiente com processador Intel(R) Core(TM) i7-5500U, no qual a máxima frequência da CPU é de 2400 MHz, apresentando 8GB de RAM.

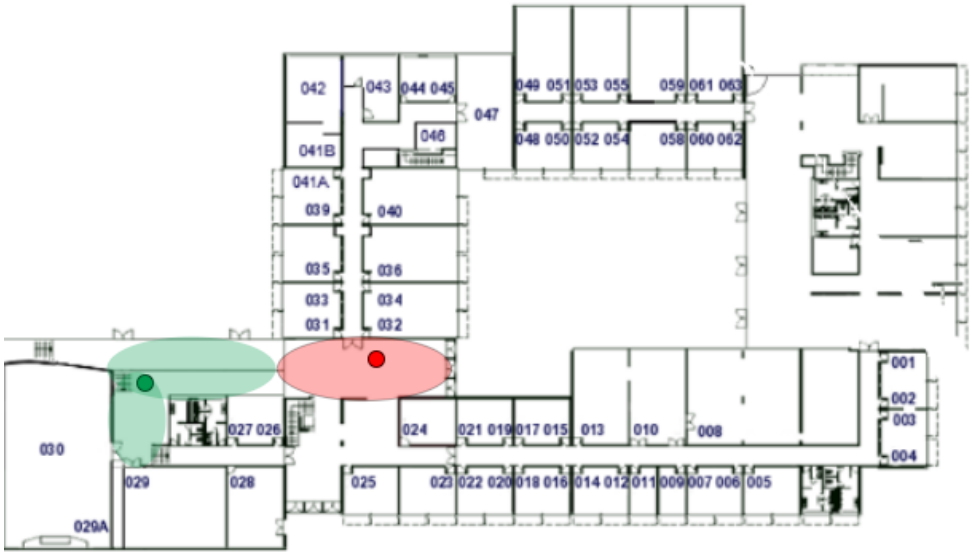



Figura 5.3: Área de controlo, dos dois sensores a utilizar na simulação.

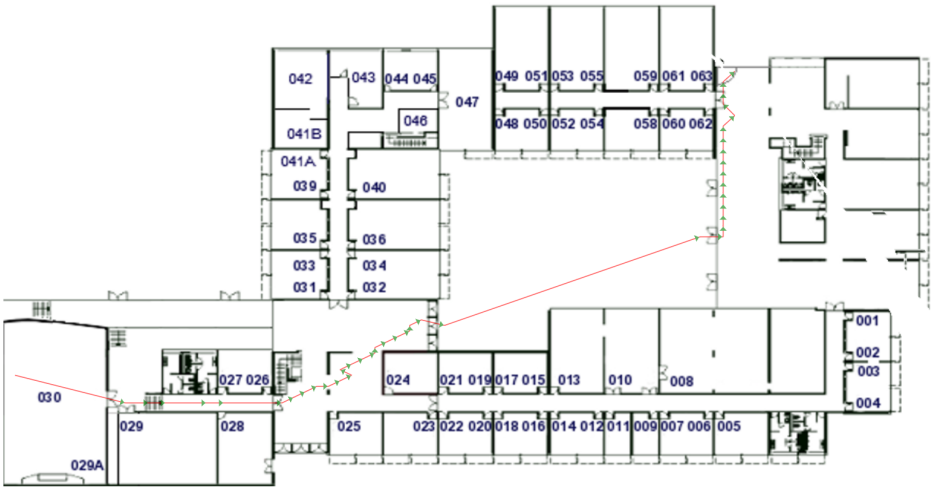



Figura 5.4: Rota indicada a percorrer à pessoa, após o disparo do sensor representado a verde.

