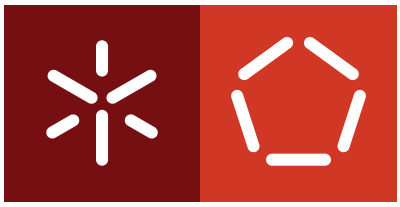
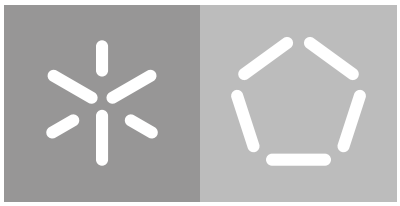


Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Rui de Sousa Miguel

Sistema de Alerta Rodoviário baseado em Comunicações V2X

Novembro 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Rui de Sousa Miguel

Sistema de Alerta Rodoviário baseado em Comunicações V2X

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Dissertação sob orientação de
Prof. Doutor António Duarte Costa
Prof. Doutora Maria João Nicolau

Novembro 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS DO AUTOR

A conclusão desta dissertação foi sempre uma das prioridades do autor. O desenvolvimento deste projeto foi um processo demorado, de extrema importância, onde se depararam dificuldades que ainda não tinham sido encontradas. A ajuda de todos os meus amigos, familiares e conhecidos, foi fundamental não só para esclarecer duvidas essenciais para a sua conclusão, mas também para efeitos morais. Por este facto agradeço os incentivos dados pelos meus amigos mais próximos, bem como a todas as outras pessoas (tanto do Mestrado em Engenharia Informática (MEI) como restantes cursos da Universidade do Minho (UM)) que de alguma forma ajudaram a encontrar soluções alternativas e a superar alguns desafios encontrados.

Agradeço especialmente aos meus orientadores, Professor Doutor António Duarte Costa e Professora Doutora Maria João Nicolau, por todo o apoio dado antes, durante e, seguramente, após o desenvolvimento de todo este projeto.

Agradeço à minha família, em especial aos pais e irmã, que apesar da distância fizeram com que fosse possível dar seguimento aos meus estudos e, por conseguinte, este projeto.

- Muito Obrigado

AGRADECIMENTOS GERAIS

Este projeto é apoiado por: Fundos Europeus Estruturais e de Investimento (*European Structural and Investment Funds*) no Componente FEDER, por meio do Programa de Competitividade Operacional e Internacionalização (COMPETE 2020) [Projeto nº 039334; Referência de financiamento: POCI-01-0247-FEDER-039334].

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Quando falamos em utilizadores em risco num cenário rodoviário, vem-nos à cabeça os peões, ciclistas e possivelmente os motociclistas. Este tipo de utilizadores, particularmente se se tratarem de crianças ou idosos, são especialmente vulneráveis em cenários de perigo quer por erro humano ou por agentes externos. O trabalho realizado no contexto desta dissertação tem o intuito de diminuir o número de fatalidades ocorridas em situações de perigo nas nossas estradas, e tem como objetivo providenciar um método auxiliar de indicação destas mesmas situações de perigo.

O tema das comunicações entre veículos tem vindo a ser abordado há alguns anos, mas ainda há muito trabalho que pode ser desenvolvido nesta vertente. Aplicações capazes de comunicar diretamente com veículos circundantes podem aumentar a segurança rodoviária como um todo, não só dos peões como também dos condutores. A expansão da Internet aos veículos vem possibilitar esta abordagem, sendo que é esperado que futuramente todos os veículos produzidos estejam apetrechados com algum tipo de mecanismo de comunicação. A ideia aqui patente é a de alargar os alertas existentes em veículos mais atuais para os dispositivos móveis dos peões, ou em sentido contrário, comunicar a presença, posição ou até direção de um peão ao veículo.

A principal dificuldade nesta abordagem continua a ser tecnológica, isto dado o facto de não existir uma tendência para tecnologia comum de ampla utilização. Atualmente os dispositivos móveis utilizam tecnologias WiFi/4G e os veículos sobretudo DSRC (brevemente LTE-V). Esta dificuldade inicial poderá, contudo, ser ultrapassada com recurso a dispositivos multi-tecnologia, instalados por exemplo na berma da via pública, sendo estes capazes de agregar e redistribuir informação.

Neste trabalho apresenta-se uma proposta para um sistema de alertas de segurança rodoviária, de baixo custo, para condutores e utilizadores vulneráveis, baseado no uso de dispositivos móveis. A arquitetura do sistema é descrita, apresentado alternativas e justificando as decisões tomadas. O sistema proposto foi implementado na totalidade num protótipo que foi posteriormente testado em cenários realistas com ajuda de um simulador construído para o efeito. Os resultados obtidos permitem constatar a sua viabilidade prática e o funcionamento de acordo com o esperado.

Palavras-Chave: Comunicação, Peão, V2X, Veículo

ABSTRACT

When we think about users at risk in a road scenario, the first thought that occurs reminds us of pedestrians, cyclists and possibly motorcyclists. These types of road users, particularly if they are children or the elderly, are especially vulnerable in dangerous scenarios whether caused by human error or by external agents. The work carried out in the context of this dissertation aims to reduce the number of fatalities occurring in dangerous situations on our roads, by providing an auxiliary method of indicating these same dangerous situations.

The subject of communications between vehicles has been addressed in the last years, but there is still much work that can be developed in this area. Applications capable of communicating directly with surrounding vehicles can increase road safety as a whole, not only for pedestrians but also for drivers themselves. The expansion of the Internet to vehicles makes this approach possible, and it is expected that in the future all vehicles produced will be equipped with some type of communication mechanism. The idea here is to extend the existing warnings on the most current vehicles to pedestrian' mobile devices or, the opposite direction, to communicate the presence, position or even velocity and direction of a pedestrian to the vehicle.

The main difficulty with this approach remains technological, given the lack of a trend towards common technology with a broad use by its users. Currently the mobile devices use WiFi / 4G technologies and the vehicles mainly DSRC (although LTE-V might be arriving soon). However, this initial difficulty can be overcome by means of multi-technology devices installed, for example on the side of public roads, having the capability to aggregate and redistribute information.

This dissertation presents a proposal for a low-cost road safety alert system for drivers and vulnerable users, based on the use of mobile devices. The system architecture is described, presented alternatives and justifying the decisions taken. The proposed system was fully implemented in a prototype that was later tested in realistic scenarios with the help of a simulator built for the purpose. The results obtained allow to verify its practical viability and the operation as expected.

Keywords: Communication, Pedestrian, V2X, Vehicle

CONTEÚDO

LISTA DE FIGURAS

SIGLAS

3GPP *Third Generation Partnership Project*. 1, 13

ACC *Adaptative Cruise Control*. 1, 18
AP *Access Point*. 1, 42, 43, 48
API *Application programming interface*. 1, 26, 29, 37, 38, 42, 45, 47, 48, 58

BLE *Bluetooth Low Energy*. 1, 9, 10, 19
BSM *Basic Safety Messages*. 1

CACC *Cooperative Adaptative Cruise Control*. 1, 18
CAM *Cooperative Awareness Message*. 1, 10, 16, 19
C-ITS *Cooperative Intelligent Transportation Systems*. 1, 13

D2D *device-to-device*. 1, 13, 19
DDoS *Denial-of-service*. 1, 40
DENM *Decentralized Environmental Notification Message*. 1, 10, 12, 19
DHCP *Dynamic Host Configuration Protocol*. 1, 8, 9
DSRC *Dedicated Short Range Communications*. 1, 10, 13, 19

EJS *Embedded JavaScript templating*. 1, 37
ETSI *European Telecommunications Standards Institute*. 1

GPS *Global Positioning System*. 1, 17, 21, 22, 29, 40, 42, 43, 47, 48, 55, 67, 71

IEEE *Institute of Electrical and Electronics Engineers*. 1, 15, 16, 18, 19, 22
IoT *Internet of Things*. 1, 9
ISO *International Organization for Standardization*. 1, 10
ISP *Internet Service Provider*. 1, 34
ITS *Intelligent Transportation Systems*. 1, 13, 14

LOS *Line-of-sight*. 1, 18
LTE *Long-Term Evolution*. 1, 9, 10, 13, 16, 19, 48, 71

MEI Mestrado em Engenharia Informática. ii, 1

NLOS *Non line-of-sight*. 1, 4, 18

OBD2 *On-board Diagnostics Version 2*. 1, 21, 47

OMS *Organização Mundial de Saúde*. 1, 4
OS *Operating System*. 1, 27

P2P *Peer-to-peer*. 1, 8–10, 13, 14, 19, 21
P2P-GO *Peer-to-peer group owner*. 1, 8, 13, 19
PM2 *Process Manager 2*. 1, 38
POV *point-of-view*. 1, 18

RSU *Roadside Unit*. 1, 10

UI *User Interface*. 1, 28
UM Universidade do Minho. ii, 1

V2H *Vehicle-to-home*. 1, 10
V2I *Vehicle-to-Infrastructure*. 1, 15
V2P *Vehicle-to-pedestrian*. 1, 9, 12, 13, 19
V2V *Vehicle-to-vehicle*. 1, 9
V2X *Vehicle-to-everything*. 1, 9, 10, 13, 15, 19
Vbr2P *vehicle broadcast to pedestrian*. 1, 15
VRU *Vulnerable road users*. 1, 7, 10, 13, 14, 16–19

WLAN *Wireless local area network*. 1, 15, 16

1

INTRODUÇÃO

Neste capítulo contextualiza-se o aparecimento de mecanismos de comunicação entre veículos e peões. Começa-se por fazer uma breve introdução ao tema, juntamente com a explicação de alguns elementos que impulsionam pesquisas efetuadas nesta área. Posteriormente é dada uma perspetiva do que se pretende como objetivos deste trabalho, bem como métodos de investigação utilizados, finalizando com uma descrição relativa à estrutura desta dissertação.

1.1 CONTEXTUALIZAÇÃO

No nosso dia-a-dia muitas vezes surgem situações inesperadas onde a segurança rodoviária é posta em causa. Estas ocorrências podem ser causadas tanto pelos condutores como pelos peões. Considere-se como exemplo o caso em que um peão se encontra distraído a utilizar o telemóvel. Visto que a sua atenção não está focada totalmente na estrada, este ficará mais vulnerável a possíveis acidentes. Além da distração, muitas outras condições podem aumentar a vulnerabilidade dos utilizadores na via pública. Uma outra possível poderá ser a idade. Como se sabe, os tempos de reação aumentam com a idade, sendo que um idoso demora mais a reagir a possíveis situações de perigo do que alguém mais jovem, pelo que este poderá ser considerado um utilizador vulnerável. Adicionalmente as crianças também constituem um tipo de utilizadores vulneráveis, visto que apresentam um certo grau de inexperiência quando deparadas com possíveis situações de perigo, não sabendo muitas vezes como reagir corretamente.

Normalmente para assegurar a segurança rodoviária é necessário garantir que todos os condutores seguem de acordo com as regras de trânsito estipuladas. Ainda mais, para assegurar uma redução significativa de acidentes ocorridos, é necessário garantir que os próprios peões estão constantemente atentos a tudo o que os rodeia bem como preparados para possíveis imprevistos que possam ocorrer, e capazes de reagir em questões de poucos segundos.

Na sociedade atual onde todos estamos constantemente a aceder a redes sociais, ver vídeos, ouvir música, etc, a plena atenção tanto dos peões como dos condutores é facil-

mente perturbada, e esta tendência tem vindo a agravar-se pelo uso deste tipo de equipamentos móveis (telemóveis, *tablets*, *IPods*, ...), que disponibilizam uma série de aplicações de entretenimento que acabam por desviar a atenção dos seus utilizadores para outro tipo de tarefas menos importantes.

Tendo todos estes fatores em consideração, rapidamente se identifica uma necessidade para algum tipo de *software* ou aplicação que seja capaz de indicar e avisar os seus utilizadores para possíveis cenários de risco, antes que os acidentes aconteçam.

1.2 MOTIVAÇÃO

Os utilizadores mais vulneráveis são definidos como aqueles que por algum critério específico estão mais suscetíveis a situações de acidentes. De acordo com as estatísticas emitidas pela *Organização Mundial de Saúde* (OMS) em Violence et al. (2013), mais de 3000 pessoas morrem diariamente devido a condução perigosa, sendo que metade destas mortes são de utilizadores vulneráveis (adiante designados como *VRUs*). A principal razão para a existência deste grande número de acidentes, relaciona-se com a incapacidade dos peões detetarem e perceberem situações de perigo a tempo útil para conseguirem implementar as respostas adequadas, prevenindo eventuais acidentes.

A atenção dada ao peões e a metodologias de previsão de possíveis colisões, com recurso a técnicas de visão por computador, tem vindo a aumentar, tal como nos indicam Gandhi and Trivedi (2007). No entanto, estas abordagens baseadas em sensores não se comportam de forma aceitável em cenários onde a visibilidade é reduzida, por exemplo durante a noite, condições climatéricas adversas, o peão a uma distância muito grande do veículo ou ainda numa situação *Non line-of-sight* (NLOS), onde não existe um ângulo de visão direto entre o veículo e o peão. Desta forma torna-se imperativo desenvolver novas tecnologias que consigam preencher esta falha existente de modo a reduzir consideravelmente as fatalidades ocorridas nas estradas.

1.3 OBJETIVOS

O objetivo principal desta dissertação passa pela correta implementação de um sistema de alertas (*software* e *WebApp*) capazes de alertar os utilizadores de risco para eventuais cenários de perigo rodoviário. Pretende-se chegar a um protótipo totalmente funcional que seja capaz de prever possíveis cenários de perigo rodoviário antes que tais situações se manifestem. Este protótipo deverá funcionar tanto em cenários virtuais como em ambientes de teste híbridos.

Para a correta conclusão deste projeto, é expectável que se desenvolva uma aplicação demonstrativa, de utilização intuitiva que possa facilmente ser adotada por todo o tipo de

utilizadores de equipamentos móveis. É necessário que esta seja flexível para lidar com todo o tipo de sistemas por parte dos seus utilizadores. Deverá ainda conseguir lidar com diferentes protocolos de troca de informação. Visto que a aplicação é definida para dispositivos portáteis, é necessário que se consiga chegar a uma utilização reduzida da bateria destes equipamentos, para aumentar a sua longevidade. (Lee and Kim, 2016)

Outro aspeto essencial a ser desenvolvido, será um algoritmo de previsão de situações de risco (o mais correto quanto possível), que seja eficiente. É ainda necessário garantir que as informações que provêm de outros agentes em todo o sistema cheguem aos seus supostos destinos e de forma confiável. Finalmente, é imperativo que esta informação chegue aos equipamentos do utilizador em tempo útil.

Concluindo e reforçando, é necessário que esta aplicação tenha uma interface simples e um custo de utilização o mais baixo possível para que possa ser adotada por um grande número de pessoas. Este aspeto é realmente importante porque quanto mais pessoas utilizarem a aplicação, mais informação poderá ser utilizada para o cálculo possíveis rotas de colisão e outros eventuais cenários de perigo.

1.4 INVESTIGAÇÃO

Neste projeto de dissertação seguiu-se uma metodologia de investigação baseda no desenho, implementação e teste de um protótipo completamente funcional. Dada a sua natureza, pressupõe-se uma investigação inicial suportada em trabalhos relacionados. Portanto inicialmente começou-se por efetuar uma pesquisa intensiva de artigos que de alguma forma ataquem os problemas que nos propomos a resolver, ou artigos que de alguma forma forneçam informações essenciais para ajudar ao desenvolvimento do sistema de alerta.

Após esta fase, procurou-se analisar e desenvolver um mecanismo simples para troca de informações entre os veículos e os dispositivos móveis dos peões, assim como se tentou perceber qual a melhor forma para alertar todos os seus utilizadores que se encontram em risco. Para o protótipo inicial, é essencial desenvolver um algoritmo que seja inicialmente simples, mas que facilmente se consiga acrescentar complexidade aumentando por consequência o número de cenários de perigo detetados.

Desenvolvida a aplicação e testado o algoritmo de previsão de situações de perigo, passou-se para a fase de testes. Aqui procurou-se melhorar o algoritmo até ao ponto em que seja possível determinar eventos mais complexos, com mais intervenientes, a uma distância maior, etc. Estes testes foram efetuados em ambientes virtuais e em ambientes híbridos, tendo-se ainda criado todas as condições para podermos chegar a casos de teste para utilização em situações práticas da vida real.

Durante o desenvolvimento foi necessário ter em consideração que os equipamentos que correm esta aplicação são equipamentos móveis, com limitações de bateria, capacidade de

alcance de antenas e grandes atenuações em cenários onde estes se encontrem em mochilas, bolsos, etc. É preciso ter todos estes estes elementos em consideração para chegar à melhor solução possível.

1.5 estrutura da dissertação

Esta dissertação está estruturada em seis capítulos, sendo a estrutura e conteúdo individual de cada um a seguinte:

O primeiro capítulo foca-se na introdução do tema desta dissertação, sendo feita a contextualização do tema, a motivação para a escolha do mesmo, os principais objetivos a cumprir, um resumo simples de alguns métodos de investigação utilizados e finalmente uma pequena descrição da organização da dissertação.

O segundo capítulo apresenta o conhecimento adquirido com referência a artigos relacionados com o tema. É feita uma introdução a alguns conceitos considerados fundamentais a reter, para que um utilizador que não esteja familiarizado com o tema consiga entender um pouco mais do que se falará posteriormente. Neste capítulo faz-se, ainda, uma classificação entre os artigos estudados. Esta divisão deve-se ao facto das soluções encontradas se focarem mais nos mecanismos de comunicação ou terem como objetivo encontrar o algoritmo que melhor prevê situações de perigo.

O terceiro capítulo é utilizado para descrever o problema em mãos, bem como os principais desafios encontrados. Neste capitulo apresenta-se a abordagem proposta para uma solução a este problema, bem como eventual arquitetura do sistema desenvolvido.

O quarto capítulo terá como foco, o desenvolvimento da solução, onde se apresentam as decisões tomadas, o método como a implementação foi realizada e os resultados obtidos.

O quinto e penúltimo capítulo é onde se apresentam os casos de estudo e experiências efetuadas em ambientes de teste. Aqui se descreve o *setup* inicial para testes, os resultados obtidos e finaliza-se com uma discussão relativa a estes últimos.

Finalmente no último capítulo apresentam-se conclusões ao trabalho realizado, bem como perspetivas de trabalho futuro para dar continuidade a este projeto.

2

ESTADO DA ARTE

Neste capítulo apresentam-se os sistemas de aviso em ambientes rodoviários e as tecnologias e protocolos de comunicações que os suportam. Inicialmente apresenta-se o conceito de VRU e de seguida algumas das tecnologias de comunicação sem fios que podem ser utilizados nos avisos em ambiente rodoviário.

Finalmente apresenta-se algum trabalho relacionado, organizado em duas partes: trabalhos mais focados nas tecnologias e protocolos de comunicação e trabalhos mais focados nos algoritmos a utilizar no sistema de avisos rodoviários.

2.1 UTILIZADORES RODOVIÁRIOS VULNERÁVEIS (VRU)

O primeiro conceito que se deverá tomar em consideração é o de *Vulnerable road users* (VRU). Habitualmente refere-se a VRUs quando se identificam utilizadores que se encontram num estado desprotegido a nível de tráfego. Esta tipologia engloba peões, ciclistas e motociclistas. Contudo é possível considerar outros aspetos para determinar o alvo do que se entende como VRU. Se for considerada a idade, então um VRU refere-se a crianças e a idosos, dada a inexperiência de reação por parte das crianças e a deterioração da capacidade de resposta de alguns idosos. Outra abordagem poderá ser relativa à velocidade do transeunte, aqui a classificação ocorre de forma semelhante à efetuada quando se trata do nível de proteção, mas atribui aos motociclistas um nível intermédio de vulnerabilidade. Assim se entende que este termo dirá respeito a todos os utilizadores da via pública que de certa forma tenham as suas capacidades de resposta condicionada, tendo em conta possíveis cenários de perigo iminente.

2.2 TECNOLOGIAS E PROTOCOLOS DE COMUNICAÇÃO

2.2.1 *Wi-Fi Direct*

A primeira tecnologia de comunicação a abordar é conhecida como *Wi-Fi Direct*, e tem como foco principal melhorar e facilitar a comunicação direta entre todo o tipo de dispositivos.

O *Wi-Fi Direct* é ainda uma tecnologia com capacidade de causar grandes impactos. Tal facto deve-se não só á elevada quantidade de equipamentos com acesso a tecnologias *Wi-Fi* atualmente disponíveis, como o também ao facto de esta poder ser implementada em *software* (não sendo necessárias alterações ao nível dos equipamentos). Note-se que esta tecnologia permite comunicações entre todos os dispositivos equipados com tecnologias *Wi-Fi*, não só de equipamentos produzidos pela mesma marca mas sim entre marcas distintas.

A utilização de *Wi-Fi Direct* premite criar grupos *Peer-to-peer* (P2P), que na sua essência se comportam de forma equivalente à infraestrutura de rede *Wi-Fi* convencional, com a diferença de que os dispositivos se podem conectar diretamente uns aos outros, estabelecendo as suas próprias comunicações sem ser necessário, por exemplo, um *router*. Um *Peer-to-peer group owner* (P2P-GO) define um dispositivo que implementa uma funcionalidade semelhante à existente num ponto de acesso à rede, enquanto todos os outros equipamentos de um grupo P2P agem como clientes, acabando por adicionar um certo nível de controlo relativamente às comunicações que serão efetuadas. Uma vez que os papéis assumidos pelos dispositivos não são estáticos, quando dois equipamentos detetam a presença um do outro, negoceiam inicialmente o seu papel na comunicação de forma a que se possa estabelecer um grupo P2P. Uma vez formado o grupo, é possível que outros clientes se juntem, como ocorreria numa rede comum, permitindo que novos equipamentos possam comunicar com os restantes elementos do grupo P2P. Esta topologia pode ser observada pelo esquema da figura 1 disponibilizado no trabalho de Camps-Mur et al. (2013).

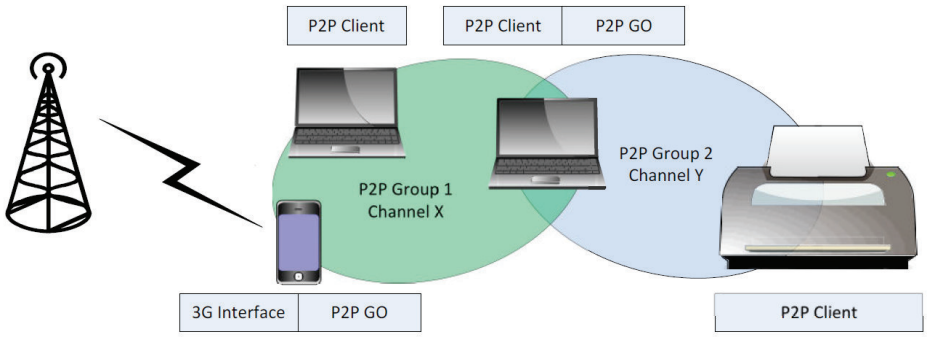


Figura 1.: Esquema de um grupo *P2P* com a co-existência de múltiplos *P2P-GO* (Camps-Mur et al., 2013)

Para que se consiga utilizar comunicação em grupos P2P, (Camps-Mur et al., 2013) sugere que o elemento P2P-GO necessita de implementar um servidor *Dynamic Host Configuration Protocol* (DHCP). Este protocolo é habitualmente utilizado para atribuir dinamicamente endereços *IP* a todos os equipamentos conectados ao dispositivo que disponibiliza este serviço. Permite ainda que dispositivos requisitem os endereços *IP* de outros dispositivos automaticamente, sem que seja necessária a intervenção de um administrador de rede. No

entanto, na eventualidade de não estar disponível um servidor DHCP, a comunicação seria possível, sendo para tal necessário configurar manualmente os *IP*s de cada elemento, neste caso apenas se conseguiria comunicar dentro da rede local.

2.2.2 *Bluetooth Low Energy (BLE)*

Outra possibilidade a ser considerada na comunicação entre equipamentos é a tecnologia *bluetooth*, mais concretamente a recente *Bluetooth Low Energy* (BLE). O BLE é tal como a tecnologia *bluetooth* original, um protocolo de comunicação bastante utilizado. As suas áreas de aplicação variam entre dispositivos *Internet of Things* (IoT), aparelhos *fitness*, relógios, transferência de ficheiros entre equipamentos, etc. Este novo BLE apresenta algumas melhorias relativamente à tecnologia anteriormente existente, e poderá ser utilizado para comunicações entre todo o tipo de equipamentos (no nosso caso, automóveis e dispositivos móveis). O BLE apresenta uma distância de comunicação teórica superior a 100m, contrariamente ao seu antecessor, uma diminuição de 95% nos tempos médios de latência e uma redução enorme nos consumos energéticos, sendo portanto uma tecnologia que dada a sua constante evolução, deverá ser tida em consideração na temática das comunicações entre equipamentos, como alternativa ao *Wi-Fi Direct* e aos grupos P2P.

2.2.3 *Long-Term Evolution (LTE)*

Considera-se ainda que pode ser necessário implementar comunicações entre dispositivo distintos de acordo com requisitos diferentes (por exemplo, capacidade de comunicação a longas distâncias), para tal faz-se uma referência ao padrão *Long-Term Evolution* (LTE). O LTE define um dos padrões de comunicação para tecnologias sem-fio (*wireless*), que possibilita uma maior capacidade e velocidade de comunicação entre dispositivos móveis. O LTE permite velocidades de comunicação na ordem dos 100 Mbps para *downloads* e 30 Mbps para *uploads*, bem como uma latência reduzida e uma maior escalabilidade. É compatível com as tecnologias anteriores GSM e UMTS, fator importante que ajudou a sua rápida difusão pelos dispositivos móveis. Este padrão normalmente é fornecido por operadoras de rede, sendo que estas ficam responsáveis por efetuar o encaminhamento da comunicação entre dispositivos.

2.2.4 *Vhicle-to-everything (V2X)*

No contexto das comunicações em estrada, podemos definir vários tipos de de comunicação, pelo que é necessário esclarecer conceitos como *Vehicle-to-everything* (V2X), *Vehicle-to-pedestrian* (V2P), *Vehicle-to-vehicle* (V2V), *etc*. A comunicação V2V define um aspeto muito

importante no que toca à comunicação entre veículos. Este mecanismo de comunicação está fortemente focado na segurança pública e pessoal dos cidadãos e tal como o nome indica entende tudo o que compreende comunicação direta entre veículos distintos. Exemplos de protocolos de comunicação entre veículos que agem de acordo com o modelo P2P visto anteriormente, pode ser as comunicações *Dedicated Short Range Communications* (DSRC). Estas compreendem comunicações *wireless* unidirecionais ou bidirecionais, que variam entre intervalos de distância curtos a médios. Estas normas foram desenvolvidas especificamente para a utilização veicular e seguem todos os protocolos e padrões da *International Organization for Standardization* (ISO) definidos para tal.