Na tabela 5.3 está representado na primeira coluna, o respetivo tempo médio que demora o módulo 1 a receber informação dos sensores, processar e enviar para o módulo 2. Na segunda coluna, o tempo que demora desde o momento em que o módulo 2 recebe os valores do primeiro, até que o servidor processe todos os dados recebidos, alterando os valores na BD. Na última coluna está o somatório dos dois valores já referidos. De realçar que optamos por tratar os valores em milissegundos, pois este tipo de transferência ocorre de forma rápida, o que tratar os dados noutra unidade não faria sentido. Analisando estes valores, concluímos que todo este processamento, desde do disparo do sensor até à mudança na BD, apresenta valores bastante razoáveis na nossa opinião.

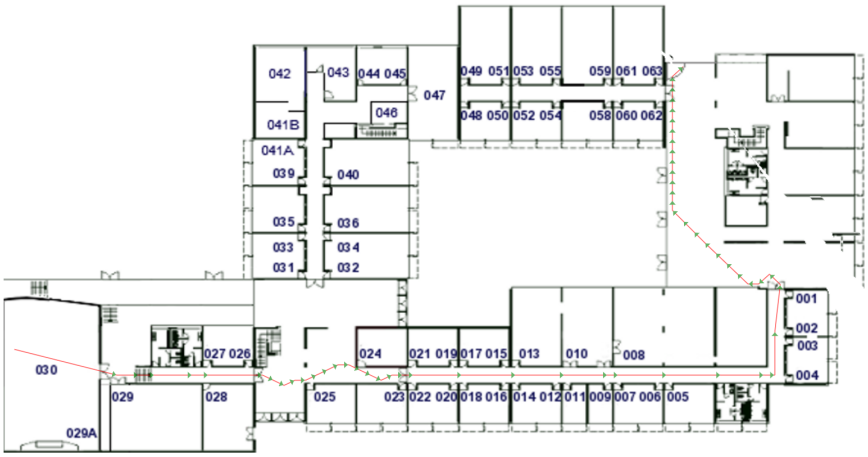



Figura 5.5: Rota indicada a percorrer à pessoa, no caso de os dois sensores terem sido disparados.

Tabela 5.1: Tempos médios do processamento de informação, em cada módulo, responsáveis pelo tratamento dos valores enviados pelos sensores até ao servidor.

| **Módulo 1 -> Módulo 2** | **Módulo 2 -> Módulo 3/Servidor** | **Tempo total** |
|---|---|---|
| 25ms | 52ms | 77ms |

Os tempos médios de processamento do nosso algoritmo estão representados na tabela 5.1. O nosso intuito, neste caso de estudo, foi calcular o tempo que demoraria o algoritmo a processar, existindo um caminho até uma saída, apenas com níveis de perigo 1, depois juntando as arestas com perigo 2, continuamente com as de 3 e 4. Ao contrário da unidade dos milissegundos utilizada na tabela 5.3, neste caso utilizamos segundos, pois nestes valores já se consegue diferenciar o tempo de cada processamento de melhor forma. De forma a tirar uma perspetiva acerca destes resultados, percebemos que o tempo de processamento vai crescendo, de maneira que chegando a rotas que tenham saída que seja necessário utilizar arestas com o máximo valor de perigo, o tempo adquirido já começa a ser elevado, sendo um problema encontrado e a ser melhorado no futuro.

Tabela 5.2: Tempos médios do processamento do algoritmo para diferentes níveis de perigo.

| **Nível 1** | **Nível 1/2** | **Nível 1/2/3** | **Nível 1/2/3/4** |
|---|---|---|---|
| 3s | 5s | 7s | 10s |

Analisamos, a nível temporal, cada etapa realizada na parte do BE, faltando apenas saber os valores referentes desde o momento do pedido, por parte do FE, até à exposição gráfica da rota para o utilizador. Para estes dados, voltamos a tratar os dados em milissegundos, visto que o seu processamento é repentino. Na primeira coluna da tabela 5.2 está representado o tempo médio que demora ao envio do pedido, por parte do FE, passando pelo processamento da resposta, por parte do BE, e posterior receção do FE. Na segunda coluna, o tempo médio definido retrata todo o processamento, desde do momento em que o FE recebe a resposta, trata a informação e consequentemente insere-a na interface gráfica. A nível da parte das etapas ocorridas no FE, esta apresenta um tempo de execução na nossa opinião bom. Porém,

a parte do FE que mais nos interessa saber é a interação com o BE, pois num futuro esta parte terá que ser aprofundada e desenvolvida com uma melhor interface para o utilizador.