As comunicações V2X suportam diferentes tipos de aplicações: assistentes de navegação, serviços de informação, entretenimento, *infotainment*, auxiliar de localização de peões, avisos de acidentes, entre outros. Engloba todo o tipo de comunicação entre veículos e outros equipamentos, por exemplo: equipamentos móveis (comunicação V2P com pedestres), veículos (comunicação V2V com outros veículos), infraestruturas (comunicações V2I para infraestruturas como *Roadside Unit* (RSU)), casas (comunicações *Vehicle-to-home* (V2H)). Entenda-se assim que quando se menciona comunicações V2X diz-se que a origem ou destino da comunicação é o veículo, podendo o outro equipamento ser qualquer tipo de dispositivos.

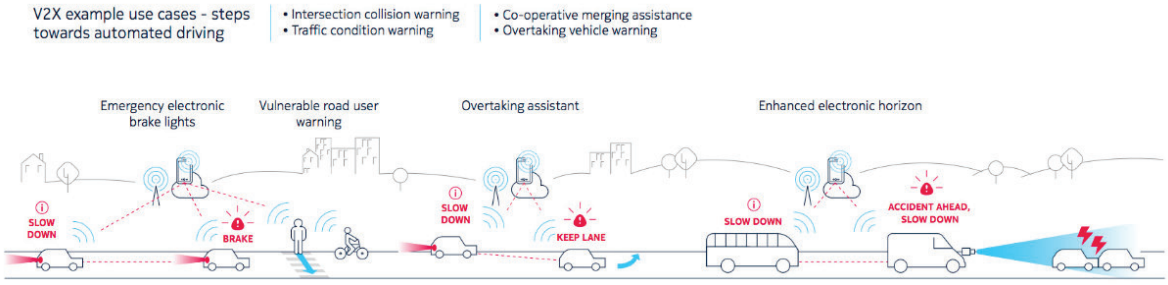


Figura 2.: *Use cases* para a comunicação V2X [NOKIA] (acedido em novembro 2018)

A figura 2 dá exemplos de casos de utilização V2X, para diferentes aplicações como aviso de localização de peões, acidentes, etc.

Já se introduziram tecnologias para comunicação entre equipamentos como BLE, LTE e grupos P2P. Também se explicaram alguns conceitos gerais como V2X e VRUs.

Finalmente introduz-se muito rapidamente conceitos de formatos para troca de informação. Existem dois tipos de formatos para esta troca de mensagens, o *Cooperative Awareness Message* (CAM) e o *Decentralized Environmental Notification Message* (DENM), sendo que ambos se enquadram dentro das comunicações DSRC.

O formato CAM, visível na figura 3, é conhecido como modelo de *beacon*, isto porque a cada segundo são enviadas 10 mensagens neste formato para que um veículo tenha a possibilidade de se dar a conhecer aos veículos circundantes, enviando informações como a

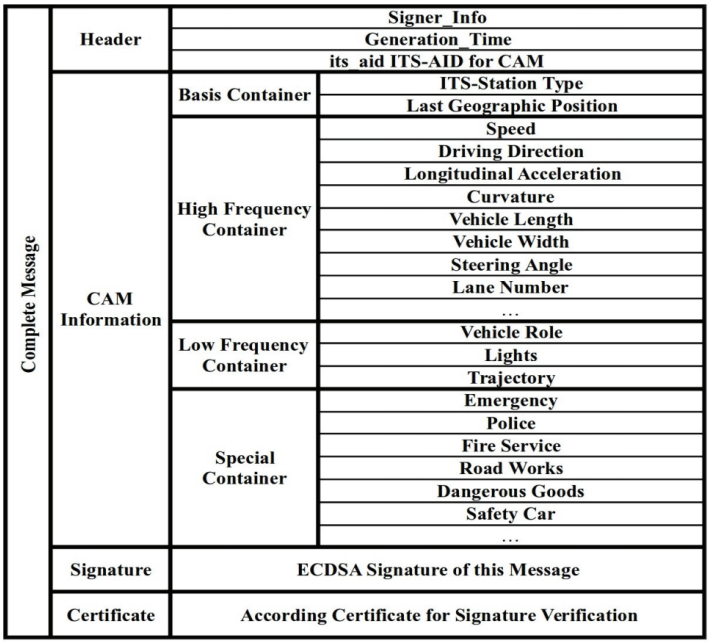


Figura 3.: Conteúdo de uma mensagem enviada com informação no formato *CAM* (Ullmann et al., 2016)

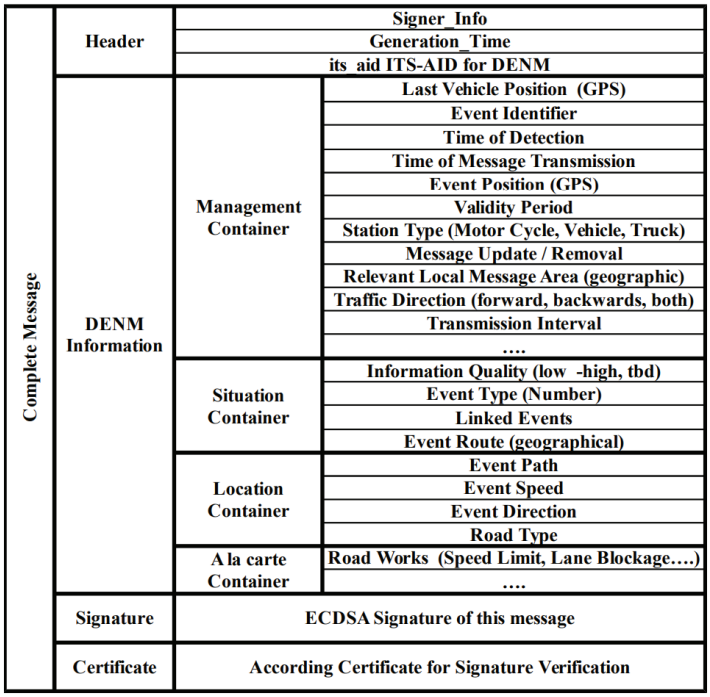


Figura 4.: Conteúdo de uma mensagem enviada com informação no formato *DENM* (Ullmann et al., 2016)

sua velocidade, direção, aceleração, posição, ângulo de viragem, entre outros. Já o formato DENM, visível na figura 4, é orientado a eventos. Desta forma uma mensagem será enviada quanto surgir uma alteração na velocidade, aceleração, direção, trajetória, entre outros.

Finda a introdução de alguns conceitos essenciais para a compreensão desta dissertação, fale-se agora de trabalho realizado nesta área das comunicações veiculares.

2.3 TRABALHO RELACIONADO

2.3.1 *Tecnologias e protcolos de comunicação*

A área da comunicação V2X sempre despertou muito interesse, e continua como motivo de desenvolvimento de uma grande quantidade de trabalhos, propostas e artigos. De seguida apresentam-se alguns trabalhos importantes que deverão ser considerados quer por se tratarem de artigos fundamentais para a compreensão de temas presentes nesta dissertação quer por se tratarem de artigos que impulsionaram a pesquisa efetuada durante a fase inicial deste projeto.

Ao longo da pesquisa efetuada, por trabalhos relacionados com o tema e com implementações próximas do que se pretende fazer, notou-se que os autores optavam por duas abordagens ao problema das comunicações V2P. Alguns focam-se maioritariamente na parte respeitante à comunicação entre veículo e dispositivos móveis, enquanto outros dão mais ênfase aos algoritmos desenvolvidos para uma maior correção de previsões efetuadas, não sendo explícita a forma como a comunicação entre veículo e equipamento de utilizador final (neste caso um peão) é efetuada.

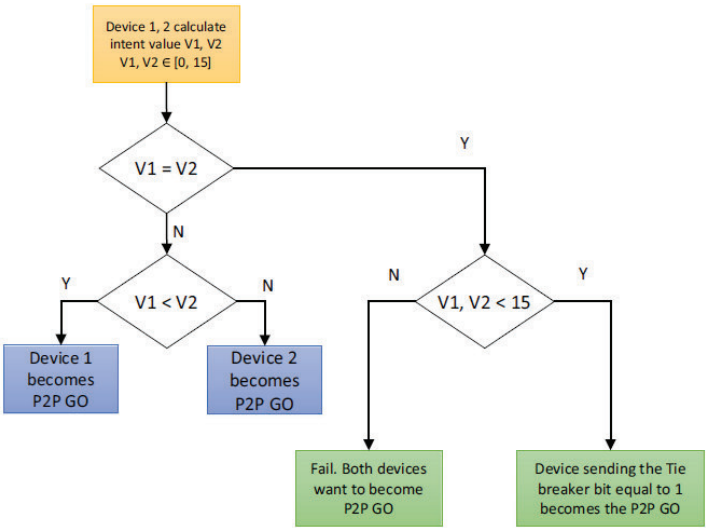


Figura 5.: Eleição do elemento *P2P-GO*

Para que seja possível comunicar eventuais situações de perigo é necessário garantir que os mecanismos de comunicação entre dispositivos sejam rápidos, eficientes e resistentes a possíveis falhas de transmissão.

O trabalho de Lee and Kim (2016) passa por utilizar *Wi-Fi Direct* como meio para transmissão de informações entre utilizadores em grupos P2P. Lee and Kim (2016) procuram uma forma de difundir mensagens de aviso de maneira eficiente que necessitem de menos consumos energéticos, procuram ainda evitar o envio de avisos em duplicado (colisões de envio) e finalmente aumentar a precisão de avisos transmitidos. Tudo isto é feito através de grupos P2P, Lee and Kim (2016) sugerem eleger um P2P-GO que tomará as decisões relativamente à transmissão de avisos. Este elemento fica encarregue da comunicação entre os veículos e os restantes peões e é determinado de acordo com um valor especifico pré-determinado, conhecido por *intent* ou intenção. Um esquema simples deste método de eleição poderá ser observado na figura 5.

O artigo de Seo et al. (2016) começa com uma discussão relativa à importância de serviços V2X (onde se enquadram as comunicações V2P e P2V), de seguida introduz-se o conceito de LTE segundo o *Third Generation Partnership Project* (3GPP) e ataca-se os principais desafios aqui encontrados: uma grande liberdade de movimentação por parte dos veículos; a elevada densidade de veículos em certas zonas; considerações técnicas gerais. No que toca a políticas de utilização de frequências para comunicação, (Seo et al., 2016) distingue várias possibilidades diferentes, visíveis pela figura 6: Utilizar espectros diferentes para a mesma operadora LTE de modo a não colidir com o espectro de operadoras diferentes; utilizar apenas um espectro de frequências para que várias operadoras consigam controlar a comunicação *device-to-device* (D2D); possibilitar apenas um espectro de frequências para todo o tipo de comunicação, seja entre utilizadores ou entre veículos; considerar que não existe cobertura para LTE sob estas condições. Finalmente este artigo faz uma breve comparação entre a proposta comunicação V2P LTE e a existente comunicação DSRC.

Poderá ainda ser interessante considerar infraestruturas como mecanismos para auxiliar a comunicação, desta forma Scholliers et al. (2017) descrevem o desenvolvimento de uma arquitetura que integre VRUs em sistemas *Intelligent Transportation Systems* (ITS) cooperativos, denominados por *Cooperative Intelligent Transportation Systems* (C-ITS). O problema por eles estudado recai em dois casos distintos: deteção de VRUs que se encontrem em possíveis situações de colisão; aviso de risco de colisão de acordo com velocidade e trajetórias dos utilizadores. Este artigo toma em consideração a interação com infraestruturas que se possam encontrar perto da via pública, como sinais de trânsito. (Scholliers et al., 2017) define três componentes funcionais essenciais para a implementação da arquitetura. A nível de VRU é necessário um transmissor (VRU-T ou uma etiqueta eletrónica capaz de enviar dados), uma estação de sistemas ITS capaz de comunicar através de ITS-G5, a capacidade de obter dados do sistema elétrico do veículo e a capacidade de este comunicar

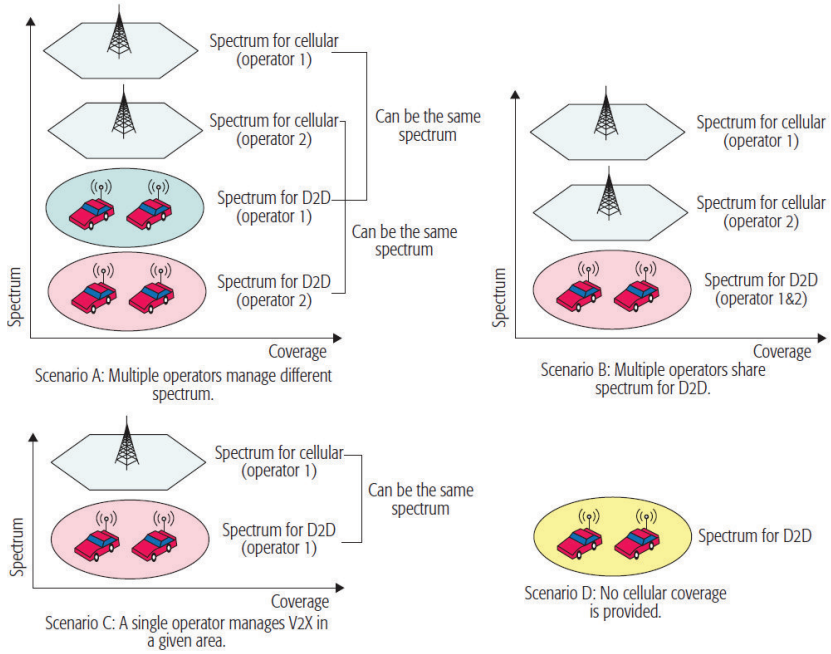


Figura 6.: Exemplos de opções de espectros para emissão de comunicações *V2X*

através de redes móveis. A nível do veículo será necessário uma estação de ITS, capacidade de comunicação entre dispositivos ou veículos, acesso ao sistema de sensores e um sistema de localização de transmissores de VRU. A nível de equipamentos em via pública também seria necessário a existência de estações ITS, controladores de luzes de trânsito e equipamentos capazes de localizar etiquetas eletrónicas dos VRUs. Adicionalmente seria necessário todo um sistema capaz de lidar com todas as informações adquiridas, fazendo a sua triagem e pós distribuição de avisos/indicações.

Camps-Mur et al. (2013) demonstram as funcionalidades permitidas pelo *Wi-Fi Direct* e testam com casos de utilização práticos e reais, tanto os atrasos presentes nas comunicações entre dispositivos móveis, como a performance dos protocolos de poupança de energia que esta tecnologia traz consigo associada. Camps-Mur et al. (2013) identificam uma tecnologia semelhante a esta, que já existia anteriormente (*ad-hoc*), mas que dada a sua pouca utilização acabou por não evoluir conjuntamente aos requisitos dos utilizadores. Foram analisados dois tempos distintos para formar grupos P2P, nomeadamente *delays* de descoberta (associados à descoberta de novos equipamentos nas proximidades) e *delays* de formação de grupo (associados à conclusão do acordo entre os papeis que cada equipamento terá para a comunicação). O problema aqui é que o tempo de descoberta é elevado (na ordem dos segundos) isto porque quando se lida com equipamentos cuja bateria é limitada, há que fazer uma boa gestão desta, assim temos de fazer uma procura por equipamentos em intervalos de tempo relativamente grandes, que poderão ir de encontro com os tempos

críticos identificados nos outros artigos. Apesar da tecnologia presente em (Camps-Mur et al., 2013) ser interessante para comunicar informações entre dispositivos, teria de sofrer algumas alterações para que seja possível ser corretamente implementada no nosso caso de estudo.

Eyobu et al. (2017) abordam um aspeto fundamental das comunicações V2X, a enorme quantidade de mensagens que são trocadas num curto espaço de tempo. O trabalho passou por desenvolver uma metodologia que ataca este problema enquanto maximiza o tempo de vida dos equipamentos móveis dos peões através de uma abordagem semelhante à proposta pelo *Institute of Electrical and Electronics Engineers* (IEEE) 802.11. (Eyobu et al., 2017) sugere que uma abordagem de *broadcast* deverá ser adotada para transmitir informação entre os veículos e os pedestres, enquanto que os pedestres utilizarão *unicast* para comunicar (apenas quando necessário) com veículos específicos que possam apresentar algum tipo de risco ao utilizador. Adicionalmente este artigo propõe a implementação de um mecanismo que assegura que na eventualidade de ocorrer uma perda de informação/mensagens (por exemplo uma interferência de sinal), e caso um utilizador não receba a informação de nenhum veículo num período de 3 segundos, este emitirá um aviso *broadcast* de prioridade elevada que indica a sua posição, velocidade e direção para veículos próximos. Este artigo introduz um termo diferente especificado como *vehicle broadcast to pedestrian* (Vbr2P).

A figura 7 demonstra uma ilustração do conceito, apenas comunicações provenientes de veículos que existem nos intervalos de distância *do*, *d1* e *d2* conseguirão invocar a transmissão de mensagens por parte dos peões. Esta taxa de transmissão varia conforme o intervalo de distancia entre o veículo e o peão.

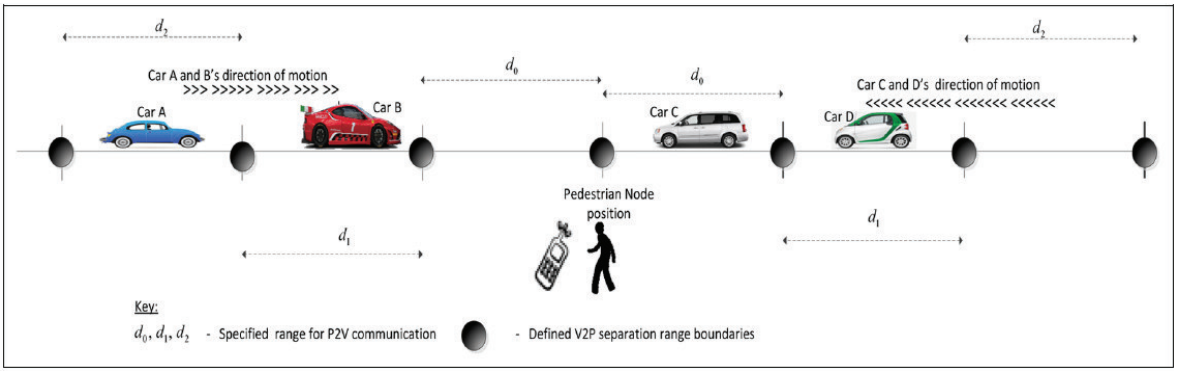


Figura 7.: Ilustração do conceito de *Vbr2P*

Em Jansons et al. (2012), os autores focam-se na comunicação *Vehicle-to-Infrastructure* (V2I) e dão ênfase à forma como a norma IEEE 802.11 para comunicação em *Wireless local area network* (WLAN) suporta a deslocação dos seus utilizadores de rede, para possibilitar um acesso otimizado à rede. Este trabalho apresenta um estudo experimental do IEEE 802.11g num cenário de teste reduzido, recorrendo a equipamentos disponíveis atualmente. Jansons

et al. (2012) desenvolvem um estudo sobre a performance das redes WLAN consoante o número de utilizadores conectados e consoante a distância a que estes utilizadores se encontram do ponto de acesso. Finalmente analisam a conectividade à rede de acordo com variações na velocidade dos veículos.

Finalmente apresenta-se o trabalho de Bagheri et al. (2014). Estes apresentam um trabalho interessante que passa por utilizar redes móveis (com foco em telecomunicações), para possibilitar a comunicação entre veículos e peões de forma a evitar colisões e eventuais cenários de perigo. Atualmente os equipamentos móveis dos VRUs não suportam comunicações sobre a norma IEEE 802.11p, que está otimizada para comunicações entre veículos. Este artigo propõe utilizar redes 3G e LTE para que se consiga estabelecer uma ligação entre estes dois tipos de utilizadores. Um aspeto positivo desta abordagem é o facto de alguns veículo já disponibilizarem comunicação por LTE de origem, outro aspeto positivo prende-se ao facto de que os carros mais antigos (que não têm métodos de comunicação nativos) poderem comunicar através do equipamento móvel do condutor. (Bagheri et al., 2014) demonstra uma abordagem semelhante à que será realizada nesta dissertação.

2.3.2 *Algoritmos para Aplicações de Alerta*

Além do foco dado na vertente de comunicação entre veículos é importante desenvolver o algoritmo de previsão de colisões, visto que este será o responsável por emitir toda a informação principal existente. Desta forma começamos pelo trabalho de Anaya et al. (2014). Estes começam por explicitar o problema que tentam resolver: as cerca de 3000 mortes diárias que ocorrem devido a condução descuidada. Referem ainda que uma das causas principais para este elevado número é o facto dos peões, ciclistas, *etc*, denominados de VRUs não serem capazes de prever situações de risco em tempo útil. A abordagem aqui utilizada passa por desenvolver uma aplicação móvel capaz de receber mensagem CAM. O envio destas informações será efetuado por Wi-Fi, onde no lado do veículo tem-se presente a norma do IEEE 802.11g, e no lado do VRU um *tablet Android* (que presumivelmente utiliza a norma IEEE 802.11p). O artigo (Anaya et al., 2014) identifica métricas que deverão ser consideradas para que se consiga realizar uma previsão mais correta de um possível caso de perigo, nomeadamente: distância mínima em que o aviso ainda é útil (considerando tempos de envio, perceção e reação); a probabilidade de um utilizador ser informado antes desta distância mínima; possível área de colisão; *threshold* de tempo de segurança, para evitar falsos positivos. Evidenciam ainda uma previsão de direção, visível na figura 8, que é utilizada para detetar situações de risco até mesmo em situações de curvas acentuadas.

Hussein et al. (2016) propõem uma solução que poderá ser utilizada tanto por veículos autónomos como para veículos convencionais (veículos de autonomia nível 0 ou 1), onde o condutor é estritamente necessário para a condução. (Hussein et al., 2016) utiliza a

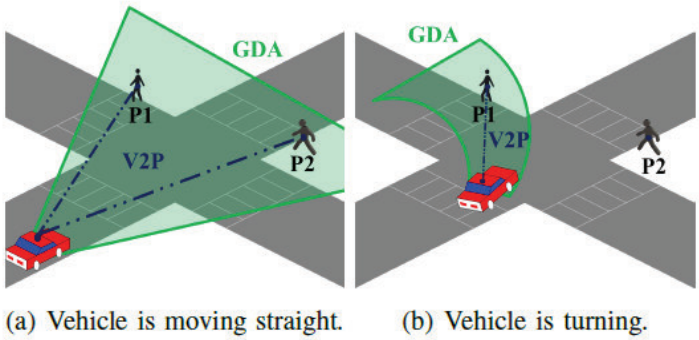


Figura 8.: Previsão da área de destino de um veículo

localização de *Global Positioning System* (GPS), a direção e a velocidade de ambos os intervenientes (veículo e VRU) para calcular um possível ponto de colisão. Pode ser considerado ainda um ângulo de incerteza para a direção de ambos, sendo que a abordagem aqui tomada acaba por ser semelhante à explicada em (Anaya et al., 2014). Este artigo propõe a utilização de um tempo para colisão em contrapartida à distância mínima para aviso em tempo útil. Este artigo fala ainda da utilização de Wi-Fi e 4G para a troca de mensagens. Refere finalmente que os tempos de latência associados à comunicação Wi-Fi são menores que os tempos associados a comunicações por 3G, e explicitam que os mais recentes 4G e 5G permitem uma comunicação mais eficiente entre equipamentos. Hussein et al. (2016) desenvolveram uma aplicação com uma interface simples que poderá ser algo semelhante ao efetuado no decorrer deste projeto. Esta interface é visível na figura 9.

Figura 9.: Exemplo de interface móvel simples

No que toca a colisões entre veículos e pedestres, Flores et al. (2018) propõe um sistema capaz de travar a viatura quando uma situação de risco é detetada. O trabalho aqui re-

alizado introduz os conceitos de *Adaptative Cruise Control* (ACC) que consiste, tal como o nome indica, numa metodologia capaz de gerir adaptativamente a forma de condução de uma viatura, e *Cooperative Adaptative Cruise Control* (CACC) que utiliza a comunicação entre veículos para controlar a velocidade do trânsito em massa, ajudando a melhorar os níveis de congestionamento das estradas, diminuindo potenciais situações de risco. Esta solução é implementada e testada em veículos totalmente autónomos contendo todo o tipo de sensores. A comunicação é efetuada de acordo com a norma IEEE 802.11g por não existir ainda a disponibilidade de IEEE 802.11p entre veículos, no entanto é assumido que, quando disponível, a transição entre IEEE 802.11g e IEEE 802.11p deverá ser bastante imediata. Apesar de bastante interessante, o trabalho de (Flores et al., 2018) é apenas focado a veículos autónomos, dependendo fortemente em informações provenientes de sensores. Deste modo apenas algumas considerações gerais serão utilizadas, contudo é interessante ter em consideração a máquina de estado utilizada nesta arquitetura, uma vez que poderá ser desenvolvido algo semelhante (ver figura 10).

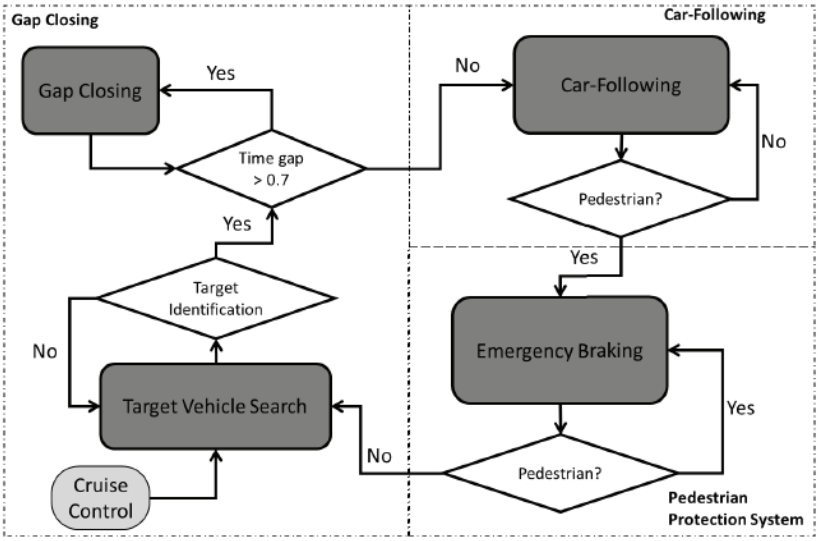


Figura 10.: Máquina de estado do algoritmo de controlo utilizado em Flores et al. (2018)

Em Merdrignac et al. (2017) ocorre uma distinção entre situações de perigo para peões, bem como possíveis soluções. Aqui definem-se dois tipos distintos, situações *Line-of-sight* (LOS) e situação NLOS, esquematizadas na figura 11. As primeiras identificam-se por situações em que se podem evitar acidentes por recorrer a meios como sensores, portanto tal como o nome indica são casos onde existe uma linha de visão entre o veículo e o VRU. Este caso pode, no entanto, referir-se ao campo de visão dos sensores identificado por *point-of-view* (POV) e não à linha de visão do condutor. O segundo caso identifica situações onde não existe a possibilidade de observar diretamente o peão, que por conseguinte apresentam desafios maiores à sua correta implementação. Merdrignac et al. (2017) admitem que este

tipo de comunicação necessita de um *delay* reduzido, sendo que por esta razão ponderou-se utilizar Wi-Fi baseado no protocolo IEEE 802.11p ou comunicações D2D sobre o protocolo LTE. Apesar disto o artigo (Merdrignac et al., 2017) optou por utilizar comunicação Wi-Fi IEEE 802.11p tanto no lado do veículo como no lado do VRU. Note-se que para tal foi necessário equipamentos móveis específicos, especialmente desenvolvidos para teste, uma vez que os equipamentos móveis atuais apenas comunicam utilizando a norma IEEE 802.11g.

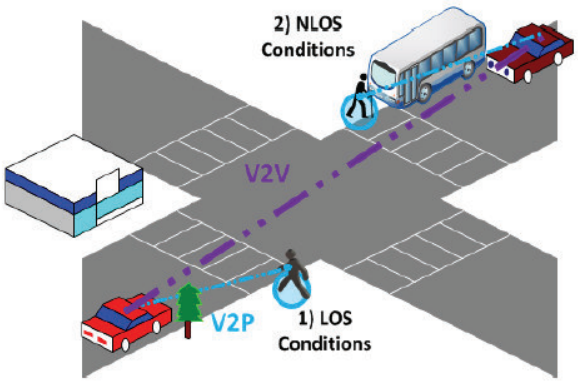


Figura 11.: Exemplo de uma situações *LOS* e *NLOS*

2.4 Discussão Final e Síntese

Neste capítulo introduziram-se essencialmente alguma terminologia e alguns conceitos fundamentais necessários no contexto do trabalho desenvolvido. Estes conceitos são certamente utilizados e explicitados ao longo desta dissertação.

Inicialmente introduziram-se os conceitos de P2P, P2P-GO., BLE, LTE, V2X e VRU bem como os formatos de trocas de informação DSRC utilizando CAM e DENM.

De seguida, e após um estudo detalhado da informação disponível nos demais artigos apresentados, conclui-se que este ramo de comunicações V2X, mais propriamente o que compreende as comunicações V2P é um assunto ainda em desenvolvimento. Ainda existe espaço para melhoramentos no que toca a protocolos de comunicação eficientes, que garantam a existência de entrega total de mensagens de aviso, de boas gestões energéticas, envios de informação duplicada, *etc*. Desta forma o objetivo inicial desta dissertação passa por desenvolver um *software* capaz de comunicar por tecnologias *bluetooth* com o equipamento do condutor, sendo que este mesmo dispositivo móvel será encarregue de enviar por LTE informações importantes com destino aos dispositivos móveis dos VRUs, que por sua vez notificarão os utilizadores de eventuais acontecimentos de risco.

3

SISTEMA DE ALERTAS RODOVIÁRIOS: UMA PROPOSTA DE SOLUÇÃO

Neste terceiro capítulo descreve-se o desenvolvimento e a abordagem seguida na resolução do problema em mãos, assim como a arquitetura e desafios encontrados.

Começa-se por apresentar a proposta de solução desenvolvida juntamente com alguns dos desafios encontrados ao longo do seu desenvolvimento. Posteriormente discute-se uma abordagem de arquitetura utilizada bem como uma explicação da arquitetura e todos os componentes do sistema: cliente, servidor e sistema de dados. Conclui-se com uma apresentação relativa a desafios adicionais que foram encontrados ao longo do desenvolvimento.

3.1 OBJETIVO E REQUISITOS

O trabalho realizado no contexto desta dissertação tem o intuito de procurar diminuir o número de fatalidades ocorridas em situações de perigo nas nossas estradas, e tem como objetivo providenciar um método auxiliar de indicação destas mesmas situações de perigo. A ideia aqui patente é a de alargar os alertas existentes em veículos mais atuais para os dispositivos móveis dos peões, ou em sentido contrário, comunicar a presença, posição ou até direção de um peão ao veículo.

Para tal é necessário desenvolver uma plataforma que permita que os seus utilizadores partilhem e recebam as localizações de outros clientes do sistema. É essencial que esta suporte múltiplos utilizadores em simultâneo e que consiga garantir uma troca de informação em tempo útil. A plataforma final deverá ser de simples utilização para que tenha uma maior adoção. Além disto deverá de alertar corretamente a existência de potenciais cenários de perigo.

3.2 SOLUÇÃO E DESAFIOS

Para desenvolver a solução proposta, capaz de responder aos requisitos que foram definidos anteriormente, foi necessário pensar nos aspetos essenciais para resolver o problema.

A ideia principal era que esta plataforma informasse os seus utilizadores sempre que um cenário de perigo fosse detetado. Assim, rapidamente se entende que é fundamental algum tipo de comunicação, nem que seja apenas para alertar os utilizadores finais (pedestres).

Chega-se assim a outro desafio. Como saber onde se encontram os utilizadores (pedestres, outros URVs, motoristas, etc)? Há que arranjar alguma forma segura de fazer com que estes partilhem a sua localização. Inicialmente pensou-se utilizar comunicação P2P entre veículos, sendo que os veículos mais modernos e com capacidade de comunicação certamente também haveriam de apresentar alguma forma de obter a sua própria localização. Neste caso metade do problema estaria resolvido, agora apenas resta saber como fazer com que a comunicação entre veículos e pedestres seja possível. Relembre-se que os veículos continuariam a partilhar a sua localização de forma P2P.

É aqui que surge um novo desafio. Ambos os veículos e os *smartphones* dos pedestres necessitam de apresentar algum tipo de capacidade de comunicação *wireless*, mas estas encontram-se em bandas diferentes, sobre normas distintas, não sendo possível a comunicação direta entre si. Uma solução para este problema passa por conseguir arranjar dispositivos capazes de utilizar a norma *802.11g* em paralelo com a *802.11p*, nos veículos, fazendo com que estes consigam comunicar com os equipamentos dos pedestres de forma direta. Claramente esta não é a melhor abordagem dado que, para utilizar a plataforma, seria necessário comprar estes dispositivos e respetivas antenas. Daqui, o cenário mais provável de acontecer seria o de que a plataforma não teria utilizadores suficientes para ser adotada em massa pelos seus potenciais utilizadores. Este requisito vai de encontro ao que tinha sido proposto e definido inicialmente como ideia chave de todo este projeto: o desenvolvimento de uma plataforma de custo muito reduzido para o utilizador final.

Foi aqui que ocorreu uma mudança na forma de pensamento. Já que também seria necessário obter a localização dos pedestres (utilizando indicações GPS do *smartphone*), porque não fazer a mesma coisa para os veículos? Assim seria possível, com um mesmo dispositivo, agir como ambos os intervenientes dos cenários de perigo que se pretendem detetar.

Nesta nova fase de desenvolvimento da solução, ponderou-se utilizar um dispositivo adicional, de custo bastante reduzido, para comunicar dados relativos ao veículo com uma maior precisão. O aparelho em questão (visível na figura 12) seria ligado à entrada *On-board Diagnostics Version 2* (OBD2) dos veículos e emparelhado por *bluetooth* ao equipamento do condutor.

Adquiriu-se um destes equipamentos para testar a viabilidade da ideia e rapidamente se percebeu que os veículos por vezes têm os acessos à porta OBD2 bastante condicionados, em

Figura 12.: Equipamento utilizado como transmissor OBD2 para *Bluetooth*

localizações de elevada dificuldade de acesso. Além disto, quando utilizado corretamente, e apenas nos veículos que disponibilizam informação relativa à sua posição GPS, a posição obtida pelo próprio GPS da viatura apresenta, nas mesmas condições, um erro na ordem dos 15 metros enquanto que o erro do GPS de um equipamento do tipo *smartphone* é na ordem dos 5 metros. Note-se que o teste efetuado foi no mesmo ambiente, em campo aberto, sem obstruções de edifícios ou algum tipo de cobertura dos equipamentos. Dada que esta hipótese não apresentava vantagens aparentes e impunha um custo de utilização adicional (embora reduzido), acabou por ser descartada na sua totalidade.

Então o que ficou decidido? Estipulou-se que seriam utilizados os equipamentos móveis dos utilizadores (*smartphones*) de forma a obter a localização destes com uma maior precisão. Esta abordagem é bastante conveniente visto que agora será possível comunicar de forma *adhoc* entre equipamentos. Todos os utilizadores da rede estariam a comunicar sobre a norma IEEE *802.11g*, deixando de existir comunicação do tipo *802.11p*.

O maior problema havia sido ultrapassado! Consegue-se comunicar entre equipamentos de condutor e de pedestre, como obter a localização com alta precisão relativa aos utilizadores. Considere-se que para já ainda não existe qualquer tipo de mecanismo que permita a distinção entre os dois tipos de utilizadores distintos da plataforma. Este problema foi empurrado até ao algoritmo de decisão, que decide se uma mensagem recebida é de um pedestre ou de um condutor de acordo com alguns dados recolhidos, nomeadamente a sua velocidade. O algoritmo será explorado mais em diante.

Ultrapassados estes desafios iniciais, deparou-se com um novo problema que havia passado despercebido. E se os utilizadores tivessem uma distância considerável entre si? Digamos de cerca de 300 metros. Seria possível comunicar a sua posição e receber informação de outros utilizadores do sistema? A resposta é não, os *smartphones* conseguem comunicar a uma distância de até cerca de 100 metros, se estes se encontrarem em condições

Edit	Bluetooth	Wi-Fi
Hardware requirement	Bluetooth adaptor on all the devices connecting with each other	Wireless adaptors on all the devices of the network, a wireless router and/or wireless access points
Range	5-30 meters	With 802.11b/g the typical range is 32 meters indoors and 95 meters (300 ft) outdoors. 802.11n has greater range. 2.5GHz Wi-Fi communication has greater range than 5GHz. Antennas can also increase range.
Power Consumption	Low	High
Ease of Use	Fairly simple to use. Can be used to connect upto seven devices at a time. It is easy to switch between devices or find and connect to any device.	It is more complex and requires configuration of hardware and software.
Latency	200ms	150ms
Bit-rate	2.1Mbps	600 Mbps

Figura 13.: Comparação entre distâncias de comunicação *bluetooth* e *Wi-Fi* [Fonte: Diffen]

óptimas de utilização, isto é, no campo de visão do outro equipamento e sem qualquer obstáculo intermédio. Pode ser observada uma amostra de informações comparativas entre as distâncias de utilização de ambas as normas de comunicação *bluetooth* e *Wi-Fi* na figura 13.

Num cenário onde no pior caso ambos os agentes são veículos (podendo ter velocidades elevadas e deslocamentos na mesma direção mas sentidos opostos), esta distância não é suficiente para conseguir prever atempadamente um potencial cenário de risco. A solução aqui passaria novamente pela adoção de antenas semelhantes à demonstrada pela figura 14, não para possibilitar a comunicação entre dispositivos mas para possibilitar a comunicação a distâncias mais longas. É necessário arranjar uma nova alternativa pois já tinha sido observado anteriormente que esta solução impunha um aumento no custo de utilização da plataforma em si. Uma vez mais, denota-se que este aumento no custo poderá levar a uma menor adoção da solução e sendo que os utilizadores são o ponto fundamental do sistema, esta tem que ser bem avaliada antes de qualquer consideração de adoção final.

O procedimento seguinte passou por alterar completamente a forma como a comunicação seria feita. Até agora o sistema dependia de comunicação direta entre os seus utilizadores, mas o que aconteceria se em alternativa se enviasse as informações a um servidor central e

Figura 14.: Exemplo de antena utilizada para comunicação V2X [Fonte: Travel Wire News]

deixar que o servidor reencaminhasse estas aos clientes finais? Neste caso tem-se tanto aspetos positivos como aspetos negativos. Um dos aspetos negativos principais desta solução seria o aumento dos tempos de latência entre clientes, note-se que o dispositivo do condutor que se encontra a 50 metros de um pedestre necessitará de enviar os dados ao servidor, ao que este posteriormente reencaminhará para o pedestre. Além disto a comunicação ficaria dependente de uma ligação à rede, que se torna um requisito adicional. Na solução anterior era possível que estes comunicassem diretamente sem existência de uma ligação à rede, não existindo qualquer tipo de sistema intermediário que induza latência, ou o requisito adicional. No entanto tem-se um aspeto positivo que traz bastantes vantagens, o facto de se puder comunicar virtualmente a qualquer distância. Dada a nova topologia seria possível (apesar não ser prático) comunicar a distância de um veículo que se desloque aqui em Portugal, para qualquer outro país (embora pouco interessante), dependendo sempre de uma ligação funcional à rede (para estabelecer comunicação com o servidor). A figura 15 apresenta a diferença de arquitetura que se retrata aqui, num abordagem cliente / servidor será sempre necessário enviar informação ao servidor para que este redistribua pelos restantes clientes, aqui note-se que os símbolos de computador etiquetados como *Device*, tanto poderão referir a um condutor como a um pedestre.

Existiram algumas dúvidas relativamente à implementação desta solução, principalmente quando se considera que a latência ou atrasos do servidor poderiam não assegurar que a

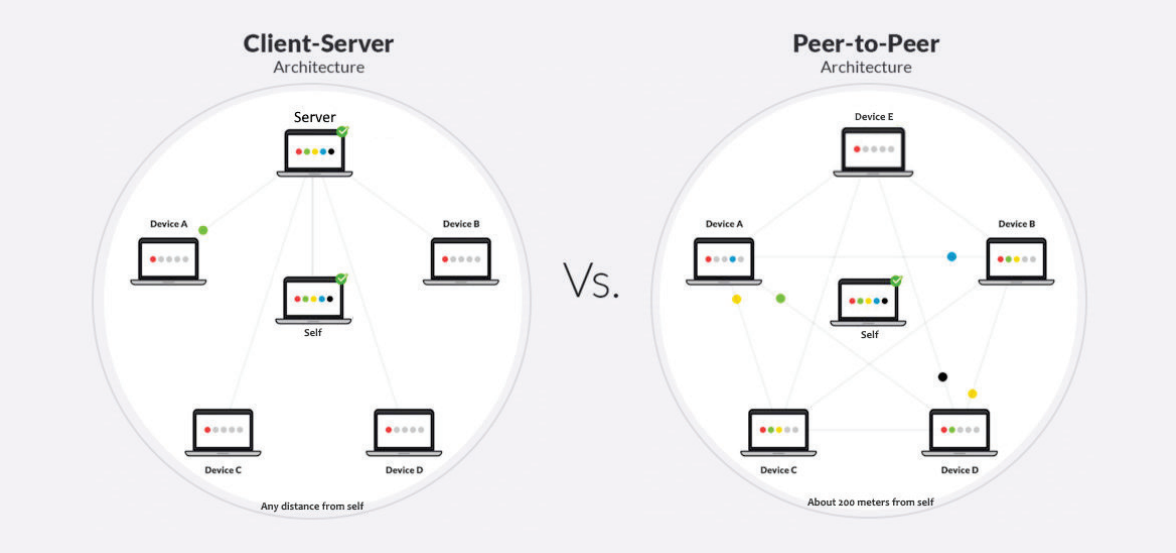


Figura 15.: Diferença entre comunicação P2P e Cliente/Servidor [Fonte adaptada de: Resilio]

mensagem chegasse ao destino a tempo útil. Este possível problema, explicado no próximo capítulo, foi resolvido adicionando métricas que definem quando o envio de cada mensagem é efetuado. Outra das desvantagens, relativa ao requisito de ligação à rede, acaba por não ser assim tão significativa, uma vez que atualmente grande parte dos potenciais condutores e pedestres possuem não só um *smartphone* como também ligação através de dados móveis.

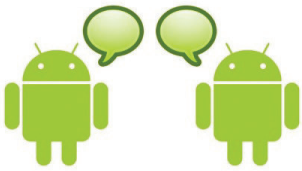
Concluindo optou-se pela solução de comunicação do tipo cliente / servidor, não só porque esta apresenta adições mínimas aos requisitos de utilização da plataforma mas também porque possibilita a deteção de possíveis cenários de colisão a distâncias bastante superiores. Será assim possível aumentar estas mesmas distâncias de deteção para minimizar o acrescido tempo de comunicação que foi introduzido.

Nesta arquitetura, o servidor pode calcular ele próprio potenciais perigos, calculando rotas de colisão entre utilizadores. O algoritmo que faz o cálculo, e que será detalhado no próximo capítulo, faz uso das posições recolhidas dos utilizadores do sistema. O histórico das posições permite estimar uma trajectória previsível e verificar os perigos daí decorrentes. Também é possível que o servidor se limite a recolher e distribuir dados para os clientes, deixando-lhes a tarefa de cálculo de risco de colisão com outros utilizadores. Em ambos os casos se fará uso de um algoritmo que, dadas as informações disponíveis, detecte situações de perigo e emita alertas rodoviários aos utilizadores envolvidos.

3.3 ABORDAGEM ADOTADA

Para dar resposta ao problema de cenários de perigo iterou-se por várias fazes e opções não só conceptuais mas também de desenvolvimento da solução. Inicialmente pensou-se desenvolver aplicações nativas tanto para sistemas *Android* como *IOS*. Para isto já existiam alguns conhecimentos básicos adquiridos anteriormente, utilizando ferramentas como o *Visual Studio* da *Microsoft* que permite a utilização de modelos para o desenvolvimento deste tipo de aplicações para sistemas diferentes que partilhem aplicações com código-fonte semelhantes (exemplo do *Xamarin*).

Após um esboço inicial, foi reparado que implementar todo este sistema impunha a inexistência de falhas ou atrasos no envio de mensagens entre equipamentos (este é um aspeto essencial para o funcionamento da solução). Infelizmente tal não seria assegurado uma vez que ficaríamos dependentes da *Application programming interface* (API) já existente, e que esta apresenta tempos de resposta consoante o sistema em que se encontre. Além disto, como esta ferramenta é parcialmente suportada pela comunidade, uma atualização a meio de desenvolvimento poderia não ser totalmente retrocompatível. Este problema teria peso adicional uma vez que se pretendia ainda utilizar uma outra API para comunicação *bluetooth*, que traria atrasos adicionais. Além disto a compatibilidade com ambos os sistemas poderia não funcionar corretamente, e recair-se-ia sobre o desafio de programar partes das aplicações direcionadas diretamente para cada um dos sistemas nativamente.



Assim se chegou à segunda iteração de desenvolvimento. Esta consiste em apenas se focar numa aplicação para um dos sistemas móveis (neste caso seria *Android*). Deste modo resolver-se-ia o problema de eventuais falhas na comunicação, ou atrasos, entre equipamentos diferentes. Esta solução tinha a acrescida bonificação da existência de uma comunidade enorme de *developers* para os sistemas *Android* que já estavam familiarizados com o ambiente de desenvolvimento, APIs, eventuais falhas e soluções de problemas que poderiam surgir. O problema desta solução era a inexperiência com o desenvolvimento de aplicações móveis nativas, existiria uma curva de aprendizagem com um inicio lento que se poderia prolongar além dos prazos definidos. Também seria complicado testar soluções alternativas uma vez que estas ocupariam tempo precioso que poderia ser gasto no desenvolvimento da solução final. Apesar de trazer as vantagens de uma base de código única, esta solução também impõe uma exclusão relativamente a todos os utilizadores de outras plataformas, tanto *IOS* como outro tipo de dispositivos móveis.

A terceira e atual iteração consiste no desenvolvimento de aplicações que não sejam focadas quer no ambiente nativo dos equipamentos móveis dos condutores e pedestres, quer no próprio *Operating System* (OS) dos dispositivos. Assim decidiu-se desenvolver uma solução que enquadrasse uma *webapp*. Esta abordagem permite que posteriormente seja possível, com um esforço reduzido, transpor a solução desenvolvida para ambientes quase nativos sendo apenas necessário uma *view* para uma página *web*. Deste modo passa a ser possível utilizar a aplicação através de qualquer dispositivo móvel com acesso à Internet, seja este um telemóvel, *tablet* ou computador. Aumentando consequentemente a compatibilidade da aplicação e permitindo a que esta chegue a um maior número de potenciais utilizadores. A figura 16 pretende demonstrar um exemplo de compatibilidade transversal proveniente de soluções com *webapps*, onde a mesma página é acessível em equipamentos distintos.

Figura 16.: Esquema de uma *webapp* representante da solução adotada

Além de apresentar uma maior flexibilidade e uma maior compatibilidade, esta solução permite desenvolvimento em ambiente *web* utilizando tecnologias como *HTML*, *CSS* e *JavaScript*. Uma vez que já existe algum conhecimento relativo a estas linguagens, torna-se também mais fácil a existência de progresso significativo na fase inicial de desenvolvimento, bem como eventuais testes para testar a eficiência de cenários ou soluções distintas.

Concluindo, houve portanto uma mudança no desenho e conceção da solução. Inicialmente pensava-se que uma solução focada nos sistemas móveis como telemóveis seria a melhor abordagem a tomar tal como havia sido proposto em (Bagheri et al., 2014). Após uma fase inicial de desenvolvimento e com a perceção dos desafios que esta opção representaria para a presente solução, esta foi ligeiramente modificada para uma solução focada apenas num OS. Finalmente adotou-se uma solução mais genérica baseada em *webapps* capaz de correr sobre qualquer tipo de dispositivo móvel. A principal desvantagem nesta abordagem poderia estar relacionada com o facto de que esta necessita obrigatoriamente de um acesso à rede para conseguir obter o serviço. Neste caso este desafio não seria um problema uma vez que será mesmo necessária uma ligação à rede para se conseguir obter os dados dos outros utilizadores. Assim assume-se esta solução como a mais adequada.

3.4 ARQUITETURA

Para que se entenda mais facilmente a abordagem de arquitetura selecionada, será dada uma visão geral sobre os módulos utilizados em cada um dos pontos de processamento (clientes finais e servidor) bem como do sistema de dados.

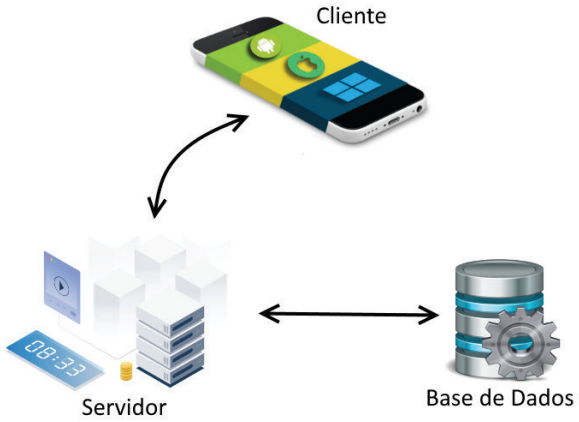


Figura 17.: Elementos essenciais na arquitetura do sistema

A figura 17 representa os três principais elementos que serão tidos em consideração de seguida. Note-se que o cliente nunca comunica diretamente com o sistema de dados, sendo que esta tarefa é feita apenas pelo servidor. Note-se, ainda, que o servidor é o intermediário entre clientes distintos, pelo que o dispositivo representante do cliente não representa exclusivamente um utilizador mas sim todos os utilizadores da plataforma.

3.4.1 *Cliente*

Em relação ao cliente, começa-se por analisar a arquitetura do cliente disponível na figura apresentada em baixo. Agora que se decidiu que a topologia da aplicação segue um modelo cliente / servidor, podemos analisar que componentes foram considerados necessários e como é que a sua utilização foi feita.

A figura 18 apresenta os módulos essenciais para o correto funcionamento da solução proposta, contudo podem ser adicionados módulos extra, ou pode-se omitir módulos, consoante a intenção do programador.

O primeiro módulo que será tratado diz respeito à interface do utilizador, lê-se *frontend* na imagem, ou *User Interface* (UI). Este tem de garantir que o utilizador consegue aceder à plataforma, e entender facilmente a informação que esta disponibiliza. A interface do utilizador serve de intermediária entre a informação proveniente de outros utilizadores (por via do servidor) e o cliente final. Neste caso, o essencial é que de alguma forma

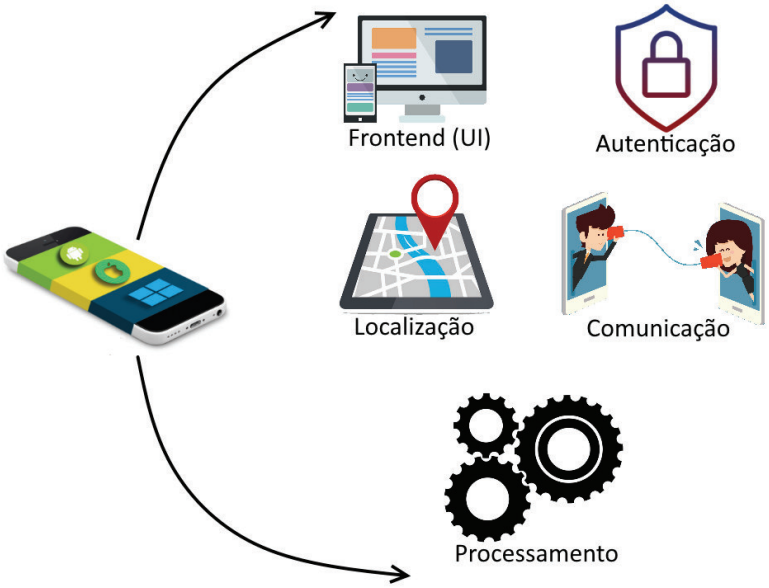


Figura 18.: Módulos existentes na arquitetura do cliente

seja possível avisar os utilizadores de que um potencial cenário de perigo foi detetado. Este módulo deverá ser obrigatoriamente implementado sempre que se pretenda utilizar o sistema para apresentar dados a utilizadores. Se apenas for pretendido algum tipo de utilização simplificada de uma API do servidor, este módulo poderá ser omitido. Exemplos da interface desenvolvida poderão ser observados no anexo A.