Tabela 5.3: Tempos médios do processamento da melhor rota, desde do pedido da melhorar rota, por parte do FE, ao BE, até à exposição gráfica.

| **Pedido ao BE -> Receção da resposta FE** | **Processamento da resposta -> exposição gráfica** |
|---|---|
| 278ms | 10ms |

Concluindo este tópico da estatística para cada resultado, foram apresentados tempos médios para cada etapa, fazendo 50 testes para cada uma, resultando o tempo médio desses testes. Não sendo valores com uma grande especificidade, visto que o computador tem outros processos a correr e podem haver outras circunstâncias no cálculo de cada tempo, serão relevantes como valores base, que servirão de termo de comparação inicial para trabalhos futuros realizados neste sistema.

# Capítulo 6

# Conclusões

No início, o nosso tempo foi passado a estudar e investigar cuidadosamente toda a pesquisa envolvente ao nosso projeto existente, permitindo assim uma boa compreensão acerca do estado da arte existente. Este passo ajudou-nos a estabelecer os pontos centrais para a produção do nosso plano de trabalhos, de forma a conseguir conduzir-nos ao objetivo proposto inicialmente. Relembro que o nosso principal objetivo era encontrar uma forma de obter a rota mais segura, e ao mesmo tempo, mais rápida, para evacuar uma pessoa, no interior de um edifício, em situação de alarme. Continuamente, este objetivo foi alcançado, apresentando nós, ideias para os diversos assuntos a tratar num problema destes, o que comparando com o estado da arte investigado, apresenta ideias que vêm enriquecer este tema.

Este é o ponto em que começámos inicialmente a discutir. Arrancámos pensando na melhor forma de representar uma planta de um edifício computacionalmente, na medida de alcançar uma generalidade para as diferentes estruturas num edifício. Após algumas ideias, umas aceites outras nem por isso, chegámos à conclusão que a atribuição de determinadas regras de posicionamento dos nós, ajudariam a completar o estado da arte deste tópico, bem como a inserção de uma grelha, apenas para um espaço amplo, no sistema e todas as conexões ligada a esta. Contudo, é um tema que levanta algumas questões, visto que todas as plantas são diferentes, daí a uma subjetividade. Por outro lado, a parte de fazer o processamento da imagem da planta e criar o respetivo grafo é um assunto que não foi tratado, como definido desde início.

Após esta definição, idealizámos dois conjuntos de métricas e uma restrição para o algoritmo a desenvolver. Por sua vez, optámos por pegar na base do AD e, de acordo com as nossas ideias, desenvolvemos daí um novo algoritmo, de forma a fazer o melhor cálculo da rota a tomar, por uma pessoa a evacuar. Através da nossa ideia, conseguimos complementar o estado da arte com uma ideia que pode ser trabalhada no futuro.

Numa terceira etapa, concluímos então as nossas ideias com a implementação de um sistema, cujo objetivo era criar uma estrutura para sistemas, que no futuro pretendam lidar com este tipo de assuntos, tendo este como base. Realizámos a fase de tratamento da informação recolhida dos sensores, consequentemente toda a parte do servidor, todo o seu mecanismo e em complemento com a BD. Por fim, elaborámos uma interface gráfica, onde simulámos todo o nosso sistema. Tivemos que nos submeter apenas à simulação, onde trataríamos dos dados de um forma elaborada por nós, visto que não conseguimos

obter ferramentas e materiais, externos ao nosso projeto, que nos possibilitassem um ambiente realístico à simulação. De referir também, que desenvolvemos toda esta nossa implementação no sistema operativo Windows e a utilização do kafka, neste ambiente, ainda apresenta alguns erros por corrigir, sendo aconselhado a sua instalação em ambiente linux, o qual não apresenta problemas nenhum para esta plataforma.