Passemos agora ao módulo de autenticação. Este módulo será essencial para assegurar que apenas utilizadores com permissão do sistema conseguem ter acesso aos dados transmitidos. Poderá ser relevante implementar um sistema de controlo de acesso por *login*. No caso em mãos, visto que será enviada informação relativa à posição GPS dos utilizadores, considerou-se que seria boa prática implementar algum tipo de mecanismo de segurança adicional. Este será um dos módulos que poderá ser eventualmente omitido. Existirão certamente cenários onde os dados tratados são públicos, e portanto poderá não fazer sentido despender recursos em assegurar que estes apenas sejam acessíveis por um determinado número de utilizadores. Neste caso, a ideia é garantir o anonimato dos utilizadores da plataforma, mesmo que as suas posições sejam conhecidas.

O módulo que diz respeito à localização deverá de alguma forma conseguir obter o máximo de informação possível relativamente ao utilizador que utiliza a plataforma. Claro que a forma como a deteção de colisões é efetuada afeta diretamente a necessidade por certos valores disponibilizados pela API utilizada. Este módulo não poderá ser omitido para o caso em estudo. É imperativa a sua presença no sistema uma vez que a posição obtida pelo módulo será não só utilizada para transmitir aos restantes utilizadores da plataforma, como também para detetar potenciais cenários de risco.

O módulo relativo à comunicação deverá assegurar que a informação recolhida pelo módulo supracito (localização) será transmitida para o servidor. Este módulo deverá ainda garantir que sempre que um outro utilizador emite uma nova mensagem, esta chegará a si para ser processada e eventualmente apresentar o tal aviso de colisão. A principal função deste módulo é então possibilitar um mecanismo de comunicação direta entre os clientes e o servidor. Note-se que de alguma forma este módulo necessita de estar ativamente em comunicação com o servidor, seja para comunicar um novo valor de posição registado ou obter atualizações de posições de outros clientes. Este módulo também não poderá ser omitido sobre o risco de se perder a essência da aplicação, isto é, sem comunicação não se obtém a localização de outros utilizadores e portanto a plataforma deixa de fazer sentido.

O módulo final que se terá em consideração diz respeito ao processamento. O módulo de processamento garante que a informação obtida pelos módulos de localização e de comunicação serão devidamente tratadas, detetando eventuais situações de risco. Este módulo comunica diretamente com o *frontend* para alertar o utilizador da deteção de um cenário perigoso. Apesar de poder ser retirado da plataforma, a sua ausência modificaria o propósito desta. Uma vez retirado, não seria possível executar o algoritmo de previsão de colisão, pelo que o utilizador apenas observaria as posições de outros utilizadores próximos de si.

Em suma, entende-se que para o correto funcionamento do sistema como um todo, é necessário que o cliente obtenha mecanismos capazes de: apresentar a informação recolhida de forma intuitiva ao utilizador final; autenticar os dados recolhidos caso estes sejam sensíveis; obter indicações relativas à posição onde o próprio utilizador se encontra; comunicar a sua atualização de posição, juntamente com a leitura de novas posições provenientes do servidor; processar os dados obtidos pelos outros módulos, processamento este que deverá ser capaz de detetar cenários de risco.

3.4.2 *Servidor*

O segundo componente essencial para o correto funcionamento da plataforma é o servidor. Este será responsável por filtrar parcialmente parte das mensagens trocadas (capítulo seguinte) e validar dados recolhidos / transmitidos. Uma vez que a comunicação com os clientes e a comunicação com o sistema de dados é feita de forma independente e com implementações distintas, neste caso a arquitetura reflete esta decisão. A figura 19 apresenta um esquema dos módulos essenciais para o funcionamento do servidor.

O módulo de comunicação com a base de dados foi separado do módulo de comunicação com o cliente dadas as diferenças de implementação. Este primeiro módulo serve para obter dados do sistema de dados utilizado, e assegurar que é possível guardar ou atualizar informação já existente. A comunicação com o sistema de dados não poderá falhar, pois é

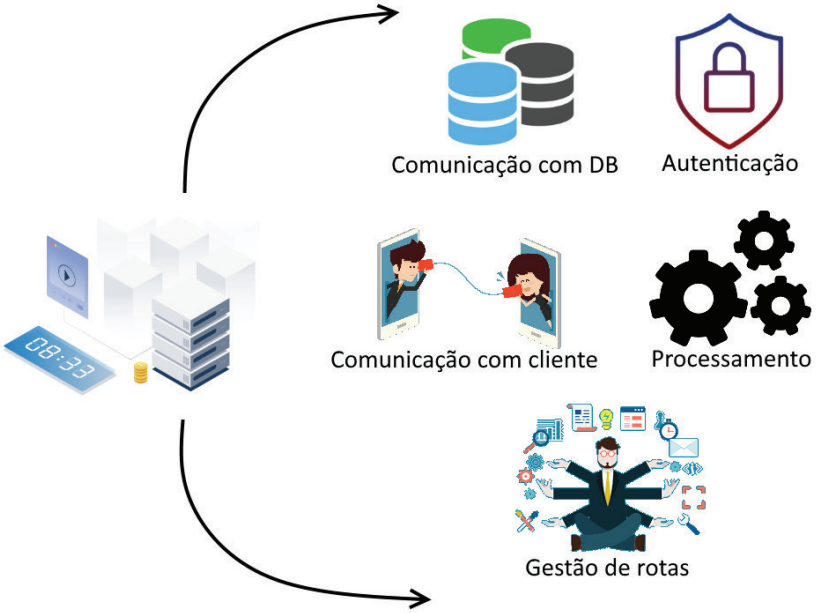


Figura 19.: Módulos existentes na arquitetura do servidor

essencial para o funcionamento da plataforma, a autenticação depende diretamente deste módulo. Caso se opte por implementar objetos armazenados sem estado / voláteis e não em memória, este módulo seria omitido, mas dado que se pretende ter uma grande escala de utilizadores (e que estes comunicam com alguma frequência) considera-se boa prática a utilização de um sistema de dados que atue como suporte. Por este motivo é que foi desenvolvido este módulo, para que na eventualidade de ocorrer uma falha nalgum aspeto de comunicação com cliente, a comunicação com o sistema de dados não seja afetada.

O módulo relativo à autenticação é semelhante ao que se encontra presente na arquitetura do cliente, com a diferença de que, desta vez, se valida os dados obtidos por parte dos clientes. Tal como qualquer outro módulo de autenticação, este deverá permitir que os seus utilizadores se registem, iniciem sessão e terminem sessão. Opcionalmente podem-se utilizar *cookies* para garantir que a sessão do utilizador se mantém ativa por um certo período de tempo estipulado. Uma vez mais, este módulo poderá ser totalmente omitido se não for relevante a existência de mecanismos que impeçam o acesso aos dados transmitidos entre clientes e o servidor.

O módulo de comunicação com o cliente é responsável por receber a informação de todos os clientes existentes na plataforma. Este deverá estar sempre disponível para receber nova comunicação bem como emitir atualizações de posição para outros utilizadores, isto é, se um utilizador comunicar a sua nova posição ao servidor, este deverá ser capaz de reencaminhar essa mesma atualização para os restantes clientes do sistema. Tal como acontece no cliente este módulo não pode ser omitido sob o risco de se perder a essência da aplicação.

É lógico que é necessário ter algum tipo de mecanismo de comunicação com os clientes da plataforma!

O módulo de processamento deverá ser responsável por filtrar de certo modo a informação obtida dos clientes. Visto que o fluxo de mensagens poderá ser bastante elevado, poderá ser interessante adicionar mecanismos que filtrem mensagens para clientes de acordo com algum tipo de lógica, por exemplo a utilizada neste caso foi a de proximidade com o cliente. Este módulo será responsável ainda por adicionar algum tipo de informação que o servidor considere importante relativamente aos dados. Este módulo poderá ser emitido visto que o cálculo dos cenários de colisão acontece no cliente final, contudo note-se que ao omitir este módulo apenas nos resta a hipótese de *broadcast* de informação, o que não será escalável. Além de filtro também poderá ser utilizado para efetuar algumas verificações simples sobre os dados, note-se ainda que este não deverá ter mecanismos complexos pois pretende-se fazer com que as mensagens chegam aos clientes finais o mais rapidamente possível após a sua emissão.

O último módulo que está presente no servidor diz respeito à gestão de rotas URL. Estas asseguram o mapeamento de *endpoints* URL para métodos específicos no servidor. Poderá ser interessante explorar a utilização de rotas distintas para diferentes funcionalidades da plataforma. No caso atual utilizaram-se subdomínios distintos para as várias funcionalidades, e utilizou-se um outro subdomínio que recolhia informações provenientes dos outros. Esta abordagem poderá facilitar a deteção problemas no desenvolvimento da plataforma. Não é um módulo obrigatório, no entanto ajuda a entender o funcionamento do sistema. Por norma todos os servidores mantém algum tipo de gestão de rotas.

Tem-se assim definidos os requisitos essenciais para o funcionamento do componente servidor. Em suma são necessários mecanismos para assegurar: comunicação com o sistema de dados utilizado; comunicação com os clientes; autenticação dos clientes caso os dados trocados entre estes e o servidor forem de cariz sensível; processamento simples dos dados de modo a não introduzir grandes atrasos na entrega de mensagens; uma gestão eficiente de rotas disponibilizadas pelo servidor.

3.4.3 *Sistema de dados*

Tal como referido anteriormente, o sistema de dados não é um requisito obrigatório para o funcionamento da plataforma, no entanto ajuda bastante na gestão de informação recolhida. Note-se que a alternativa à utilização deste tipo de sistemas poderá ser armazenar os dados em estado interno do servidor. Esta abordagem tem vantagens relativamente à velocidade do acesso à informação, no entanto quando se trata de um grande volume de dados é boa prática optar por algum mecanismo com capacidade extra de organização de informação.

Para o sistema em si, apenas é necessário manter registo de utilizadores da plataforma para efeitos de autenticação. É ainda interessante adicionar algum tipo de mecanismo que permita guardar os avisos lançados pelo sistema, visto que estes podem ajudar a detetar eventuais falhas de deteção de potenciais cenários de colisão.

Uma vez que as mensagens obtidas pelos utilizadores são atualizadas com bastante frequência, e que apenas as precisamos de tratar uma única vez, não fará sentido gastar espaço desnecessário para manter registo de todas as posições. Não obstante, isto poderá ser feito caso se pretenda ter efetivamente todo o registo de atividade que passou pelo sistema.

3.5 DESAFIOS ADICIONAIS

Tal como já foi referido anteriormente, juntamente com a tomada de decisão da arquitetura atual, existiram diversos desafios que influenciaram esta escolha. A arquitetura necessária para que todo este sistema funcione corretamente depende de vários fatores importantes. Pode-se considerar como fator principal a necessidade que existe para que as mensagens sejam trocadas rapidamente entre equipamentos distintos. Esta necessidade advém do facto de que atrasos na ordem dos segundos no processamento de mensagens pode fazer com que uma mensagem deixe de ter importância em certos cenários. Este facto é facilmente observável se se considerar o caso em que um veículo se desloca a alta velocidade (ex.: 72km/h), onde um atraso de entrega de apenas um segundo pode-se refletir numa diferença de posição de cerca de 20 metros. Assim surgiu o primeiro desafio, garantir que a troca de mensagens é rápida o suficiente para que estas ainda tenham algum tipo de utilidade.

Outro desafio adicional que surgiu esteve relacionado com a forma como as mensagens eram difundidas. Inicialmente optou-se por uma metodologia de difusão que passava por fazer simplesmente um *broadcast* para todos os outros elementos da rede, o que rapidamente se tornaria impraticável.

A arquitetura utilizada, juntamente com algumas bibliotecas e mecanismos dos quais se falarão no capítulo seguinte, ajudaram a minimizar estes problemas.

Relativamente à arquitetura, houve uma dificuldade inicial a garantir que o sistema se mantinha acessível a partir da rede. Existe o problema de que a rede onde o servidor se encontra hospedado mude de *IP*, tornando-o inacessível remotamente. Uma solução encontrada para isto foi a utilização de um *bot* que automaticamente verifica se a página se encontra disponível. Utilizou-se mais concretamente o *UptimeRobot*, que pode ser encontrado em anexo, na figura 70. Esta ferramenta verifica a cada 20 minutos se a plataforma se encontra disponível, caso contrário é enviado um email ao gestor. Sempre que é detetada uma falha, pode-se pensar em três causas imediatas: o tempo de latência pode ter sido

demasiado alto, tendo originado um falso positivo (que acontece na maioria dos casos); ocorreu um erro inesperado no servidor em que este bloqueou as respostas a clientes (ou simplesmente utilizou um mecanismo de prevenção de *spam* e baniu o *bot*); ou o *IP* da rede mudou.

O primeiro caso é geralmente fácil de resolver, apenas basta esperar. A melhor solução seria aumentar a largura de banda disponibilizada, mas visto que neste caso a largura de banda ou a velocidade de ligação estão relacionadas com os serviços disponibilizados por um *Internet Service Provider* (ISP), esta tarefa torna-se impossível (sem custos adicionais).

A segunda causa pode ser resolvida temporariamente reiniciando o servidor. Isto apenas nos possibilita algum tempo adicional com o sistema em funcionamento, uma vez que nada garante que este não possa falhar novamente. Há que ver os registos do servidor e apurar se existiu algum erro para que a resposta falhe, ou se simplesmente foi um caso pontual. O motivo mais comum para falha de resposta poderá ser o *spam* de pedidos por parte de utilizadores mal intencionados. Neste caso o servidor entra em modo alerta e deixará de responder ao *IP* do utilizador que provocou o ataque. Note-se que muitas vezes o *IP* detetado não é de um cliente em si mas sim da rede onde este se encontra, sendo que portanto é possível que vários clientes sejam afetados pela contra-ação do servidor.

A terceira hipótese estipula que o *IP* de rede sofreu alterações. Infelizmente este seria o cenário mais difícil de resolver remotamente, uma vez que o *IP* muda, deixa de ser possível aceder remotamente ao servidor para fazer qualquer tipo de manutenção. Felizmente o ISP utilizado na rede do servidor disponibiliza uma aplicação que permite ver o endereço *IP* de rede contratado. Assim, juntamente com a *dashboard* de gestão de endereçamento (anexos na figura 71), é possível retificar prontamente o erro/atualização de endereços. Feita a alteração, é necessário aguardar uns minutos ara que estas surtam efeito, tudo deverá estar operacional novamente.

O último desafio a nível de arquitetura como um todo foi a forma como se optou por repartir o sistema. A ideia foi ser o mais especifico possível, utilizando subdomínios diferentes para cada uma das funcionalidades. Aqui houve problemas em gerar certificados e fazer com que estes fossem validados pela plataforma utilizada (em anexo na figura 72). O *SSL4Free* tem por base o *letsencrypt*, uma ferramenta muito conhecida para gerar certificados temporários SSL para *websites*. O maior problema aqui foi gerir juntamente com a arquitetura de rotas que estava disponível no servidor naquela altura. Foi necessário alterar toda a implementação a nível de código para que os domínios novos pudessem ser acessíveis a partir da rede.

Uma vez que se está a lidar com dados importantes do utilizador é essencial garantir a sua segurança, por isso é que se optou por ter esta acrescida dificuldade de implementação, para garantir que os dados se mantém tão seguros quanto possível.

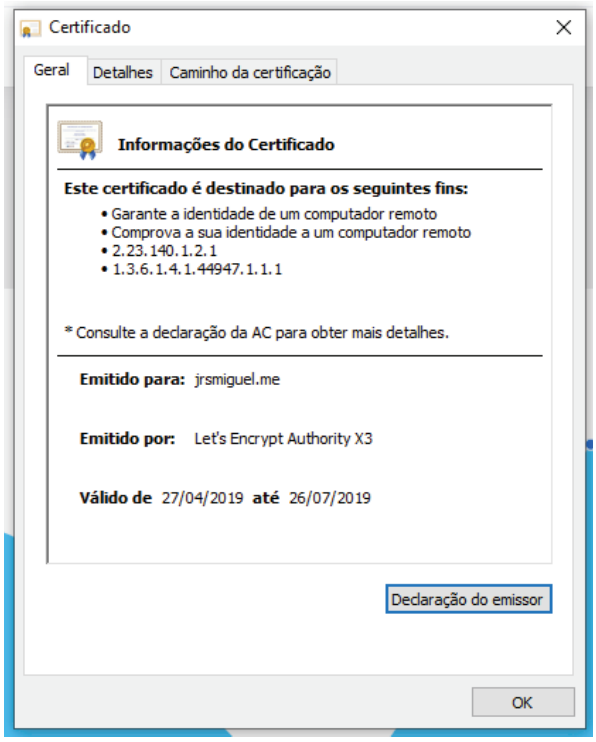


Figura 20.: Certificado SSL gerado para todo o sistema

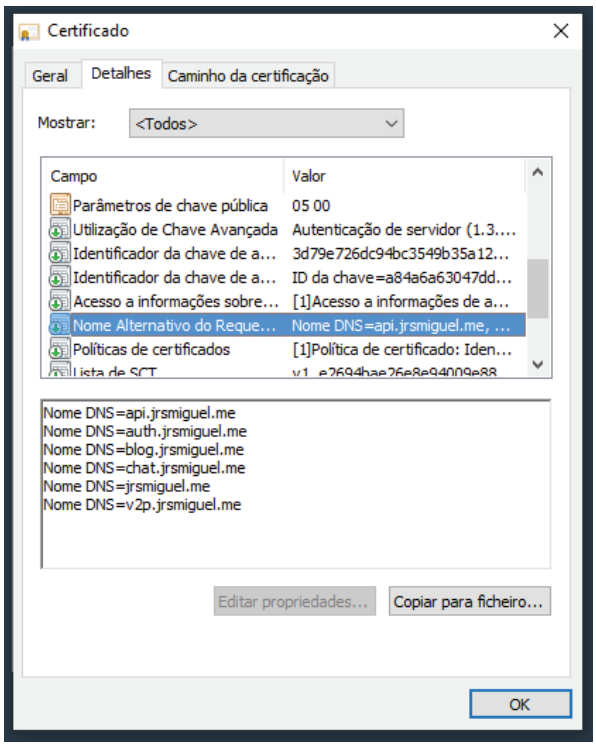


Figura 21.: Este certificado SSL permite subdomínios distintos específicos à plataforma

4

DESENVOLVIMENTO

Neste capítulo de desenvolvimento, descrevem-se detalhadamente as escolhas de implementação que foram feitas bem como os desafios que se colocaram durante a implementação da arquitetura de solução especificada no capítulo anterior.

Inicialmente faz-se uma apresentação das ferramentas utilizadas para o desenvolvimento do sistema. De seguida justificam-se as tomadas de decisão, apresentando as hipóteses alternativas às escolhas efetuadas (no contexto de implementação). Por fim descreve-se a implementação e a forma como esta foi conseguida. O capítulo termina com uma mostra de resultados obtidos através do algoritmo utilizado para trocas de mensagens e concluí-se com um resumo relativo ao trabalho realizado no desenvolvimento da solução.

4.1 FERRAMENTAS DE IMPLEMENTAÇÃO

Figura 22.: Logótipo da base de dados utilizada (PostgreSQL)

Para o sistema de base de dados utilizou-se *PostgreSQL*. Esta base de dados relacional é totalmente *open source*, o que significa que poderá ser utilizada abertamente por qualquer utilizador. Além disto o *PostgreSQL* já se encontra em desenvolvimento constante ao longo de mais de 30 anos, pelo que ganhou uma forte reputação no que toca à sua performance, robustez e confiabilidade. Os 30 anos em desenvolvimento permitem que a comunidade que revolve em torno desta solução já tenha tido muitas experiências de resolução de problemas, pelo que facilita todo este processo. A importância inicial e a necessidade para

a implementação de um sistema de base de dados deve-se à utilização de um sistema de *login* para que se consiga fazer um acesso autenticado à plataforma. Esta será utilizada para guardar informações de sessão e posteriormente, informações relativas a eventuais cenários de risco para que se consigam apresentar estatísticas aos utilizadores.

Figura 23.: Logótipo da *framework* utilizada (NodeJS)

Para que toda a plataforma funcione corretamente foi desenvolvido um servidor em *NodeJS*. O *NodeJS* permite uma arquitetura assíncrona, orientada a eventos e foi especialmente desenvolvido tendo em mente aplicações de rede escaláveis. Regra geral o *NodeJS* apenas é executado quando um evento é despoletado, sendo que serão necessários poucos recursos para a manutenção e constante execução de um serviço. Esta *framework* possibilita a execução de código *JavaScript* fora do contexto do *browser*. A versão inicial foi lançada à cerca de 10 anos sendo atualizada com bastante frequência. Esta *framework* é cada vez mais utilizada nos dias que correm sendo que a comunidade que rodeia este ambiente também se encontra em expansão constante. A flexibilidade do *NodeJS* permite que o servidor corra código semelhante ao cliente final, fazendo com que não seja necessário utilizar tecnologias distintas entre os clientes e o servidor em si. Outra grande vantagem desta solução é a existência de um vasto leque de APIs que facilitam a implementação de todo o tipo de funcionalidades. Uma das utilizações para o caso em estudo poderá ser, por exemplo, a maior facilidade de ligação com a base de dados em *PostgreSQL*, ou a fácil implementação de um sistema de migração de base de dados. Existem imensas alternativas de implementação para uma mesma funcionalidade, o que por vezes pode ser contraproducente, e até impedir que se chegue a uma solução apropriada às nossas necessidades.

A flexibilidade do *NodeJS* permite a utilização de motores de *templating*. Estes facilitam o desenvolvimento da parte *frontend* de uma *WebApp*, neste caso foi utilizado *Embedded JavaScript templating* (EJS). Este motor permite a execução de código em *JavaScript* por parte do servidor para embutir código ou texto num *template HTML* antes que este seja enviado para o utilizador. Esta funcionalidade permite enviar páginas *web* distintas para cada cliente. A

Figura 24.: Logótipo do motor de *templating* utilizado (EJS)

utilização pode ser feita para customizar uma *WebApp* ou, por exemplo, para mostrar dados exclusivos a cada utilizador sem que seja necessário qualquer outro tipo de API para tal.

Figura 25.: Logótipo do gestor de processos utilizado (PM2)

O passo final para a implementação deste sistema foi um mecanismo que permitisse que a plataforma estivesse sempre em constante execução. Para tal utilizou-se um gestor de processos, neste caso o *Process Manager 2* (PM2) completamente compatível com servidores em *NodeJS*. O PM2 é um gestor de processos em produção para *NodeJS* que contém mecanismos de balanceamento de carga na essência da sua arquitetura. Com a utilização desta ferramenta é possível definir que uma *WebApp* se manterá ativa indefinidamente, mesmo que aconteça um cenário de erro no código, o que caso contrário causaria a paragem de execução do serviço. Com esta ferramenta ainda é possível criar múltiplas instâncias do mesmo servidor, que serão invocadas de acordo com a carga de cada uma delas, de modo a que a resposta ao cliente seja o mais eficiente possível.

Figura 26.: Logótipo do serviço que mantém o repositório com o código desenvolvido (GitHub)

Para armazenamento remoto do código e controlo de versões, prevenindo eventuais perdas do trabalho feito, utilizou-se o *GitHub*. Para além desta ferramenta permitir guardar o código em *cloud*, também possibilita um controlo de versões. É possível desenvolver funcionalidades diferentes sobre um código original idêntico criando um novo ramo / *branch*, seguindo-se de uma consequente integração de ambas versões num código conjunto híbrido. Mais que isto esta ferramenta permite a gestão de tempo despendido em cada fase do projeto. É possível definir *triggers* que movem cartas de categorização automaticamente entre as etiquetas de tarefas em desenvolvimento, tarefas em teste e tarefas concluídas. Sempre que se pretende criar uma nova funcionalidade gera-se um novo *issue*, este pode ser associado a um *branch* relativo a uma funcionalidade, uma correção de *bugs*, melhoramento, entre outros. Esta arquitetura interligada com ferramentas de integração contínua têm uma importância enorme para uma boa gestão do desenvolvimento do projeto como um todo.

Figura 27.: Logótipo do gestor de integração contínua (CircleCI)

Para garantir que qualquer atualização do código desenvolvido não afete o correto funcionamento da aplicação, foi necessário desenvolver mecanismos que assegurassem a execução de testes unitários de forma automática. Para isto utilizou-se o *CircleCI*, uma *framework* de integração contínua que possibilita executar testes unitários sempre que uma alteração seja efetuada. Esta funcionalidade garante que uma alteração num certo ponto do nosso código-fonte não causa falhas noutro lugar da *WebApp* tornando assim a implementação de soluções para uma fase de produção, mais robusta e segura. Esta ferramenta interage diretamente com o *GitHub*, assim que se efetua um *push* de alterações para o repositório onde o código se encontra, o *CircleCI* corre automaticamente *containers* em *Docker* que montam imagens de uma base de dados *PostgreSQL*, de um servidor em *NodeJS*. Quando as imagens estiverem prontas, começa-se um ciclo de testes sobre o código para garantir que

o resultado de certos métodos implementados em código se mantém constantes a nível de formato e informação contida.

4.2 DECISÕES

Nesta secção serão analisadas as decisões efetuadas durante a implementação deste projeto.

A primeira decisão reflete a escolha efetuada relativamente à comunicação entre o cliente e o servidor. A implementação inicial passava por enviar diretamente as informações de atualização de posições GPS dos clientes por métodos *POST HTTP*. Para recolher informações atualizadas do servidor, cada cliente precisaria de realizar um *probing* para verificar as novas posições do outros clientes do sistema. Esta abordagem torna-se ineficiente para um grande número de utilizadores podendo ocorrer o caso em que o servidor seja sobrecarregado constantemente por pedidos, tendo efeitos semelhantes aos de um ataque *Denial-of-service* (DDoS). O funcionamento de um pedido *HTTP* requer sempre iniciativa por parte do cliente, como é esquematizado na figura 28.

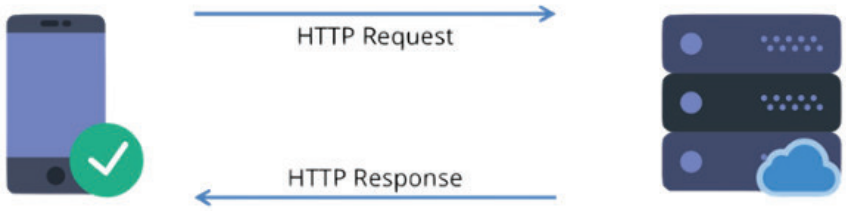


Figura 28.: Exemplo de pedido HTTP a um servidor

A solução adotada para este desafio passou pela utilização de *websockets* como canais de comunicação direta entre o cliente e o servidor. Esta decisão fez com que não fosse necessário recorrer a *probing* constante de atualização, sendo que o servidor tomaria iniciativa e teria a opção de comunicar diretamente com os seus clientes. Esta decisão passou por implementar uma arquitetura semelhante à conhecida como *publish - subscribe*. Este tipo de arquitetura permite que um agente do sistema subscreva a um tópico de comunicação, recebendo toda a informação sobre este. No presente caso considerou-se que o servidor se encontra subscrito a todas as posições de localização e que cada cliente subscreve apenas à sua área. Isto permite efetuar uma filtragem inicial das respostas recebidas pelos clientes fazendo com que o sistema se torne mais escalável.