Mesmo assim, estamos convencidos de todo o trabalho que desempenhámos ao longo desta dissertação. Por um lado, conseguimos tratar os diferentes pontos definidos inicialmente, conseguindo idealizar novos fundamentos, novas ideias, novos princípios para o tema em questão. Por outro lado, conseguimos também proporcionar o desenvolvimento de uma estrutura de um sistema que permitirá, num futuro, ajudar pessoas envolvidas neste setor da segurança, no interior de edifícios.

## 6.1 Trabalho Futuro

Embora acreditemos firmemente que tivemos uma contribuição válida e que avançámos com a inclusão de ideias no estado da arte respetivamente a este tópico, há muitas áreas nas quais a nossa abordagem pode ser aprimorada. Terminamos esta tese precisamente dando uma lista de algumas ideias possíveis para trabalhos futuros.

**Planta em ambiente computacional**

As nossas ideias desenvolvidas neste parametro merecem ser utilizadas numa plataforma que permita fazer a conversão de planta para grafo. Como tal, teremos de desenvolver uma que nos permita esse efeito. Por outro lado, o levantamento de questões referente à implementação das regras, no posicionamento dos diferentes nós do grafo, devido a fatores externos, será um dos aspetos a estudar. Um dos estudos a fazer destina-se para os espaços amplos, onde pensaremos na implementação de câmeras que, através do movimento de pessoas, consigam criar padrões de deslocamento e daí analisar a localização de cada obstáculo do local.

**Métricas**

Tendo estas uma grande influência sobre o nosso algoritmo, uma das métricas que propusemos relativa a um nível de perigo, levanta várias questões. Estas questões terão que ser respondidas, fazendo um estudo fora da nossa área, mais direcionando aos fenómenos existentes e depois decidir o nível de perigo correspondente face a uma condição no espaço do edifício.

**Performance**

É necessário garantir que os processamentos efetuados no nosso sistema, sejam efetuados com a máxima rapidez possível. Como tal, o desempenho do nosso algoritmo, a troca de informação entre o sensor até chegar ao servidor e, por sua vez, entre o servidor e exposição gráfica da melhor rota a tomar, terá que ser ainda mais aprofundado, de forma a chegarmos cada vez mais, a melhores resultados.

**Sistema**

Querendo nós atingir sempre mais relativamente ao nosso trabalho, de forma a torná-lo ainda mais fidedigno, experimentaremos o sistema, tentando aproximá-lo ao máximo, a

um ambiente realístico evitando sempre que possível as simulações. Continuamente, a parte de FE será um aspeto forte a melhorar, de forma a ser mais intuitiva e mais explícita para a pessoa a evacuar.

**Publicação**

Visto que queremos que o nosso trabalho seja útil para outros investigadores, estamos a pensar em produzir, pelo menos, uma publicação científica a descrever toda a nossa abordagem feita ao projeto.

# Bibliografia


[1] feup_plantaemergencia. https://sigarra.up.pt/feup/pt/conteudos_geral.ver?pct_pag_id=250033&pct_parametros=pv_unidade=97&pct_grupo=8037&pct_grupo=8036#8036. Acessed January 2020.

[2] dccfcuppiso0. https://sigarra.up.pt/fcup/pt/instal_geral.edificio_view?pv_id=1332&pv_num_piso=0, . Acessed February 2020.

[3] dccfcuppiso1. https://sigarra.up.pt/fcup/pt/instal_geral.edificio_view?pv_id=1332&pv_num_piso=1, . Acessed February 2020.

[4] Security laws. http://www.segurancaonline.com/legislacao/?doc=1#1122. Acessed January 2020.

[5] Feda AlShahwan and Klaus Moessner. Providing soap web services and restful web services from mobile hosts. In *2010 Fifth International Conference on Internet and Web Applications and Services*, pages 174–179. IEEE, 2010.