Outra grande decisão relativa à implementação do sistema relaciona-se com a capacidade de resposta múltipla a pedidos dos clientes. Inicialmente tinha sido utilizado no servidor, um mecanismo de respostas sequenciais a pedidos dos clientes. Um exemplo pode ser observado na figura 29.

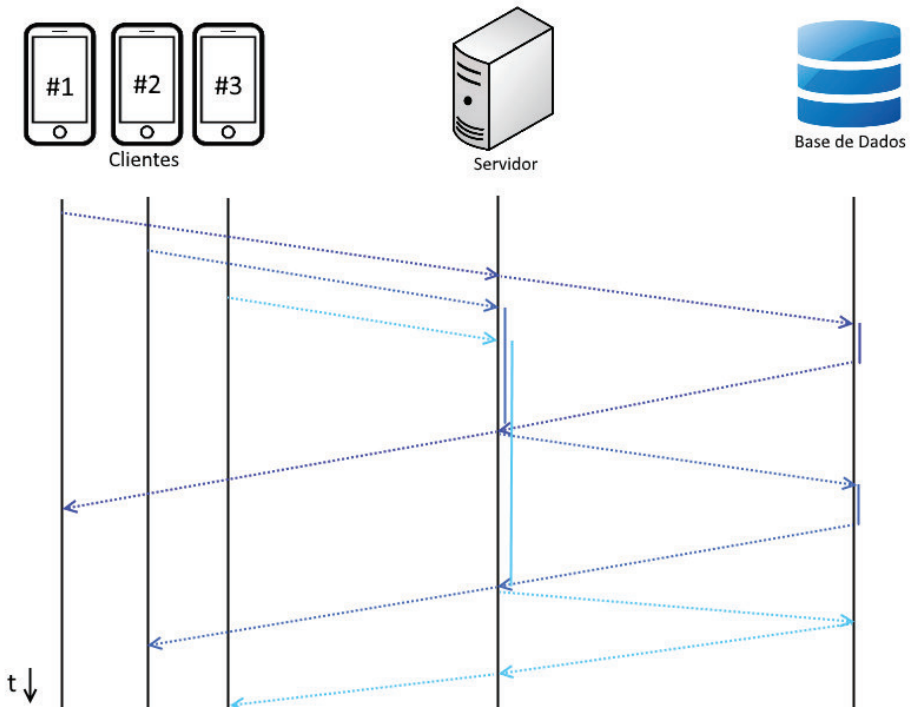


Figura 29.: Exemplo de pedido síncrono ao servidor

Isto implica que enquanto um cliente não obtém a resposta pretendida, nenhum outro cliente conseguirá ter acesso a este recurso. Esta opção de implementação agrava-se quando é necessário aceder a informação existente no sistema de dados, podendo aumentar exponencialmente os tempos de resposta aos pedidos de todos os clientes. Tendo esta ideia como ponto de partida alternativo, e tentando resolver os problemas que esta impõe, optou-se por seguir uma implementação de rotas URL assíncronas. Visto que todo o desenvolvimento do servidor foi efetuado utilizando *JavaScript*, decidiu-se utilizar *promises*. Estas são utilizadas para lidar com operações assíncronas, e servem como alternativa à implementação da metodologia orientada a eventos que existia anteriormente. A sua utilização possibilita uma melhor organização do código desenvolvido bem como um melhor controlo sobre erros gerados. A utilização deste tipo de resposta assíncrona permite que o servidor aumente a sua produtividade no sentido em que este poderá responder sempre a clientes sem que estes necessitem de esperar por quantidades absurdas de tempo. Sempre que o servidor recebe um pedido de um cliente, gera uma *promise* e prossegue a responder a novos clientes. A própria *promise* só continuará a execução quando a *query* ao sistema de dados terminar (quero seja com sucesso ou com erro). Ao utilizar esta metodologia aumenta-se o *throughput* do servidor, pois este nunca deixará um novo cliente à espera por longos períodos de tempo enquanto não obtiver a resposta para um outro cliente anterior, como é exemplificado na figura 30.

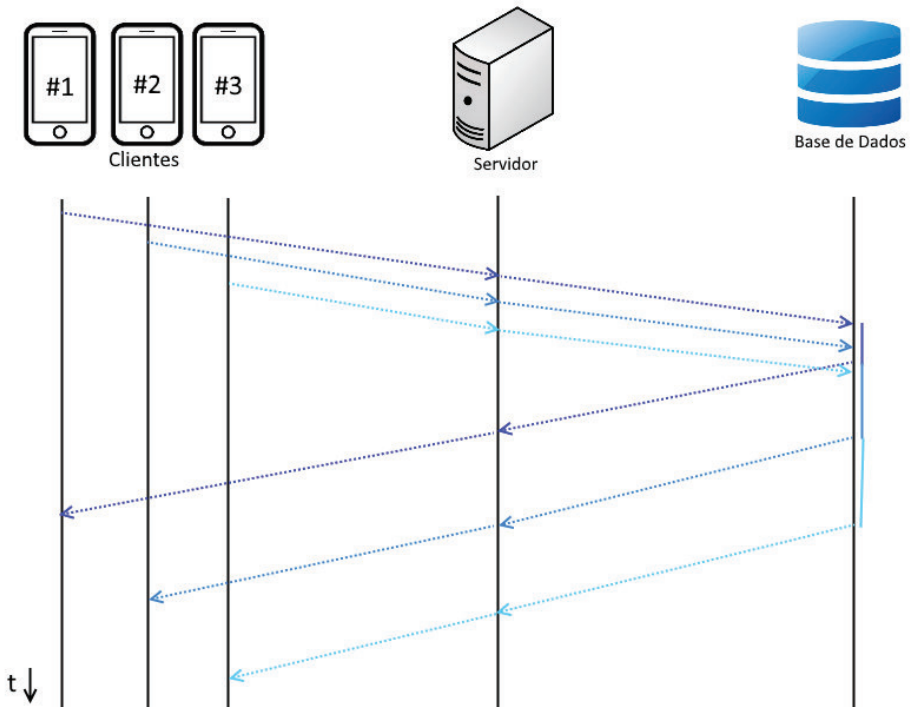


Figura 30.: Exemplo de pedido assíncrono ao servidor

A terceira decisão tomada durante o desenvolvimento do sistema é a primeira que lida e impacta diretamente o funcionamento do algoritmo de deteção de cenários de perigo. A decisão efetuada estava relacionada com a precisão das medições de posição GPS. Existiam duas possibilidades: diminuir a precisão e os tempos de atualização da posição, aumentando a duração da bateria do dispositivo (diminuição do consumo energético); ter uma maior precisão relativamente aos valores medidos e atualizá-los com uma maior frequência (o que diminui significativamente a duração da bateria). Dado que a arquitetura cliente / servidor introduz latência na comunicação entre utilizadores, considerou-se mais vantajoso utilizar medições o mais precisas e o mais frequentes possíveis. A API utilizada dava a possibilidade de implementar ambas as formas que haviam sido ponderadas. Existia, por exemplo, a possibilidade de observar periodicamente a posição do utilizador. Esta posição seria dada pela rede onde o cliente se encontra e posteriormente seria transposta para coordenadas GPS. Em casos de interior de edifícios ou zonas cobertas, a localização era equivalente à dos *Access Point* (AP) *wireless*, podendo apresentar algumas dezenas de metros de erro da posição real. Neste caso, visto que a localização apenas seria atualizada se o cliente se conectasse a um AP diferente, existe um erro grande que teria de ser resolvido de alguma forma. A outra opção, também disponibilizada pela API, permite utilizar a posição através do próprio GPS do dispositivo, com uma margem de erro de no máximo 3 metros (resultados obtidos por teste). Após verificada esta diferença significativa de erros de lei-

tura, considerou-se melhor optar por uma leitura correta em detrimento da bateria. Além de ser mais precisa, esta solução acaba por emitir atualizações de localização com maior frequência, reduzindo o impacto que a falha de uma mensagem teria. Apesar desta vantagem existem dispositivos onde o GPS pode não estar disponível, neste caso a localização será utilizada de acordo com a posição da rede ou do AP.

O próximo passo de desenvolvimento implica uma decisão relativa ao local onde se localiza o cálculo usando o algoritmo de decisão. Existiam várias possibilidades de como efetuar esta repartição de necessidade de cálculo, entre elas: realizar os cálculos de previsão exclusivamente no cliente; efetuar os cálculos de previsão exclusivamente no servidor; adotar uma ideologia híbrida onde se efetua parte do cálculo no servidor e outra parte no cliente. Como dito anteriormente, o servidor apenas deveria efetuar algumas validações básicas aos dados obtidos de modo a não adicionar uma latência acrescida à transmissão de mensagens. Assim não faria sentido colocar o cálculo necessário para o funcionamento do algoritmo, no lado do servidor. Caso fosse possível aumentar os recursos do servidor faria sentido transpor este cálculo para o servidor, como tal não é possível neste caso (por falta de recursos), optou-se por fazer o cálculo apenas nos clientes. Uma das maiores desvantagens desta abordagem é a necessidade de efetuar o cálculo em todos os clientes, sendo que se calcula o mesmo cenário múltiplas vezes.

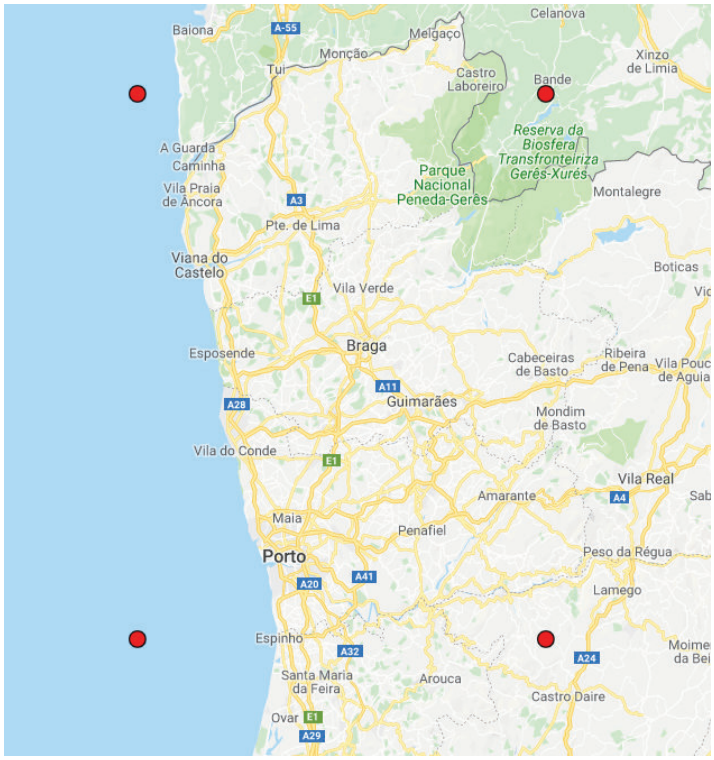


Figura 31.: Grelhas latitude / longitude definidas para efeitos de teste

Para tentar reduzir ao máximo esta duplicação de cálculo, adotou-se uma validação muito simples no servidor que limita as mensagens enviadas por grelhas de latitude e longitude. A ideia aqui é que sempre que um utilizador envie uma nova mensagem, esta seja processada pelo servidor e apenas será enviada para outros utilizadores que se encontrem na mesma área que o cliente que a enviou. Deste modo limita-se significativamente, não só o número de cálculos que cada cliente terá de efetuar, como também o número de vezes em que um cálculo semelhante é efetuado. Por exemplo um utilizador entre as grelhas de latitude de 41º e 42º e as grelhas de longitude de -8º e -9º, visíveis pela figura 31, apenas enviará mensagens (e receberá mensagens) de utilizadores que se encontrem na própria grelha. Ao adotar esta ideia assume-se que os dispositivos finais têm algum tipo de capacidade de processamento para conseguir realizar os cálculos, este fator poderá ser importante no sentido em que pode ser limitativo relativamente ao número de clientes que podem ter acesso ao sistema.

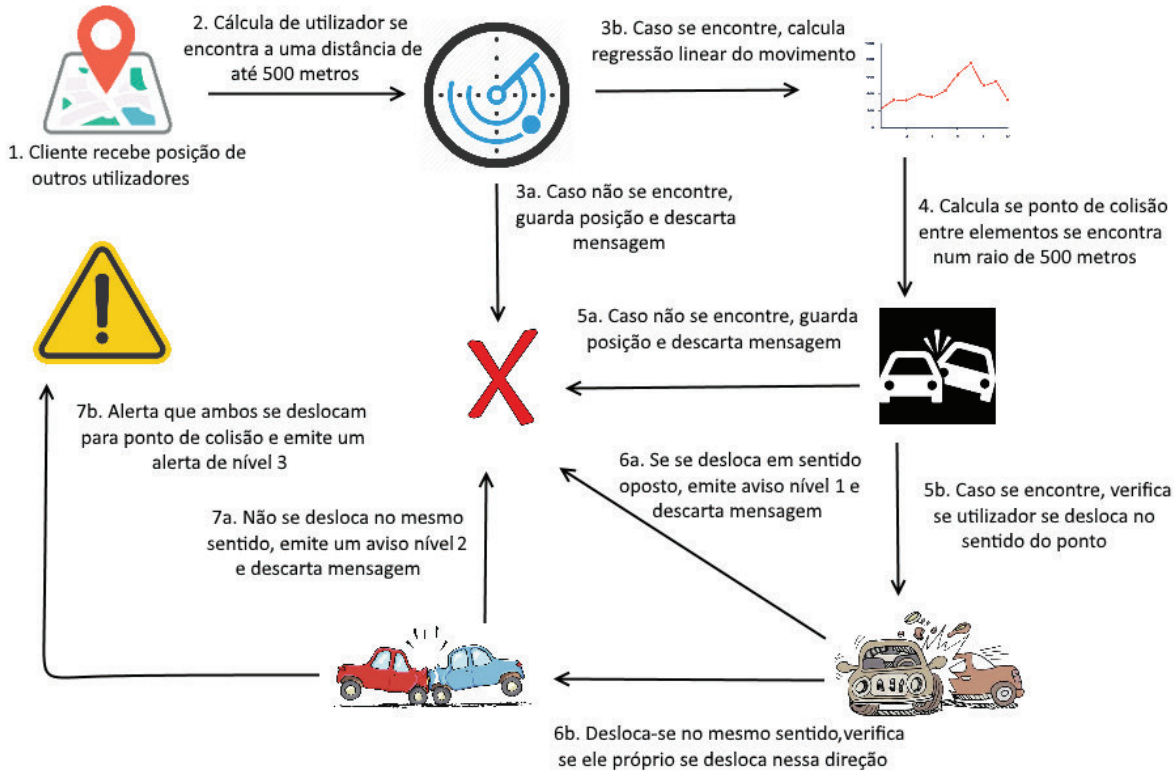


Figura 32.: Diagrama de funcionamento do algoritmo de previsão

A próxima decisão diz respeito ao modo como o algoritmo de decisão foi implementado. Tal como foi visto no segundo capítulo, existem várias formas de conseguir implementar um algoritmo de decisão. Claro que algumas destas estão dependentes da existência de câmaras, sonares, *lidares*, capazes de detetar parâmetros que não são de fácil obtenção

através de telemóveis comuns. A escolha de utilização de dispositivos móveis dos utilizadores como método de obter informação relativa à sua localização, não só limita as informações que são possíveis de obter, como também limita a capacidade de processamento que cada cliente tem à sua disposição. Dada que uma das decisões consistiu em migrar o processamento para o lado do cliente, e visto que cada cliente necessita de calcular possíveis cenários de risco, é fundamental reduzir os cálculos efetuados pelo algoritmo para que este permaneça utilizável. O primeiro passo é efetuado no servidor com a limitação das grelhas de latitude e longitude. De seguida cada cliente verifica que utilizadores se encontram numa distância de até 500 metros até si, mantendo este registo durante até 1 minuto (ou enquanto não existir informação de 5 novas atualizações de posição). Aqui se conclui que na verdade são mantidas as últimas cinco posições de cada utilizador num raio de 500 metros. Sempre que existem mais que três posições registadas, procede-se para o próximo passo, o cálculo da regressão linear de movimento do outro utilizador. Quando se obtém, ou quando é possível obter, esta regressão linear (pelos três a cinco últimos pontos de localização), deteta-se se o ponto de interseção das regressões de outros clientes com a do próprio utilizador se encontra num raio de até, também, 500 metros. Caso exista este ponto de interseção então é necessário verificar se ambos utilizadores se deslocam na sua direção, onde caso afirmativo é enviado um aviso ao utilizador e ao cliente. O modo de funcionamento deste algoritmo pode ser visto na figura 32.

4.3 Implementação

Durante o desenvolvimento da plataforma foram tentadas várias alternativas de implementação de certas funcionalidades. Além disto notou-se que existia a necessidade de incorporar no projeto, certos mecanismos que possibilitem um desenvolvimento mais organizado e estruturado.

O projeto foi implementado *JavaScript* utilizando o motor *NodeJS* para execução em ambientes que não sejam *web browsers*. Escolhida a linguagem resta saber como se dará a sua implementação, o primeiro passo foi escolher uma *framework* capaz de expor um (ou vários) *endpoints* que podem vir a servir o sistema. Estes *endpoints* podem ser relativos à aplicação *web* que se pretende desenvolver ou a possíveis APIs que o serviço possa consumir. Como escolha de *framework*, utilizou-se a muito conhecida *ExpressJS*, esta é exclusiva para servidores desenvolvidos em *NodeJS* e foi escolhida dada a sua simples utilização quando se pretende gerar aplicações *web* semelhantes à proposta. Além disto, a sua utilização possibilita a implementação rápida de eventuais APIs que venham a ser necessárias, tudo o que tinha sido proposto anteriormente.

Após escolha e desenvolvimento inicial da plataforma, foi necessário escolher a forma como a aplicação seria apresentada ao utilizador, ou pelo menos escolher a forma como o

utilizador acedia à plataforma. Aqui decidiu-se utilizar um domínio que existia anteriormente (*https://jrsmiguel.me*), no entanto existia outro desafio, esta página já tinha conteúdo a ser apresentado de forma estática. Desta forma decidiu-se adicionar sub-domínios, onde estes seriam utilizados para dividir a funcionalidade da aplicação. Foram gerados os certificados para os sub-domínios de: api, auth, chat e v2p. Assim foi possível apresentar de forma mais intuitiva cada funcionalidade distinta do sistema. **API** trata de responder a todos os pedidos que sejam efetuados que não necessitem de qualquer interface gráfica. Serve para expor dados do servidor para que estes possam ser acedidos por qualquer cliente. **AUTH** trata de apresentar uma simples página *web* onde disponibilizará o registo de novos utilizadores na plataforma, bem como o seu *login* e *logout*. **CHAT** foi utilizado inicialmente quando se pretendia testar uma nova forma de comunicação, através de *sockets* que permitissem que as informações no lado do cliente fossem atualizadas a partir do servidor. Esta funcionalidade foi implementada como método de aprendizagem para o funcionamento do sistema de *sockets*, contudo permanece ativo para que seja possível comunicação entre utilizadores da plataforma. **V2P** é utilizado para expor o serviço que deverá ser acedido para utilizar o sistema de alerta. É portanto o foco essencial desta dissertação. Todo o desenvolvimento acaba por ser consumido por este sub-domínio.

O primeiro passo de implementação foi verificar que o serviço respondia quando se acedia às páginas expostas aos clientes. Utilizou-se *EJS (Embedded JavaScript Templates)* para desenvolver as páginas, estas permitem embutir informação relativa aos utilizadores no próprio servidor, e enviar estes valores como página estática ao cliente. Após se verificar que a aplicação respondia às rotas que haviam sido definidas, era necessário decidir como seriam trocadas as informações entre o cliente e o servidor. Inicialmente a ideia seria utilizar pedidos *HTTP GET* para obter novos dados do servidor e utilizar pedidos *HTTP POST* para atualizar dados valores no servidor. Esta ideia falha quando se percebe que um utilizador terá de atualizar informação noutro. Como proceder neste caso? Se fosse necessário efetuar um pedido *GET* para obter informação, então era necessário implementar uma espécie de *probbing*? Não seria isto ineficiente? A resposta breve é que sim, seria necessário efetuar *probbing* constante e seria bastante ineficiente. Além disto rapidamente se correria o risco de sobrecarregar todo o sistema dado que todos os clientes teriam de efetuar este dito *probing*. A aplicação não seria escalável. Para evitar este problema optou-se por migrar a implementação que havia sido feita para que esta compreendesse a utilização de *sockets*. Qual é a vantagem dos *sockets*? Estes possibilitam a que o servidor atualize o estado / informação existentes em clientes, e vice-versa. Então com a sua utilização seria possível garantir que um utilizador poderia informar o servidor da sua localização, o servidor trataria dos dados e enviaria para todos os outros clientes a quem esta interessasse. Para testar o funcionamento de tal ferramenta, antes de comprometer totalmente

a primeira implementação (por pedidos *HTTP*), desenvolveu-se a plataforma de *chat*, que posteriormente foi melhorada com a interface apresentada em anexo, figura 54.

Tinha-se então desenvolvido o essencial para o funcionamento da aplicação, com uma plataforma acessível a todos os utilizadores, e ainda uma forma de comunicar entre utilizadores. Resta saber como se obtém a posição / localização de cada utilizador. Tal como foi discutido anteriormente, estudou-se a possibilidade de utilizar dispositivos *bluetooth* conectados às entradas OBD2 dos veículos, para obter informações destes no que toca à sua posição, velocidade, direção, entre outros. Esta abordagem estaria suportada por uma API de *bluetooth* que é nativa aos *web browsers* mais atuais (incluindo os de plataformas móveis). Esta ferramenta possibilitaria a comunicação direta entre o dispositivo a ser ligado à viatura e à plataforma *web* que havia sido desenvolvida. Com um simples clique o utilizador poderia emparelhar o seu dispositivo ao equipamento conectado à entrada OBD2. Mais tarde se percebeu que na verdade o dispositivo dos utilizadores (*smartphone*) consegue captar muito mais informação por si só, e que a velocidade seria o valor mais preciso que se obteria a partir da utilização deste equipamento (OBD2) auxiliar. Note-se que a posição GPS obtida por um *smartphone* (testado com um Nokia6 de 2017) tem uma menor margem de erro que a posição dada por uma viatura (testado com um Mercedes de 2016), sendo que em alguns casos esta nem sequer está disponível (testado com Hyundai e Peugeot ambos de 2013). Este caso originou uma das decisões explicadas anteriormente, onde não se justificaria o gasto energético de ter a interface *bluetooth* conectada. Ainda assim, apresentam-se algumas noções básicas de como esta API funcionaria. O primeiro passo necessário para originar a troca de informações seria o dito emparelhamento com o *browser*, para tal será necessário requisitar autorização para comunicar com um dispositivo, onde o utilizador selecionaria o seu equipamento OBD2. De seguida seria efetuada uma tentativa de conexão com este mesmo equipamento, onde caso esta fosse efetuada com sucesso seriam pedidos a listagem de serviços que o dispositivo OBD2 disponibilizaria. Após detetar os serviços disponíveis, a plataforma subscreve a certas características sendo possível efetuar a leitura / recolha de valores desejados (neste caso obter-se-ia a velocidade, por exemplo). Parte essencial do código utilizado para o funcionamento desta funcionalidade pode ser encontrado no anexo C.2 (64).

Após se verificar que a utilização dos equipamentos OBD2 seria descartada, bastava verificar como se acederia diretamente pelo *browser* às informações de GPS disponibilizadas pelos equipamentos dos utilizadores. Para isto os *browsers* mais atuais permitem uma interface também bastante intuitiva de geolocalização, parte desta pode ser observada no anexo C.4 (66). Um aspeto essencial a referir aqui é a utilização da *flag enableHighAccuracy*. Esta garante que a posição que será obtida a partir dos dispositivos dos utilizadores será obtida através do GPS do equipamento, tendo um erro máximo de posição de cerca de 3 metros. Caso esta não esteja presente, o seu valor por defeito será o oposto ao que se pretende, pelo

que o equipamento decidirá se utiliza ou não o GPS para efetuar a leitura da posição. Neste caso, na eventualidade do utilizador ter definido no seu equipamento que a localização deverá ser obtida por GPS, tudo ocorrerá como esperado, no entanto, também por defeito, a localização é extrapolada por informação de rede, onde os erros, embora pouca frequência, podem ser na ordem dos quilómetros. Isto deve-se porque na verdade a localização que se obtém é relativa ao AP ao qual o equipamento se conectou e não à sua localização real. Foi efetuado um teste onde um utilizador ligado a uma rede doméstica em Braga, obtinha posições de latitude e longitude do Porto. Ora se estivermos na presença de uma ligação por LTE nada garante que esta posição não tenha um erro semelhante, e dado que este AP pode encontrar-se a grandes distâncias, o erro na leitura da localização é superior. Note-se então que esta *flag* não deverá a qualquer custo ser omitida, sob o risco de alterar erraticamente os valores obtidos pelos clientes de acordo com o algoritmo de decisão.

Ao longo do desenvolvimento do projeto notou-se que este estaria a tomar grandes proporções pelo que caso fosse necessário modificar algum método ou valor faria sentido começar a desenvolver algum tipo de documentação. Para este efeito utilizou-se uma ferramenta denominada de *jsDoc* que permite comentar diretamente funções ou variáveis em código e posteriormente exportar as informações num formato *HTML* de muito fácil leitura e compreensão. A adoção de um *plugin* capaz de transpor comentários em código para um *frontend web*, fez com que o desenvolvimento pudesse continuar ao mesmo ritmo que anteriormente, não sendo necessário manter vários registos distintos relativos a funcionalidades do sistema. Ao se juntar comentários e código, é possível ter uma ideia concreta de como uma alteração ao código poderá afetar o funcionamento de certas funções. Assim notou-se que a utilização deste *plugin* contribuiu não só para um maior controlo sobre o código desenvolvido, como também possibilita a que facilmente se disponibilize uma API que poderá vir a ser consumida por outro tipo de serviços externos. Para que seja possível obter uma ideia mais precisa do que se fala acima podem-se observar as figuras nos anexos A.7 e C.3, mais concretamente as imagens que dizem respeito a um excerto da documentação no formato *web* e a um comentário diretamente em código (figuras 56 e 65 respetivamente).