[6] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[7] Gabriele Bernardini, Matteo Azzolini, Marco D'Orazio, and Enrico Quagliarini. Intelligent evacuation guidance systems for improving fire safety of italian-style historical theatres without altering their architectural characteristics. *Journal of Cultural Heritage*, 22:1006–1018, 2016.

[8] Huibo Bi and Erol Gelenbe. Cloud enabled emergency navigation using faster-than-real-time simulation. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 475–480. IEEE, 2015.

[9] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1, 2000.

[10] Wendel Chan and Costas Armenakis. 3d building evacuation route modelling and visualization. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(2):221, 2014.

[11] Jehyun Cho, Ghang Lee, Jongsung Won, and Eunseo Ryu. Application of dijkstra's algorithm in the smart exit sign. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 31, page 1. IAARC Publications, 2014.

[12] Constantine K Christakos. Sensor networks applied to the problem of building evacuation: An evaluation in simulation. *Proc. 15th IST Mobile and Wireless Summit*, pages 134–138, 2006.

[13] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[14] Antoine Desmet and Erol Gelenbe. Graph and analytical models for emergency evacuation. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 523–527. IEEE, 2013.

[15] Navid Dianati. Unwinding the hairball graph: pruning algorithms for weighted complex networks. *Physical Review E*, 93(1):012304, 2016.

[16] Qian Ding, Zhenghong Peng, Tianzhen Liu, and Qiaohui Tong. Multi-sensor building fire alarm system with information fusion technology based on ds evidence theory. *Algorithms*, 7(4):523–537, 2014.

[17] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[18] Avgoustinos Filippoupolitis and Erol Gelenbe. A decision support system for disaster management in buildings. In *Proceedings of the 2009 Summer Computer Simulation Conference*, pages 141–147. Society for Modeling & Simulation International, 2009.

[19] Avgoustinos Filippoupolitis and Erol Gelenbe. A distributed decision support system for building evacuation. In *2009 2nd Conference on Human System Interactions*, pages 323–330. Ieee, 2009.

[20] JC García-Ojeda, B Bertok, F Friedler, and LT Fan. Building-evacuation-route planning via time-expanded process-network synthesis. *Fire Safety Journal*, 61:338–347, 2013.

[21] Daniele Gianni, Paolo Bocciarelli, Andrea D'Ambrogio, and Giuseppe Iazeolla. A model-driven and simulation-based method to analyze building evacuation plans. In *2015 Winter Simulation Conference (WSC)*, pages 2644–2655. IEEE, 2015.

[22] Gokce Gorbil, Avgoustinos Filippoupolitis, and Erol Gelenbe. Intelligent navigation systems for building evacuation. In *Computer and Information Sciences II*, pages 339–345. Springer, 2011.

[23] Huixian Jiang. Mobile fire evacuation system for large public buildings based on artificial intelligence and iot. *IEEE Access*, 7:64101–64109, 2019.

[24] David B Johnson. Routing in ad hoc networks of mobile hosts. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 158–163. IEEE, 1994.

[25] Joao Leitao, José Pereira, and Luis Rodrigues. Hyparview: A membership protocol for reliable gossip-based broadcast. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 419–429. IEEE, 2007.

[26] Chang Liu, Zhan-li Mao, and Zhi-min Fu. Emergency evacuation model and algorithm in the building with several exits. *Procedia Engineering*, 135:12–18, 2016.

[27] Qingsong Lu, Yan Huang, and Shashi Shekhar. Evacuation planning: a capacity constrained routing approach. In *International Conference on Intelligence and Security Informatics*, pages 111–125. Springer, 2003.

[28] Nenad Medvidovic and Richard N Taylor. Software architecture: foundations, theory, and practice. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 471–472. IEEE, 2010.

[29] Meng-Shiuan Pan, Chia-Hung Tsai, and Yu-Chee Tseng. Emergency guiding and monitoring applications in indoor 3d environments by wireless sensor networks. *International Journal of Sensor Networks*, 1(1/2):2, 2006.