Concluída esta implementação, existe um servidor que disponibiliza páginas aos clientes, utilizando sub-domínios distintos. Este mantém uma conexão por sockets entre clientes, possibilitando a troca de informação entre ambos. Além disto também já se utiliza documentação, melhorando a compreensão do código desenvolvido. De seguida falta decidir como efetuar o envio de informação entre clientes. Enviar-se-ia tudo para todos (numa espécie de *broadcast*)? Filtrar-se-ia clientes de acordo com critérios específicos? Claro que para um número reduzido de clientes será possível manter a plataforma com uma metodologia de *broadcast*, mas seria esta opção viável a longo prazo? Naturalmente que não! É necessário ter em mente que existe a possibilidade de ocorrer um elevado número de mensagens (cada cliente emite uma sempre que a sua posição altera). Dado que estamos a

tratar de veículos, e que estes tendem a mover-se a velocidades consideráveis, a sua posição altera-se constantemente. Por este motivo existe um limite (que será quase sempre atingido) onde os clientes apenas poderão enviar uma atualização de posição por segundo. Assim se vê que numa plataforma com um volume de 1% da população portuguesa seria necessário de tratar de 10000 pedidos por segundo. Falta testar qual seria então a melhor forma de filtrar estes pedidos, para que cada cliente não recebesse todas as atualizações provenientes de outros clientes.

Este desafio é mais complexo do que parece. Lembre-se que temos uma limitação na capacidade computacional do servidor, pelo que neste caso não será viável utilizar este para fazer um processamento complexo da informação. Contudo, num cenário ideal onde não existisse estes limites, seria possível manter um registo a nível de servidor de todos os clientes que se encontram num certo raio uns dos outros. Isto permitia que o servidor reencaminhasse automaticamente as alterações de posição a potenciais clientes que possam estar numa área de maior risco de colisão. Esta abordagem utilizaria memória para manter um registo de distancias entre clientes (ou mecanismo semelhante) e utilizaria processamento para calcular estas distâncias, além que existiria a necessidade de atualizar a lista de vizinhos sempre que um utilizador entre ou saia desta mesma área. Na abordagem implementada, tentou-se filtrar as mensagens recebidas de forma semelhante, apesar das limitações que existem. O mecanismo utilizado foi limitar entre grelhas de latitude e longitude. Esta solução não necessitaria de manter registo de posições de clientes, sendo que o servidor apenas teria de ler as posições provenientes na atualização de posição e apenas procedia ao seu reenvio caso a localização estivesse entre os limites definidos. Com esta abordagem seria até possível ter um sistema de filtragem em níveis sendo que estes seriam representativos de áreas cada vez menores, mediante a utilização naquela zona. Para efeitos de teste, como apenas existe uma instância do servidor, apenas se garante a transmissão de mensagens numa só área, sendo que se um cliente se apresentar numa zona onde a posição esteja fora da área pré-definida será apresentada uma mensagem de erro.

Esta filtragem, juntamente com decisões efetuadas no algoritmo de decisão já foram exploradas na secção anterior. Saiba-se que cada pedido proveniente do sub-domínio **V2P** passa pelo mesmo filtro de posição que o **API** dado que o primeiro utiliza mecanismos do segundo diretamente na sua implementação. Isto significa que para efeitos práticos é possível existirem utilizadores em páginas *web* distintas, que obtenham as posições de outros utilizadores. Esta abordagem faz com que se diminua a quantidade de código em duplicado, para que ocorram menos erros. A nível de algoritmo de decisão, este foi implementado no lado do cliente, para minimizar os recursos necessários no servidor. Para tal considerou-se que os equipamentos capazes de obter a sua localização já são suficientemente sofisticados para conseguir efetuar o seu próprio processamento (dada a tal filtragem inicial de pedidos). Esta alternativa faz com que exista um cálculo múltiplo de distâncias

entre clientes, mas como isto é efetuado de forma distribuída não existe um peso significativo no correto funcionamento da plataforma. O algoritmo está estruturado quase como que por filtros. Cada filtro têm parâmetros mais rígidos, estando cada nível associado a um código de alerta. Quanto maior a probabilidade de colisão, maior o código de erro. Este algoritmo filtra clientes que se encontrem a uma distância **X** do utilizador, clientes que tenham rota de colisão numa distância de **Y**, clientes que se dirigem para o ponto de colisão e o cenário em que o utilizador também se dirige para este ponto. Todos os avisos emitidos são guardados para que posteriormente se possam obter estatísticas que comprovem o funcionamento do algoritmo. Uma verificação simples é a de que quanto mais baixo for o código de alerta, maior deverá ser a frequência com que este ocorre.

Considera-se que a plataforma está funcional quando se consegue detetar com sucesso um possível cenário de colisão. Neste momento a plataforma já é capaz de o fazer. De seguida otimizou-se a interface com o utilizador, melhorou-se a eficiência com que os dados eram obtidos da base de dados e definiram-se mecanismos que permitissem com que estes melhoramentos não originassem uma falha num sistema que já se encontrava funcional. Fala-se, claro, da utilização de mecanismos de integração contínua, mais concretamente da sua necessidade de testes unitários. Desde o momento em que certas funcionalidades foram definidas e se encontravam funcionais existia a necessidade de garantir que estas se manteriam desta forma após alterações ou acrescentos ao código. Deste modo, utilizou-se uma biblioteca adicional JavaScript denominada de *Jest*. Esta biblioteca permite gerar testes unitários, também em JavaScript, para testar os resultados produzidos por certas funções do código o sistema. Testa-se os resultados obtidos pelas bases de dados, os resultados de funções e garante-se que no caso de ocorrência de erros, estes são tratados de forma adequada. Um exemplo de implementação pode ser observado no anexo C (figura 67). Neste exemplo verifica-se a existência de *queries* disponíveis no sistema e tenta-se, por exemplo, criar um novo utilizador.

Visto que o desenvolvimento do código para a plataforma não era efetuado na mesma máquina onde o servidor se encontrava em execução, utilizou-se os testes unitários como intermediário de análise de erros em código. Sempre que se pretendia efetuar uma alteração era necessário executar manualmente os testes, posteriormente fazer o *upload* da alteração para o repositório e, já no lado do servidor, fazer o *download* e reiniciar o servidor com as alterações efetuadas. Para tentar agilizar este processo introduziu-se o CircleCI. Este disponibiliza ferramentas de integração contínua que interagem diretamente com o *GitHub* (onde o código se encontra). Como já se explicou anteriormente o método de funcionamento, explica-se agora a configuração utilizada, disponível no anexo C (figura 68). A configuração utilizada procura imagens do *NodeJs* e do *PostgreSQL* para conseguir executar em simultâneo o servidor aplicacional e a base de dados, respetivamente. O passo seguinte após obter estas imagens passa por obter o código do repositório juntamente com a

instalação de possíveis bibliotecas que o projeto necessite. Como este processo pode implicar que sejam transferidos vários *gigabytes* de informação, utiliza-se mecanismos de cache. Estes garantem que apenas se transferem dependências, nos caso em que se adiciona ou remove bibliotecas do projeto, ao invés de se transferir sempre que uma alteração é efetuada. Este serviço executa os testes unitários automaticamente, produzindo como resultado uma das mensagens apresentadas na figura seguinte.

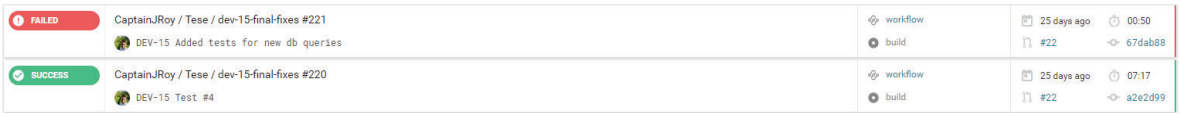


Figura 33.: Exemplo de resultados obtidos com falha e sucesso de execução de testes unitários

4.4 Arquitetura do Sistema

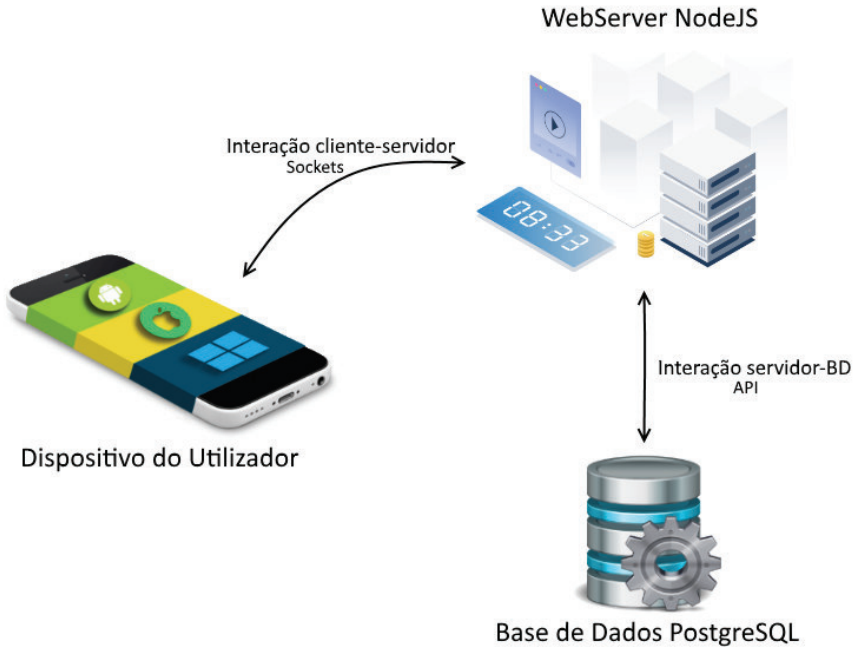


Figura 34.: Esquema da comunicação entre o cliente (WebApp) e o servidor (NodeJS)

Tal como é esquematizado na figura acima, o utilizador apenas pode efetuar contacto com o servidor *web*, não sendo possível comunicação direta com o sistema de base de dados. A comunicação aqui é inicialmente feita para registo de um novo utilizador (utilizando o *template* disponível no anexo A, figura 50), para *login* de utilizadores já existentes (utili-

zando o *template* anexo na figura 51), para apresentação da interface *web* do sistema (com o *template* descrito em anexo, figura 53), entre outros. No entanto esta comunicação também pode ser utilizada para verificar se um utilizador se encontra registado no sistema e retirar possíveis informações de utilizadores como é o caso da foto de perfil que estes disponibilizam. Esta comunicação é semelhante ao que aconteceria no servidor para comunicar com a base de dados, no entanto existe uma filtragem dos atributos existentes no modelo de dados relacional para não deixar escapar informação relevante (como um *hash* das palavras passe, por exemplo) para qualquer utilizador *web*. Novamente, poderá ser observado um esquema da informação disponibilizada relativamente a todos os utilizadores registados na plataforma (através da figura 58 do anexo B), e da informação disponibilizada para um utilizador específico (visível na figura 59 do anexo B).

Assim pode-se descrever uma interação completa do utilizador no sistema desenvolvido como:

- Registo do utilizador na plataforma;
- Inicio de sessão por parte do utilizador;
- Redirecionamento para a página de perfil do utilizador;
- O utilizador poderá agora utilizar o chat desenvolvido para testes (*template* no anexo A, figura 54);
- O utilizador pode ainda aceder à API inicial da aplicação (*template* no anexo A, figura 55);
- Finalmente é possível aceder à plataforma de alerta presente na figura 53 do anexo A.

A utilização da plataforma é muito simples, simplesmente é necessário dar acesso à utilização da localização por parte do *browser*. Feito isto o dispositivo do utilizador começará a emitir a sua localização para o servidor, sendo que este ficará responsável por filtrar as mensagens e fazer com que estas cheguem a outros utilizadores (na mesma área de interesse). Sempre que a localização do utilizador atualizar, esta será novamente enviada ao servidor para que os cálculos de possíveis rotas de colisão sejam sempre o mais atuais e precisos possíveis.

4.5 Arquitetura de desenvolvimento

É natural que durante a fase de desenvolvimento de qualquer projeto seja necessário voltar atrás e refazer certas funcionalidades. Assim se passa o mesmo com o nosso caso. A arquitetura presente na figura 35 pretende fazer com que alterações efetuadas no código sejam testadas antes de que este possa causar algum tipo de problemas.

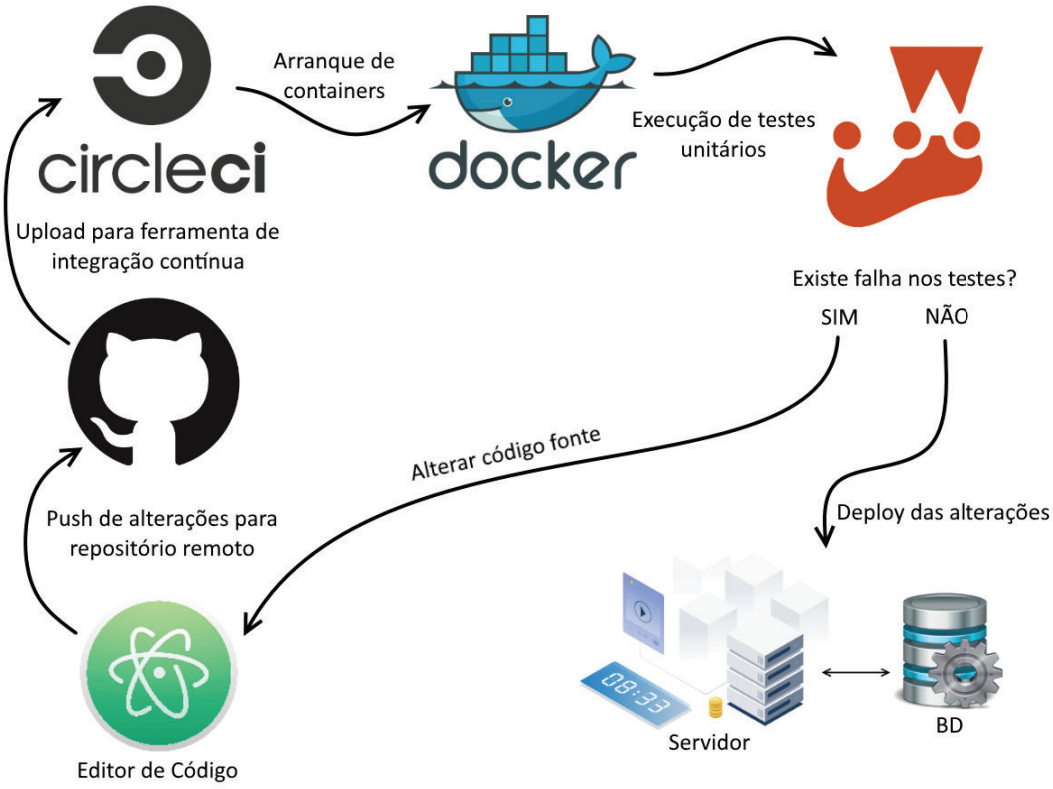


Figura 35.: Esquema da arquitetura de desenvolvimento para o servidor

Para o desenvolvimento de um projeto de *software* será sempre necessário um editor de código, aqui utilizou-se o *Atom* que permite uma integração direta com o *GitHub* (o repositório remoto onde se encontra o nosso código desenvolvido). O *GitHub* serve apenas para controlo de versões neste caso, uma vez que é interessante existir a possibilidade de desenvolver várias tarefas em simultâneo e, no fim, juntar tudo sem que ocorram problemas. O *CircleCI* aqui age apenas como motor que nos permite executar testes unitários sobre alterações efetuadas no código. Este arranja *containers docker* que depois executam automaticamente testes unitários utilizando a biblioteca *Jest*. Caso estes testes sejam efetuados com sucesso, o código é então atualizado para o servidor que está a correr todo o sistema, caso contrário não será possível adicionar as alterações (uma vez que algum teste falhou) dado que uma funcionalidade atualmente correta, deixaria de funcionar.

É importante referir que estes testes unitários são executados não só aquando de alterações ao código em si, como também em alterações à estrutura da base de dados. Atualmente sempre que se pretende adicionar uma tabela nova é preciso executar uma migração de todo o sistema de dados. Isto torna-se um processo lento quando a quantidade de dados a migrar é elevada, mas assegura-nos não só a coerência dos dados como garante retrocompatibilidade do sistema *webapp* com o sistema de dados.

Um exemplo de migração poderá ser visto no código presente no anexo C (63).

4.6 SIMULAÇÃO E RESULTADOS

Um dos passos fundamentais para o correto funcionamento da plataforma, passa pelo teste da solução desenvolvida. Neste sentido, foi necessário implementar mecanismos capazes de replicar vários utilizadores simultaneamente para que posteriormente se pudesse detetar possíveis cenários de perigo. Tinha sido posta a hipótese de utilizar um simulador (mais concretamente o simulador *SUMO*), no entanto considerou-se que o esforço necessário para implementar este cenário de testes seria mais elevado do que realizar a nossa própria solução. Sabemos que para simular um cenário de teste convém que existam pedestres que se desloquem em passeios (ou esporadicamente em estradas) e veículos (que circulem em estradas). Sabe-se ainda que a velocidade dos veículos é por norma mais elevada que a velocidade dos peões. Como se procederia para gerar um caminho, num mapa especifico, para que fosse possível testar o algoritmo? E se fosse possível gravar um percurso e posteriormente o replicá-lo? Seria possível? Sim! E se se utilizasse os *logs* do servidor para deslocamentos que já haviam ocorrido anteriormente então já existiam vários caminhos disponíveis. Foi por esta abordagem que se adotou uma proposta de cenário para simulação de resultados.

O primeiro desafio no desenvolvimento do simulador passou por conseguir arranjar informação suficiente para que o simulador conseguisse replicar. Para isto recorreu-se aos *logs* que já existiam no sistema. Visto que esta preocupação só ocorreu vários meses depois de começar a gravar percursos, já existiam pequenos percursos pré-definidos que poderiam voltar a ser utilizados como cenários de teste. O segundo desafio passou por implementar uma interface *frontend* simples que consuma os dados e utilize os mesmos *endpoint* para comunicação de posição (neste caso continua dependente do sub-domínio **API**). Opcionalmente procurou-se um serviço que conseguisse mapear os pontos gravados para que fosse mais fácil comprovar que a rota gravada realmente se encontrava correta.

4.6.1 *Script em Python*

Para que se conseguisse rapidamente tratar dos dados gravados anteriormente, desenvolveu-se um *script* em *Python* que lesse informação do ficheiro de registos / *logs* (que seria dado como *input*) e produzisse um output que seria lido pelo simulador.

A figura 36 mostra os dados que são recebidos pela *script* em *Python*, note-se que esta figura apenas apresenta 8 resultados representantes de cerca de 8 segundos de deslocamento. Estes são equivalentes às informações guardadas sempre que um cliente se encontre a utilizar a plataforma. É possível verificar as informações que são trocadas entre

```
1 {"subdomain":"api","username":"captainroy","timestamp":1555595960929,"accuracy":9.64799976348877,"altitude":72.1474609375,"altitudeAccu
racy":null,"heading":325,"latitude":37.017714,"longitude":-8.9553888,"speed":16.6200008392334,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
2 {"subdomain":"api","username":"captainroy","timestamp":1555595961930,"accuracy":6.432000160217285,"altitude":72.8619384765625,"altitude
Accuracy":null,"heading":326,"latitude":37.0177359,"longitude":-8.9554991,"speed":16.739999771118164,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
3 {"subdomain":"api","username":"captainroy","timestamp":1555595962929,"accuracy":5.360000133514404,"altitude":72.1708984375,"altitudeAcc
uracy":null,"heading":325,"latitude":37.0178461,"longitude":-8.9556209,"speed":16.610000610351562,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
4 {"subdomain":"api","username":"captainroy","timestamp":1555595963930,"accuracy":5.360000133514404,"altitude":74.14813232421875,"altitud
eAccuracy":null,"heading":326,"latitude":37.0179574,"longitude":-8.9557117,"speed":16.940000534057617,"marker":"marker-md-0B1D35-
0B1A2E-captainroy.png"}
5 {"subdomain":"api","username":"captainroy","timestamp":1555595964930,"accuracy":4.288000106811523,"altitude":73.095458984375,"altitudeA
ccuracy":null,"heading":326,"latitude":37.0181176,"longitude":-8.955812,"speed":16.81999969482422,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
6 {"subdomain":"api","username":"captainroy","timestamp":1555595965929,"accuracy":4.288000106811523,"altitude":77.4869384765625,"altitude
Accuracy":null,"heading":326,"latitude":37.0182301,"longitude":-8.9559079,"speed":16.56999969482422,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
7 {"subdomain":"api","username":"captainroy","timestamp":1555595966925,"accuracy":4.288000106811523,"altitude":75.38677978515625,"altitud
eAccuracy":null,"heading":327,"latitude":37.0183754,"longitude":-8.9560341,"speed":16.40999984741211,"marker":"marker-md-0B1D35-0B1A2E-
captainroy.png"}
8 {"subdomain":"api","username":"captainroy","timestamp":1555595968000,"accuracy":3.9000000953674316,"altitude":70.9847412109375,"altitud
eAccuracy":null,"heading":326,"latitude":37.0185108,"longitude":-8.9561585,"speed":15.960000038146973,"marker":"marker-md-0B1D35-
0B1A2E-captainroy.png"}
```


Figura 36.: Exemplo de dados provenientes do ficheiro de registos do servidor / *logs*

clientes, existe um *timestamp* associado à assinatura temporal de quando uma atualização foi produzida, existe uma *accuracy* que permite informar a exatidão (em metros) da posição fornecida, e existem mais informações relevantes como altitude, latitude, longitude, velocidade e direção. Todas estas deverão ser utilizadas para formar um novo ficheiro que será consumido pelo simulador. Encontra-se no anexo C, figura 69, um excerto do *script* desenvolvido para este efeito. Este gera não só um ficheiro para o simulador, mas também gera um ficheiro contendo uma série de posições GPS no formato *CSV* que podem ser lidas pela plataforma *GPSVisualizer*. A *dashboard* e exemplo desta plataforma podem ser vistas no anexo D, figura 74.


```
{"0": {"time_delta": 0, "latitude": 41.555493, "altitudeAccuracy": null, "altitude": 207.6717529296875, "speed": 0, "heading": null,
"longitude": -8.401529, "accuracy": 10.15999984741211}, "1": {"time_delta": 1000, "latitude": 41.5554535, "altitudeAccuracy": null,
"altitude": 232.9779052734375, "speed": 0.20999999344348907, "heading": 67, "longitude": -8.4015205, "accuracy": 8.21399974822998}, "2":
{"time_delta": 1000, "latitude": 41.555431, "altitudeAccuracy": null, "altitude": 234.4156494140625, "speed": 0, "heading": null,
"longitude": -8.4015319, "accuracy": 6.504000186920166}, "3": {"time_delta": 1000, "latitude": 41.555415, "altitudeAccuracy": null,
"altitude": 235.1549072265625, "speed": 0.1599999964237213, "heading": 179, "longitude": -8.4015324, "accuracy": 4.85699987411499}, "4":
{"time_delta": 1000, "latitude": 41.5554098, "altitudeAccuracy": null, "altitude": 237.318115234375, "speed": 0, "heading": null,
"longitude": -8.4015354, "accuracy": 4.550000190734863}, "5": {"time_delta": 1000, "latitude": 41.555405, "altitudeAccuracy": null,
"altitude": 238.2099609375, "speed": 0.28999999165534973, "heading": 225, "longitude": -8.4015412, "accuracy": 4.077000141143799}, "6":
```


Figura 37.: Exemplo de dados obtidos pela execução do script desenvolvido para consumo do simulador

A figura 37 apresenta o resultado que se obtém quando se executa a *script* desenvolvida sobre dados com estrutura semelhante aos presentes na figura 36. Esta apenas contém informação relativa às 5 posições e modifica ligeiramente os dados antes de os adaptar para o simulador. Nesta figura é possível observar que o *timestamp* foi omitido e substituído por um *time_delta*. Este último é nada mais nada menos que a diferença temporal

(em milissegundos) entre duas mensagens consecutivas do ficheiro de *logs* recebido como *input*. Isto é utilizado para que se consiga gerir quando existe a necessidade de emitir uma nova atualização de posição. Sempre que o simulador executar, e visto que este é também desenvolvido em *JavaScript* podemos recorrer a uma função que nos permite definir uma espécie de temporizador. Esta função evita execuções desnecessárias ou esperas ativas até que o simulador esteja preparado para enviar uma nova mensagem. Este *delta_time* é utilizado precisamente para definir esse temporizador. Os restantes dados são transmitidos como aconteceria num utilizador habitual.

	elevation	latitude	longitude
2	207.67175293	41.555493	-8.401529
3	232.977905273	41.5554535	-8.4015205
4	234.415649414	41.555431	-8.4015319
5	235.154907227	41.555415	-8.4015324
6	237.318115234	41.5554098	-8.4015354
7	238.209960938	41.555405	-8.4015412
8	237.554870605	41.5554054	-8.4015522

Figura 38.: Exemplo de dados obtidos pela execução do script desenvolvido para consumo da ferramenta *GPSVisualizer*

A figura 38 demonstra o outro tipo de *output* que é produzido quando se executa a *script*. Este ficheiro é exclusivamente para consumo por parte da plataforma *GPSVisualizer*, e serve para demonstrar se o percurso gravado é realmente o que foi efetuado. Este ficheiro contém informações relativas a todos os pontos detetados durante um percurso, sendo que tem a capacidade de tornar muito mais intuitiva a compreensão da informação que estamos a ler.

Quando se vê a informação a ser trocada no servidor apenas se observa estes números de altitude, latitude e longitude, e não se consegue ter uma ideia clara de que realmente o sistema funciona com um certo nível de precisão. Contudo quando se utiliza esta ferramenta percebe-se que realmente a informação está a ser adquirida com um nível de erro bastante reduzido, e que é adquirida com a frequência suficiente para que se consiga traçar a rota apresentada na figura 39.

4.6.2 *Simulador web frontend*

O segundo passo de implementação para os cenários de teste passou pelo desenvolvimento de um servidor *web*. Para tal foram utilizados os dados que foram obtidos pela *script* em *Python*. O objetivo do simulador (o qual pode ser observado no anexo A, figura 57) é reenviar atualizações de pontos / coordenadas que já haviam sido transmitidas anteriormente. Esta ferramenta começou por apenas permitir a retransmissão de rotas, no entanto, poste-

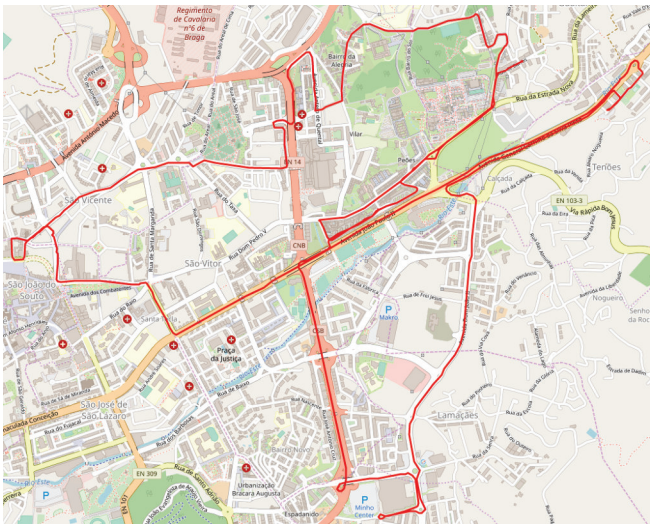


Figura 39.: Mapa obtido pela execução do script sob uma rota gravada anteriormente, utiliza a ferramenta *GPSVisualizer*

riormente adicionou-se a opção de adaptar uma rota gravada num veículo para uma rota que seria efetuada por um pedestre. Para conseguir isto adiciona-se um erro ao ponto transmitido (simulando que o peão se encontra na berma) e diminui-se a velocidade de deslocamento para que esta seja inferior a 4 metros por segundo (velocidade à qual deixa de ser considerada um pedestre).

Além de fazer esta melhoria, foi pensado que se poderia aproveitar um percurso para conseguir distribui-los por vários veículos distintos. Na verdade entende-se que faz sentido também percorrer esse mesmo percurso no sentido oposto ao gravado, então porque não permitir tudo isto? Este foi o próximo passo. Sempre que se pretenda testar um caso de cenário de perigo, faz-se o *upload* do ficheiro obtido anteriormente, e escolhe-se a opção pretendida. Neste caso utilizaremos o simulador para simular deslocação de 10 carros distintos sobre uma outra rota gravada anteriormente.

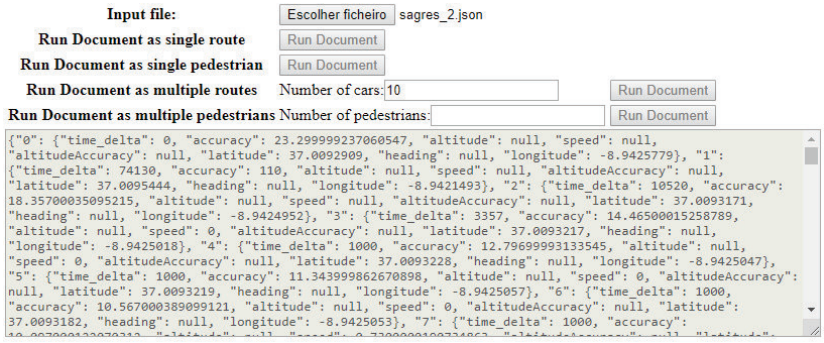


Figura 40.: Frontend web quando se executa uma rota pré-gravada

A figura 40 apresenta o resultado obtido no *frontend web* quando se executa um caminho gravado anteriormente. É possível ver uma pré-visualização dos dados que serão transmitidos.

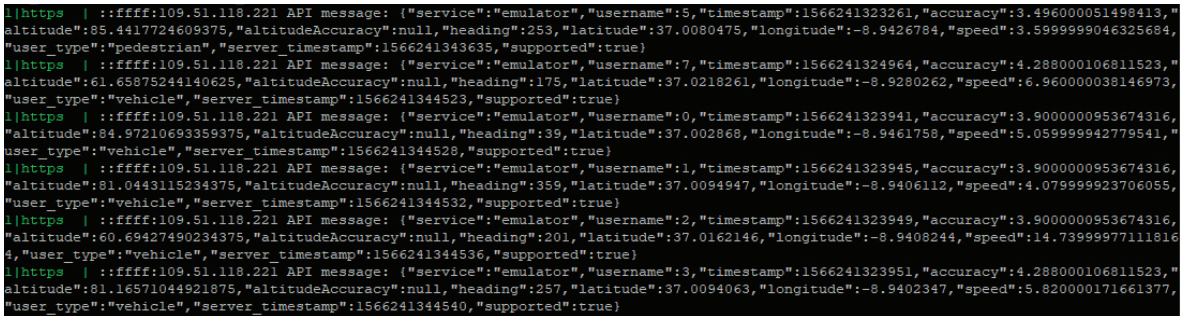


Figura 41.: Registo de atividade detetado no servidor

A figura 41 apresenta os registos de atividade que a emissão / simulação presentes na figura 40 originaram. Com a observação desta imagem é possível observar que o serviço que emitiu esta atualização foi o *emulator*. Nesta imagem ainda é possível observar que o simulador atribui autonomamente nomes de utilizadores conforme o id do veículo que transmite a mensagem. Neste exemplo é possível observar seis utilizadores distintos dos 10 que se encontravam em circulação. Apesar de tudo parecer correto, e de realmente existir um fluxo de informação no servidor de cerca de 10 mensagens novas por segundo, não se percebe facilmente que a solução proposta está a funcionar, ou que os pontos que são transmitidos dizem realmente respeito a um veículo.

O que se fez de seguida foi tentar arranjar um método que demonstrasse em tempo real onde se encontravam os outros utilizadores da plataforma. Para que tal fosse possível desenvolveu-se ainda mais o sub-domínio **API** para que este pudesse conter a imagem / mapeamento de coordenadas latitude e longitude num mapa. Um *template* desta interface pode ser visto no anexo A, na figura 55, ainda assim para que seja mais percetível demonstrar o que se trata, apresente-se a imagem do estado do mapa da **API** após execução do simulador com os ditos 10 veículos.

A figura 42 apresenta precisamente esta pequena simulação de teste. É possível observar os pontos relativos aos 10 clientes distintos, embora alguns deles comecem a ter rotas que coincidem umas com outras.

Tal como já foi referido anteriormente, o desenvolvimento desta solução foi feito totalmente em *JavaScript*. Para desenvolver o *frontend* utilizou-se uma API pública conhecida por *MapQuest.js*. Esta permite traçar num mapa um certo ponto, dado por valores de latitude e longitude. Esta API é pública e *open-source*, contudo tem limites de utilização que não tinham sido considerados corretamente. Esta biblioteca permite apenas que sejam efetuados 15 mil pedidos por mês. À primeira vista pode parecer suficiente para apresentar

Figura 42.: Registo de atividade detetado no sub-domínio **API**

a todos os utilizadores, mas rapidamente se percebe que se todos os utilizadores tiverem acesso a esta plataforma, não seria possível conseguir obter informação de forma fidedigna. Rapidamente se esgotariam estes pedidos, e ficaríamos limitados a acreditar que o sistema continuaria a funcionar mesmo sem ser apresentadas as atualizações de posição.

Para tentar anular este problema, a *dashboard* que os clientes utilizarão para aceder à plataforma não apresentará a localização de outros clientes. Assim retira-se um pequeno processamento adicional, e não iniciamos vícios indesejados aos clientes. Se não existir a possibilidade de apresentar mapa, então eventualmente estes habituam-se à forma como os avisos serão apresentados. Além disto, os custos para manter esta opção viável não fazem sentido numa fase de protótipo.

4.7 Sumário de desenvolvimento

Conclui-se este capítulo com a noção de que o desenvolvimento desta plataforma compreendeu desafios não só de programação, como de resolução de problemas de comunicação.

Aqui demonstrou-se como se fez o *setup* inicial do servidor, que mecanismo de persistência de dados foi utilizado, que ferramentas de implementação é que foram escolhidas. Percebeu-se que o desenvolvimento de um projeto complexo, de raiz, implica uma estrutura

bastante sólida de desenvolvimento, não só para que o progresso ocorra mais rapidamente mas para que os problemas sejam resolvidos com uma maior brevidade.

Neste capítulo também se explicaram algumas decisões tomadas para que o sistema funcionasse corretamente assim como o motivo de estas funcionarem em detrimento de outras soluções. Desenvolveu-se o algoritmo de previsão que consegue separar avisos em 3 níveis distintos (inicialmente), e que será totalmente compatível com quaisquer melhoramento futuro. O desenvolvimento serviu para ter noções práticas de como se faz o *deploy* de um servidor de raiz, que fica funcional, bem como eventuais problemas que a manutenção deste possa originar. Observou-se a necessidade de confirmação de testes e simulação de cenários, e refletiu-se sobre a necessidade de apresentar estes de forma mais imediata aos utilizadores.

Notou-se, acima de tudo, que foi tão importante desenvolver uma arquitetura aplicacional de funcionamento, como uma arquitetura de desenvolvimento que permitisse a manutenção do projeto.

5

CASOS DE ESTUDO E TESTES REALIZADOS

Neste capítulo apresentam-se os casos de estudo usados para demonstrar e avaliar o sistema proposto no âmbito desta dissertação, ou seja, os testes que foram realizados. Inicialmente é descrita a forma como foi estabelecido o *setup* inicial para testes, posteriormente apresenta-se os resultados obtidos juntamente com uma análise dos mesmos. Por se ter optado pelo desenvolvimento de um sistema de testes de raiz, testam-se os vários passos necessários até chegar ao funcionamento do sistema final.

5.1 SETUP DOS TESTES

Foram preparados dois cenários distintos para teste: um onde é esperado que se detete um possível cenário de colisão e outro em que tal não suceda.

Para explorar ambas as hipóteses gravaram-se dois percurso com duração de cerca de 30 minutos de deslocamento cada. O primeiro foi um percurso por Braga, visível na figura 39, o segundo define um percurso por Sagres, visível na figura 43.

Para que fosse possível testar mais cenários, os limites das grelhas de latitude e longitude foram ajustados convenientemente de modo a permitir a inclusão de utilizadores localizados em todo o país, excepto ilhas (apenas Portugal continental).

O percurso da figura 39 será utilizado para detetar possíveis cenários de perigo. Para tal o utilizador que pretende testar o funcionamento da plataforma precisará de se encontrar a uma distância mínima de 500 metros de um dos pontos do percurso. Além disso, é necessário que o utilizador se desloque ligeiramente. Este passo é essencial visto que para que o sistema comece a funcionar é preciso serem registadas três atualizações de posição. Mal se acede à plataforma onde se pretende realizar os testes, é necessário introduzir o percurso no simulador como foi demonstrado no capítulo 4, secção 6. Deverá adaptar-se os valores dos parâmetros desejados. Se tudo correr bem será esperado receber vários avisos de nível 1, menos de nível 2 e muito poucos avisos de nível 3. Recorde-se que o nível do aviso é tanto maior quanto maior for o nível de perigo. Posteriormente será necessário confirmar que o algoritmo funcionou como esperado.

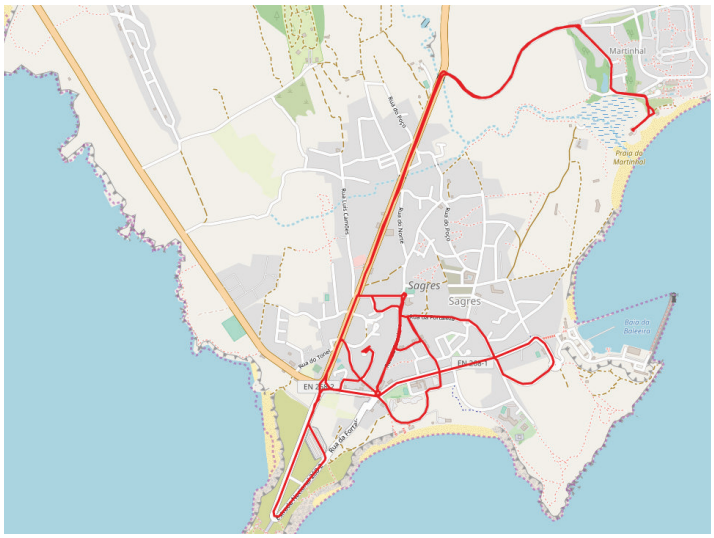


Figura 43.: Gravação de percurso adicional em Sagres

Ao longo da apresentação dos resultados, utilizam-se duas designações distintas: *utilizador* e *cliente*. Note-se que *utilizador* pode referir-se a qualquer agente que utilize a plataforma, ou todos os agentes simulados no simulador desenvolvido. Quando se refere a *cliente*, fala-se do agente sobre o qual se realiza os testes.

O percurso da figura 43 será utilizado para ilustrar um cenário onde não ocorre emissão de qualquer tipo de aviso. Este percurso foi gravado a uma distância superior a 500 metros (cerca de 500 quilómetros em linha reta) pelo que a apresentação de um aviso básico de nível 1, representará um erro na plataforma. Para testar este cenário, é necessário inicializar a *dashboard* do sistema tal como no cenário anterior (esta página é a que se encontra no anexo A, figura 53). É também necessário que o utilizador se desloque ligeiramente para registar pontos iniciais de localização. Após este *setup* inicial pode-se fazer o *load* do novo percurso para o simulador e atualizar os parâmetros desejados. Visto que não deverão ser apresentadas quaisquer informações ao utilizador, observar-se-á os registos que estão a ser emitidos para o servidor para demonstrar que efetivamente o simulador está em funcionamento e que a plataforma apenas não deteta nenhum cenário de perigo.

5.2 TESTES E RESULTADOS

O primeiro passo para testar as funcionalidades do sistema desenvolvido, é aceder à plataforma. Para que seja possível aceder o utilizador terá de se registar no portal apresentado no anexo A, figura 50. Caso o registo seja efetuado com sucesso pode-se avançar para o próximo passo, iniciar sessão na plataforma, o que pode ser realizado através da página *web* visível no anexo A, figura 51. Quando a sessão é iniciada com sucesso, o utilizador é

automaticamente redirecionado para o seu perfil. Este tem algumas informações relativas à sessão do utilizador, e outras informações utilizadas como *placeholders* temporários. A página de perfil (igual à apresentada na figura 52 do anexo A) contém referências para os sub-domínios de **API** e **CHAT**, contudo estes apenas servem para testes, sendo que a plataforma final deverá ser acedida pelo subdomínio V2P.

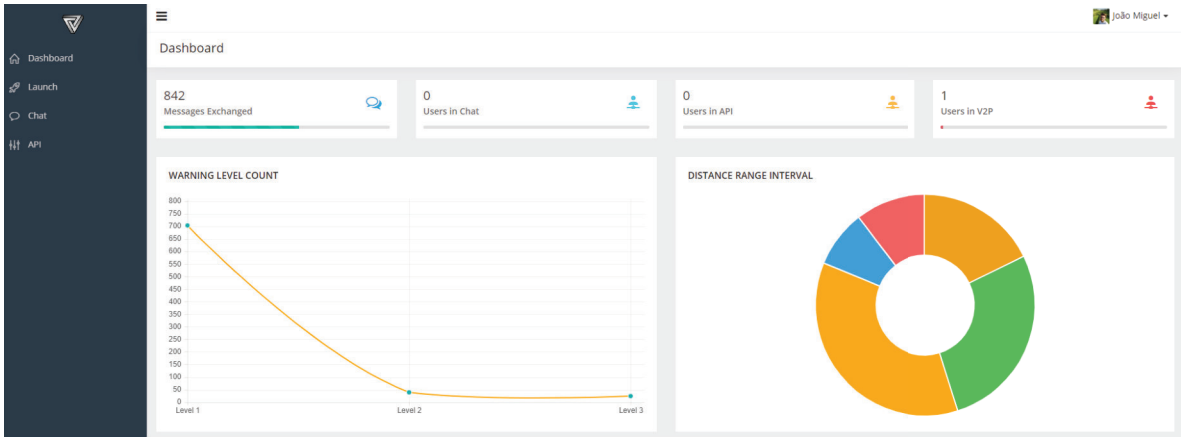


Figura 44.: Página inicial da plataforma V2P

A figura 44 apresenta o resultado que se obtém quando se acede à plataforma, aqui são observáveis alguns valores que já foram obtidos com outros testes bem como algumas métricas gerais. Para aceder à página de utilização do sistema deve-se aceder à aba que se encontra no menu esquerdo com a etiqueta "*Launch*". O resultado do acesso a esta nova página, pode ser visualizado na figura 53 do anexo A.

Agora com o sistema ativo, o utilizador apenas terá de se movimentar ligeiramente, para registar 3 atualizações de posição. Para *developers* pode-se aceder através da consola existente no *browser* (neste caso *Google Chrome*) ao objeto denominado de *neighbours*. Este apresenta a informação utilizada pelo algoritmo para calcular as regressões lineares relativas ao movimento dos utilizadores. Quando o parâmetro *linearRegression* se encontra definido, significa que já foram obtidas 3 atualizações de posição, e que o algoritmo se encontra em funcionamento.

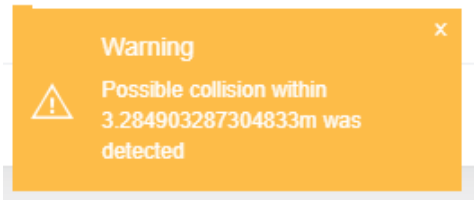


Figura 45.: Aviso de colisão com automóvel do simulador

O próximo passo é aceder à plataforma de simulação, apresentado no anexo A, figura 57. Para este teste utilizou-se o percurso gravado em Braga, figura 39, com um teste de execução de 10 veículos simultaneamente. Após alguns segundos nota-se um alerta na plataforma, visível na figura 45. Este indica que nos encontramos a uma distância de cerca de 3 metros de um possível ponto de colisão.

Se agora acedermos aos valores guardados no objeto *neighbours*, notar-se-á que existem mais valores além dos do cliente. A figura 46 monstra estes valores. O objeto *neighbours* contém mais objetos, sendo que cada um é identificável pelo nome do utilizador que o enviou. Quando uma atualização é emitida pelo simulador, o nome de utilizador é alterado para conter o *ID* do veículo que procedeu à emissão da atualização.

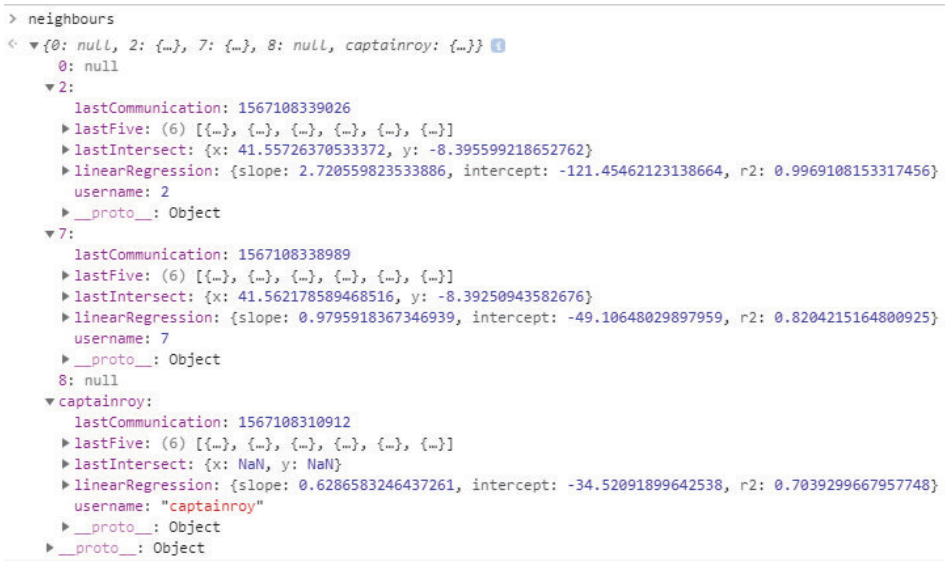


Figura 46.: Atualização do objeto *neighbours* na plataforma V2P

Os utilizadores zero e oito, têm o valor nulo atribuído dado que não comunicam atualizações de posição há mais de 1 minuto. No entanto o cliente deteta a existência de dois outros utilizadores que podem estar na sua zona de colisão. Note-se que neste caso o cliente que está a utilizar a aplicação é denominado como "captainroy", enquanto os outros utilizadores são designados por um *id* que varia de zero a nove. Para que seja mais percetível ao leitor, traçou-se na figura 47 um esboço das linhas de movimento dos três utilizadores, tendo-se identificado a sua posição atual aproximada. É possível observar que todos se encontram a uma distância de até 500 metros entre si, pelo que foram guardados neste registo do objeto *neighbours*. O utilizador 7 encontra-se muito próximo do utilizador teste *captainroy* / cliente, contudo as suas retas de deslocamento são bastante distintas, este facto faz com que estas se intersetem, naquele momento, num ponto que se encontra a cerca de 3 metros de distância. Note-se que este valor será atualizado sempre que o utilizador peão ou veículo

se desloquem, pelo que poderá ser emitido um novo aviso sempre que estes se aproximem ou afastem. O utilizador 2 também tem um ponto de interseção num raio de 500 metros em comum com o cliente *captainroy*, no entanto dada a eventualidade de um destes se encontrar a deslocar em sentido oposto a este ponto, pode-se concluir que não foi emitido qualquer tipo de aviso de nível 3.

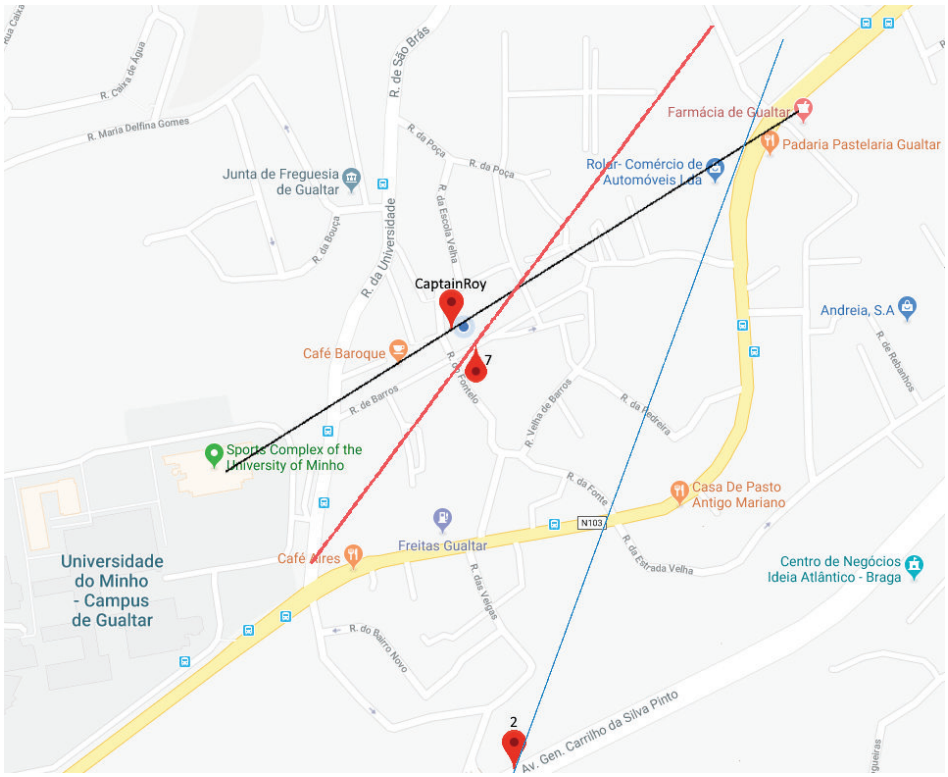


Figura 47.: Esboço de possíveis colisões detetadas no teste em Braga

A figura 48 apresenta os avisos emitidos pelo cliente, quando foi detetado um cenário de colisão. Ao observar esta figura, nota-se que o cliente 7 teve pontos de interseção cuja distância variou entre 3 e 161 metros. Pela observação do aviso de nível 3 com o *ID* 1057 conclui-se que ambos os utilizadores se encontram em deslocamento no sentido do ponto de colisão. Esta diferença entre distâncias de colisão pode ser originada por uma alteração na direção de deslocamento. Dadas as retas visíveis na figura anterior, consegue-se entender com alguma facilidade que um novo cálculo de regressão linear, com uma nova posição originará uma reta com um declive ligeiramente diferente. Esta diferença, apesar de pequena é suficiente para originar os incrementos e decrementos de distância apresentados. Relativamente ao número de mensagens trocadas durante o teste, dado que quando se começou o teste já tinham sido trocadas em testes anteriores 842 mensagens (conforme a figura 44) e no final do teste foram trocadas 1247 mensagens, podemos concluir que durante este teste foram emitidas um total de cerca de 405 mensagens de atualização de posição.

{"id":1050,"level":1,"heading_towards":null,"distance":46,"detected_by":"captainroy","other_user":"7"},{"id":1051,"level":1,"heading_towards":null,"distance":159,"detected_by":"captainroy","other_user":"7"},
{"id":1052,"level":1,"heading_towards":null,"distance":17,"detected_by":"captainroy","other_user":"7"},{"id":1053,"level":1,"heading_towards":null,"distance":12,"detected_by":"captainroy","other_user":"7"},
{"id":1054,"level":1,"heading_towards":null,"distance":7,"detected_by":"captainroy","other_user":"7"},{"id":1055,"level":1,"heading_towards":null,"distance":3,"detected_by":"captainroy","other_user":"7"},
{"id":1056,"level":1,"heading_towards":null,"distance":3,"detected_by":"captainroy","other_user":"7"},{"id":1057,"level":3,"heading_towards":true,"distance":3,"detected_by":"captainroy","other_user":"7"},
{"id":1058,"level":2,"heading_towards":true,"distance":3,"detected_by":"captainroy","other_user":"7"},{"id":1059,"level":1,"heading_towards":null,"distance":5,"detected_by":"captainroy","other_user":"7"},
{"id":1060,"level":1,"heading_towards":null,"distance":15,"detected_by":"captainroy","other_user":"7"},{"id":1061,"level":1,"heading_towards":null,"distance":161,"detected_by":"captainroy","other_user":"7"},
{"id":1062,"level":2,"heading_towards":true,"distance":161,"detected_by":"captainroy","other_user":"7"},{"id":1063,"level":1,"heading_towards":null,"distance":90,"detected_by":"captainroy","other_user":"7"},
{"id":1064,"level":1,"heading_towards":null,"distance":147,"detected_by":"captainroy","other_user":"7"}]

Figura 48.: Atualização de posição em avisos emitidos durante o teste

O passo seguinte será testar com uma rota que se encontra a cerca de 500 quilómetros de distância. Para comprovar que não foi emitido qualquer tipo de aviso, note-se que o último *ID* de aviso guardado para o cliente *captainroy* foi o 1064. É esperado que quando se verificar novamente a listagem de avisos, esta se mantenha exatamente igual á presente na figura 48.

Neste momento resta aceder novamente à plataforma de simulação, introduzir o ficheiro de percurso de longa distância e tentar novamente executar a simulação. Desta vez deixou-se o contador de mensagens chegar às 2683 mensagens trocadas, tal como apresentado na figura 49, isto significa que durante esta nova simulação foram enviadas 1436 atualizações de posição, sendo que o cliente não detetou qualquer risco de colisão.