[30] Jake Pauls, Albert J Gatfield, and Edwina Juillet. Elevator use for egress: the human-factors problems and prospects. In *Symposium on Elevators and Fire, Baltimore, MD*, pages 63–75, 1991.

[31] Hector Moner Poy and Brian Duffy. A cloud-enabled building and fire emergency evacuation application. *IEEE Cloud Computing*, 1(4):40–49, 2014.

[32] Guylène Proulx. Evacuation by elevators: Who goes first? In *Workshop on the use of elevators in fires and other emergencies*, 2004.

[33] Pradeep Ramuhalli and Subir Biswas. Managed traffic evacuation using distributed sensor processing. In *Nondestructive Detection and Measurement for Homeland Security III*, volume 5769, pages 48–58. International Society for Optics and Photonics, 2005.

[34] Haichao Ran, Lihua Sun, and Xiaozhi Gao. Influences of intelligent evacuation guidance system on crowd evacuation in building fire. *Automation in Construction*, 41:78–82, 2014.

[35] Chang-Su Ryu. Iot-based intelligent for fire emergency response systems. *International Journal of Smart Home*, 9(3):161–168, 2015.

[36] Thomas L Saaty. Decision making with the analytic hierarchy process. *International journal of services sciences*, 1(1):83–98, 2008.

[37] KAFA Samah, Burairah Hussin, and Abd Samad Hasan Basari. Modification of dijkstra's algorithm for safest and shortest path during emergency evacuation. *Applied Mathematical Sciences*, 9(31):1531–1541, 2015.

[38] Long Shi, Qiyuan Xie, Xudong Cheng, Long Chen, Yong Zhou, and Ruifang Zhang. Developing a database for emergency evacuation model. *Building and Environment*, 44 (8):1724–1729, 2009.

[39] Vijay Surwase. Rest api modeling languages-a developer's perspective. *Int. J. Sci. Technol. Eng*, 2(10):634–637, 2016.

[40] Tatiana Tabirca, Kenneth N Brown, and Cormac J Sreenan. A dynamic model for fire emergency evacuation based on wireless sensor networks. In *2009 Eighth International Symposium on Parallel and Distributed Computing*, pages 29–36. Ieee, 2009.

[41] Yu-Chee Tseng, Meng-Shiuan Pan, and Yuen-Yung Tsai. Wireless sensor networks for emergency navigation. *Computer*, 39(7):55–62, 2006.

[42] Takuya Wada and Tomoichi Takahashi. Evacuation guidance system using everyday use smartphones. In *2013 International Conference on Signal-Image Technology & Internet-Based Systems*, pages 860–864. IEEE, 2013.

[43] Shaohua Wan, Yu Zhao, Tian Wang, Zonghua Gu, Qammer H Abbasi, and Kim-Kwang Raymond Choo. Multi-dimensional data indexing and range query processing via voronoi diagram for internet of things. *Future Generation Computer Systems*, 91: 382–391, 2019.

[44] Jing Wang, Stephan Winter, Daniel Langerenken, and Haifeng Zhao. Integrating sensing and routing for indoor evacuation. In *International Conference on Geographic Information Science*, pages 268–283. Springer, 2014.

[45] Chung-Chuo Wu, Kun-Ming Yu, Shao-Ting Chine, Shao-Tsai Cheng, Yuan-Shao Huang, Ming-Yuan Lei, and Jiann-Horng Lin. An intelligent active alert application on handheld devices for emergency evacuation guidance. In *2013 Fifth international conference on ubiquitous and future networks (ICUFN)*, pages 7–11. IEEE, 2013.

[46] Yuanzhe Xu, Zixun Wang, Qingqing Zheng, and Zhiyong Han. The application of dijkstra's algorithm in the intelligent fire evacuation system. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 3–6. IEEE, 2012.

[47] Moustafa A Youssef, Mohamed F Younis, and Khaled A Arisha. A constrained shortest-path energy-aware routing algorithm for wireless sensor networks. In *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No. 02TH8609)*, volume 2, pages 794–799. IEEE, 2002.