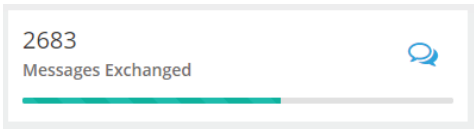


Figura 49.: Atualização de valor de posição enviadas pelo simulador

Além de observar o aumento no número de mensagens emitidas pode-se observar os registos no servidor onde se nota um fluxo de mensagens bastante significativo, cerca de 10 mensagens por segundo (equivalentes aos 10 veículos que se selecionaram novamente). Pode-se ainda ver que os registos de emissão de avisos do cliente *captainroy* continuam iguais aos apresentados na figura 48.

5.3 Discussão

Após verificar os resultados obtidos com o teste, nota-se que o algoritmo é bastante sensível a possíveis cenários de perigo. Este facto pode ser observado nos testes anteriores, onde uma alteração de direção pode alterar bastante a localização de eventuais pontos de colisão.

O funcionamento da solução atual passa por adquirir os últimos 5 pontos de localização do cliente, calcular uma reta que se aproxime o máximo de todos estes (cálculo de regressão linear) e depois verificar se ambos se deslocam para o ponto de interseção de retas (próprio cliente e restantes utilizadores), ao qual chamamos de ponto de interseção / colisão. Neste caso foram utilizados 5 pontos por se considerar que existiria uma variação rápida no de-

clive desta reta quando o cliente curva para um dos lados. Caso apenas se utilizassem 3 pontos, as variações e os erros de leitura, embora pequenos, seriam suficientes para alterar bastante o declive da reta calculada. Ao utilizarmos os 5 pontos garante-se que um ponto fora do traço de deslocamento atual não terá tanto impacto no cálculo da reta de deslocamento, naquele instante.

Uma alternativa à utilização de 5 pontos pode passar por utilizar mais pontos, contudo sempre que se utiliza mais pontos o algoritmo demorará mais tempo a registar eventuais alterações de posição. Isto é particularmente relevante nos cenários de curvas que podem apresentar os maiores riscos, por exemplo curvas feitas a grandes velocidades com maior probabilidade de risco de colisão. Por outro lado, se se optasse por utilizar menos pontos aumentar-se-ia a dependência da precisão de leitura dos valores de GPS, pelo que é necessário algum cuidado na definição deste parâmetro.

Utilizou-se a regressão linear para que fosse possível mapear com menor erro o deslocamento de um veículo, no entanto esta poderá não ser a melhor solução. Seria possível utilizar regressões quadráticas, por exemplo, mas considera-se que esta solução apenas seria melhor para cenários de mudança de direção. Uma eventual solução seria a adaptação automática do algoritmo para que alterasse autonomamente a sua forma de funcionamento. Este poderia utilizar regressão linear sempre que deteta um movimento retilíneo por parte do cliente, e uma regressão quadrática sempre que deteta uma mudança de direção. Claro que esta hipótese adicionaria maior processamento no lado dos clientes.

Ainda assim, apesar da volatilidade dos mecanismos utilizados para calcular possíveis pontos de interseção, considera-se que o algoritmo funciona como esperado. Dados dois utilizadores que se deslocam num sentido que se interseta, e dado que o cálculo do ponto de interseção origina um aviso de distância de 3 metros, conclui-se que o resultado apresentado foi o correto para aquele momento. Note-se que caso um utilizador altere a sua direção, deixará de existir aviso de perigo de colisão. Desta forma, dado que os avisos podem ser relativamente voláteis estes apenas têm uma duração de 5 segundos. Isto significa que o cliente verá um aviso de perigo de colisão, se dentro de 5 segundos este aviso não for atualizado com uma distância inferior então considera-se que o perigo já não existe. Contudo se a plataforma atualizar este valor de distância de forma a que seja cada vez menor, considera-se que o cenário de perigo ainda é válido, ou pelo menos é válido por cerca de 5 segundos.

Foi possível observar que clientes que se encontram a uma distância superior a 500 metros não têm processamento adicional no cliente. O objeto *neighbours* serve para apresentar os clientes que se encontram a uma distância menor do que 500 metros, e notou-se na utilização do percurso a uma distância de 500 quilómetros da zona do teste, que não existe qualquer registo de utilizadores apesar desta informação estar a ser recebida no servidor, e retransmitida para o cliente. Note-se que o cliente apenas guarda as posições dos utilizado-

res que se encontram até 500 metros de distância, então ocorre um processamento, apesar de não ser apresentado nenhum. Note-se ainda que na solução final os clientes recebem no máximo distâncias de utilizadores que se encontrem a cerca de 50 quilómetros e não 500. Isto significa que o cliente apenas terá de calcular a distância de utilizadores nesta nova área, antes de efetuar qualquer processamento adicional. Com o aumento dos recursos no lado do servidor, seria possível diminuir ainda mais esta zona, contudo com os níveis atuais de utilização considera-se que será totalmente viável utilizar uma distância destas.

5.4 RESUMO

Este capitulo serviu para apresentar alguns dos testes que foram realizados. A ideia aqui patente passou por apresentar um cenário de risco real, e um cenário onde não deveria ser apresentado qualquer indicação de risco. Os resultados obtidos foram de encontro ao esperado, quando foi detetado um cenário de perigo o cliente foi devidamente notificado. Observa-se ainda que o cliente teve a capacidade não só de detetar o perigo iminente, como transmitir esta informação para o servidor, para que este a possa guardar. O segundo cenário não despoletou qualquer aviso de perigo, tal como seria esperado.

Apresentou-se inicialmente de um *setup* simples para a realização dos testes. Estes utilizaram a plataforma de simulação desenvolvida no subdomínio **API**. Após a realização dos testes faz-se uma pequena discussão relativa ao desempenho deste, bem como possíveis alternativas. Reflete-se ainda sobre os efeitos que alternativas de implementações diferentes da atual poderiam impor no funcionamento do algoritmo.

Finalmente conclui-se que o desempenho do serviço foi capaz não só de suportar o cliente e as suas alterações de movimentação, mas também de suportar um simulador que utilizou 10 veículos em simultâneo. O teste máximo que se realizou no simulador compreendeu cerca de 100 veículos, a partir deste valor o *browser* começa a ficar lento e não é capaz de enviar informações no tempo correto. Quando isto acontece deverá utilizar-se outra máquina para fazer mais testes, contudo considera-se que este facto não apresenta um problema pois a limitação está no lado do cliente e não do lado do servidor.

6

CONCLUSÃO

Neste capítulo final apresentam-se algumas conclusões relativas ao trabalho desenvolvido. Posteriormente debate-se sobre possibilidades alternativas e perspetivas de trabalho futuro.

6.1 CONCLUSÕES

Este trabalho tinha como objectivo a concepção e desenvolvimento de um sistema capaz de alertar utilizadores vulneráveis da via pública, para eventuais cenários de perigo. Inicialmente fez-se um levantamento de estudos, projectos ou produtos já realizados ou em curso, neste âmbito, no entanto notou-se que não existe nenhum sistema que faça exatamente o que era pretendido. Existem muitas alternativas, sejam elas utilizar equipamentos auxiliares (que muitas vezes custam várias centenas de euros), restringir sistemas de alerta a veículos mais recentes (que deixam uma grande parte dos utilizadores da via pública excluídos da solução) ou desenvolver soluções hipotéticas que eventualmente poderão ser passíveis de implementar.

Um dos focos deste projeto passou por desenvolver uma solução económica, que possa ser adotada de forma imediata, por um grande número de utilizadores. A plataforma desenvolvida fez exatamente isto que se pretendia, possibilitando que qualquer pessoa com um *smartphone*, *tablet*, ou computador portátil, possa ser um interveniente nesta troca de informações rodoviárias. Não só pode agir como condutor, como também poderá agir como peão / pedestre. Inicialmente pensou-se utilizar um equipamento extra, com custo reduzido, para obter mais informações relativamente ao veículo, contudo esta solução foi descartada dado que os dados obtidos não eram apresentados de forma consistente. Durante o desenvolvimento notou-se que existiam algumas dificuldades tecnológicas que iriam dificultar a chegada ao objetivo, contudo estas foram prontamente resolvidas, utilizando alternativas viáveis.

A complexidade deste projeto foi bastante elevada, não só durante a fase conceptual mas também durante a sua implementação. Notou-se a necessidade de implementar mecanismos que assegurem não só o bom desenvolvimento da plataforma (como documentação,

migrações para o sistema de dados, testes unitários, entre outros) como a sua fácil manutenção. Ferramentas de integração contínua foram essenciais para isto, e é algo que nunca havia sido utilizado anteriormente.

Ainda a nível de desenvolvimento, deparou-se com alguns problemas relativos à forma como se trataria da troca de informação. Debateu-se sobre a possibilidade de utilizar métodos *HTTP POST* mas concluiu-se que esta implementação limitaria a escalabilidade da solução, pelo que foi posta de parte em detrimento da comunicação por *sockets* que acabou por ser adotada. Existiram dúvidas relativamente à localização do processamento de informação. Ao contrário do implementado, em que se foca este esforço no cliente, o processamento poderia ser feito no servidor, contudo dado os recursos do servidor na solução atual não faria sentido manter tal filosofia de implementação. Noutro contexto, com mais recursos a nível de servidor, pode ser explorada esta hipótese a detalhe. Foi feita a decisão de implementação de pedidos assíncronos ao sistema de dados, assim como na resposta ao cliente, para que toda a plataforma conseguisse dar resposta a vários pedidos simultâneos.

A principal decisão de filtragem passa por limitar a utilização da plataforma a grelhas de latitude e longitude. Esta solução poderá não fazer sentido atualmente, dado que com o número atual de utilizadores registados poder-se-ia não fazer qualquer verificação e aceitar comunicações provenientes de todo o planeta. Contudo esta solução pressupõe uma utilização massiva, pelo que é necessário implementar mecanismos que controlem o acesso à troca de informação com o servidor.

Todas estas decisões de implementação regem-se não só pelo ambiente de desenvolvimento como pelo contexto atual que se vive, pelo que se este projeto tivesse sido realizado há alguns anos, talvez não fizesse sequer sentido depender do *smartphone* dos utilizadores de tal forma como foi aqui feito.

Finalmente, considera-se que todos os objetivos foram cumpridos na totalidade visto que existe uma plataforma *online* que está totalmente funcional e pronta a ser testada. Esta reage da forma esperada perante cenários de perigo, e alerta os seus utilizadores para que estes consigam reagir atempadamente.

6.2 PERSPETIVAS DE TRABALHO FUTURO

Apesar de se considerar que o desenvolvimento da solução proposta foi bem sucedido, acredita-se que a solução pode evoluir e melhorar.

O trabalho futuro que se antevê para este projeto passaria por melhorar a interface com o utilizador, de forma a conseguir transmitir o máximo de informação possível, de forma clara. Um dos aspetos fundamentais do projeto, o algoritmo, pode vir a ser melhorado adicionando mais verificações antes de gerar um alerta aos clientes. Esta abordagem pode tornar o algoritmo demasiado rígido, não avisando quando um potencial cenário de risco

poderá acontecer, no entanto também poderá melhorar a sua eficácia. Seria necessário fazer ainda mais testes e tentar perceber se realmente compensa aumentar o processamento a ser feito no cliente.

Outra melhoria possível, seria relativa à maneira como a filtragem de informação é feita no servidor. Atualmente considera-se que a plataforma apenas pode ser utilizada numa certa zona do país mas se existissem mais nós computacionais equivalentes ao servidor atual, seria possível atribuir uma certa zona a cada um desses nós, e posteriormente criar um nodo gestor que distribuiria a informação de forma hierárquica. Esta solução também poderá introduzir tempos de latência adicionais na troca de informação.

Adicionalmente poderia ser explorada a possibilidade de integrar a plataforma numa aplicação nativa a sistemas *Android* ou *iOS*. A plataforma poderá vir a ser utilizada em veículos mais atuais, que já contém computadores de bordo com estes sistemas. Seria interessante explorar um mundo onde todos os veículos têm capacidade de comunicar com recurso a LTE a sua posição GPS. Não só entre veículos, mas para qualquer outro tipo de dispositivo!

BIBLIOGRAFIA

José Javier Anaya, Pierre Merdrignac, Oyunchimeg Shagdar, Fawzi Nashashibi, and José E Naranjo. Vehicle to pedestrian communications for protection of vulnerable road users. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 1037–1042. IEEE, 2014.

Mehrdad Bagheri, Matti Siekkinen, and Jukka K Nurminen. Cellular-based vehicle to pedestrian (v2p) adaptive communication for collision avoidance. In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pages 450–456. IEEE, 2014.

Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. Device-to-device communications with wi-fi direct: overview and experimentation. *IEEE wireless communications*, 20(3):96–104, 2013.

Odongo Steven Eyobu, Jhihoon Joo, and Dong Seog Han. A broadcast scheme for vehicle-to-pedestrian safety message dissemination. *International Journal of Distributed Sensor Networks*, 13(11), 2017.

Carlos Flores, Pierre Merdrignac, Raoul de Charette, Francisco Navas, Vicente Milanés, and Fawzi Nashashibi. A cooperative car-following/emergency braking system with prediction-based pedestrian avoidance capabilities. *IEEE Transactions on Intelligent Transportation Systems*, 1(99):1–10, 2018.

Tarak Gandhi and Mohan Manubhai Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on intelligent Transportation systems*, 8(3):413–430, 2007.

Ahmed Hussein, Fernando García, José María Armingol, and Cristina Olaverri-Monreal. P2v and v2p communication for pedestrian warning on the basis of autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 2034–2039. IEEE, 2016.

Janis Jansons, Nikolajs Bogdanovs, and Aleksandrs Ipatovs. Vehicle-to-infrastructure communication based on ieee 802.11 g. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 2(1):46–50, 2012.

Sungwon Lee and Dongkyun Kim. An energy efficient vehicle to pedestrian communication method for safety applications. *Wireless Personal Communications*, 86(4):1845–1856, 2016.

Pierre Merdrignac, Oyunchimeg Shagdar, and Fawzi Nashashibi. Fusion of perception and v2p communication systems for the safety of vulnerable road users. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1740–1751, 2017.

Johan Scholliers, Marcel Van Sambeek, and Kees Moerman. Integration of vulnerable road users in cooperative its systems. *European transport research review*, 9(2):15, 2017.

Hanbyul Seo, Ki-Dong Lee, Shinpei Yasukawa, Ying Peng, and Philippe Sartori. Lte evolution for vehicle-to-everything services. *IEEE communications magazine*, 54(6):22–28, 2016.

Markus Ullmann, Christian Wieschebrink, Thomas Strubbe, and Dennis Kuegler. Secure vehicle-to-infrastructure communication: Secure roadside stations, key management, and crypto agility. *International Journal on Advances in Security Volume 9, Number 1 & 2, 2016*, 2016.

World Health Organization. Violence, Injury Prevention, and World Health Organization. *Global status report on road safety 2013: supporting a decade of action*. World Health Organization, 2013.

A

TEMPLATES DE FRONTEND

A.1 TEMPLATE DE REGISTO PARA NOVO UTILIZADOR

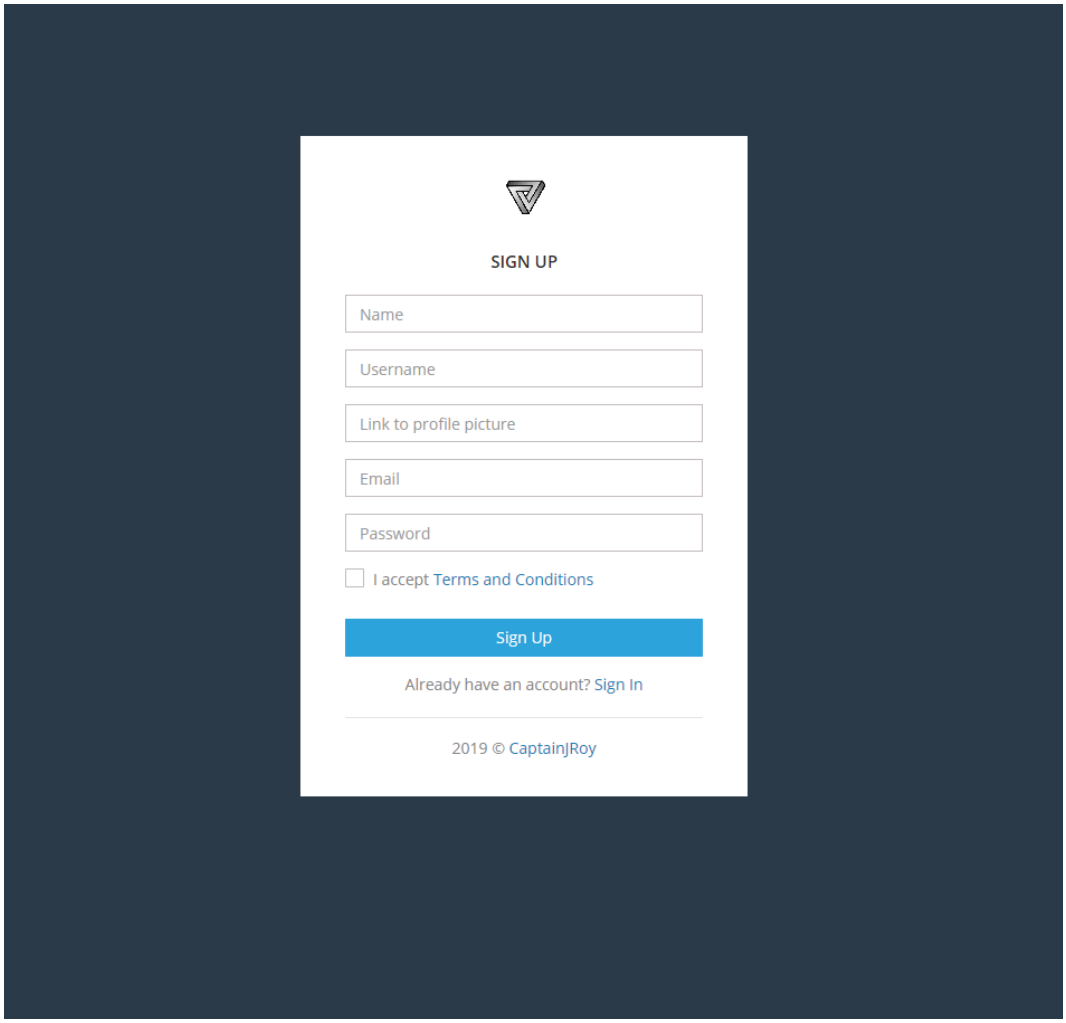


Figura 50.: Exemplo da interface *web* para registo de um novo utilizador no sistema (acedido em junho 2019)

A.2 TEMPLATE DE LOGIN PARA UTILIZADORES EXISTENTES

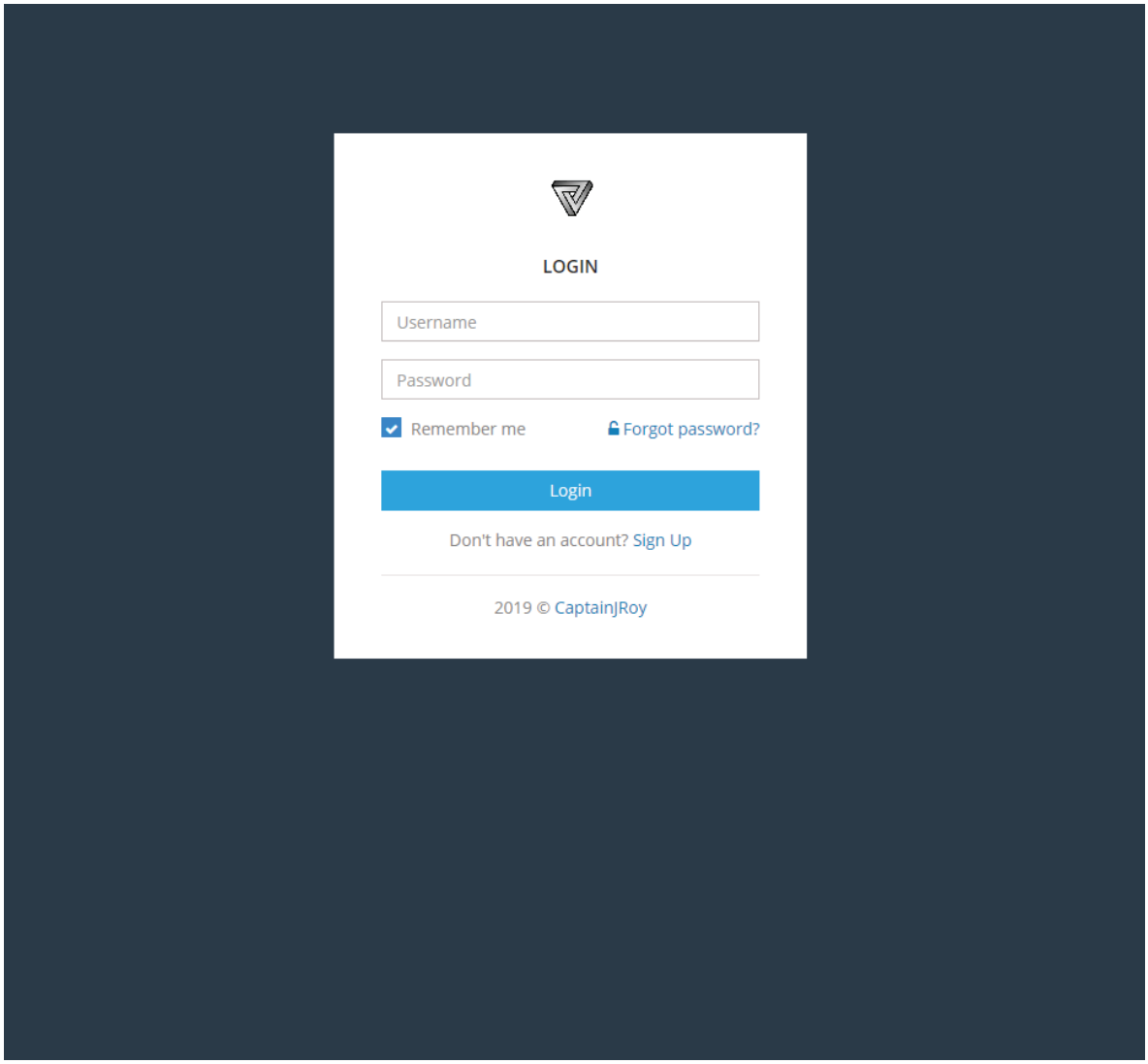


Figura 51.: Exemplo da interface *web* para iniciar sessão de utilizador já existente no sistema (acedido em junho 2019)

A.3 Template do perfil de utilizador

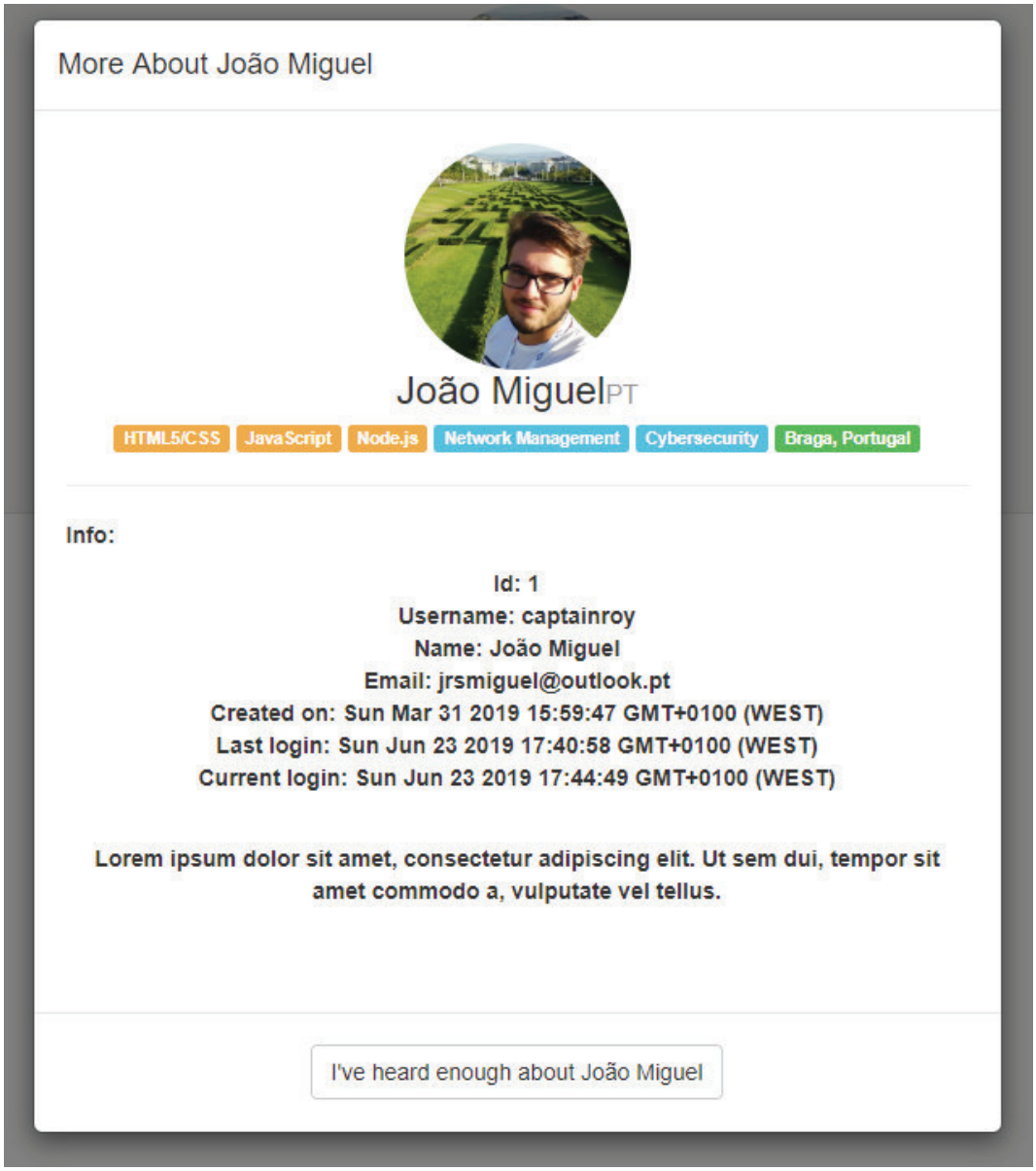


Figura 52.: Exemplo da interface *web* para visualização do perfil de utilizador (acedido em junho de 2019)

A.4 TEMPLATE DO MENU LAUNCH DA WEBAPP DESENVOLVIDA

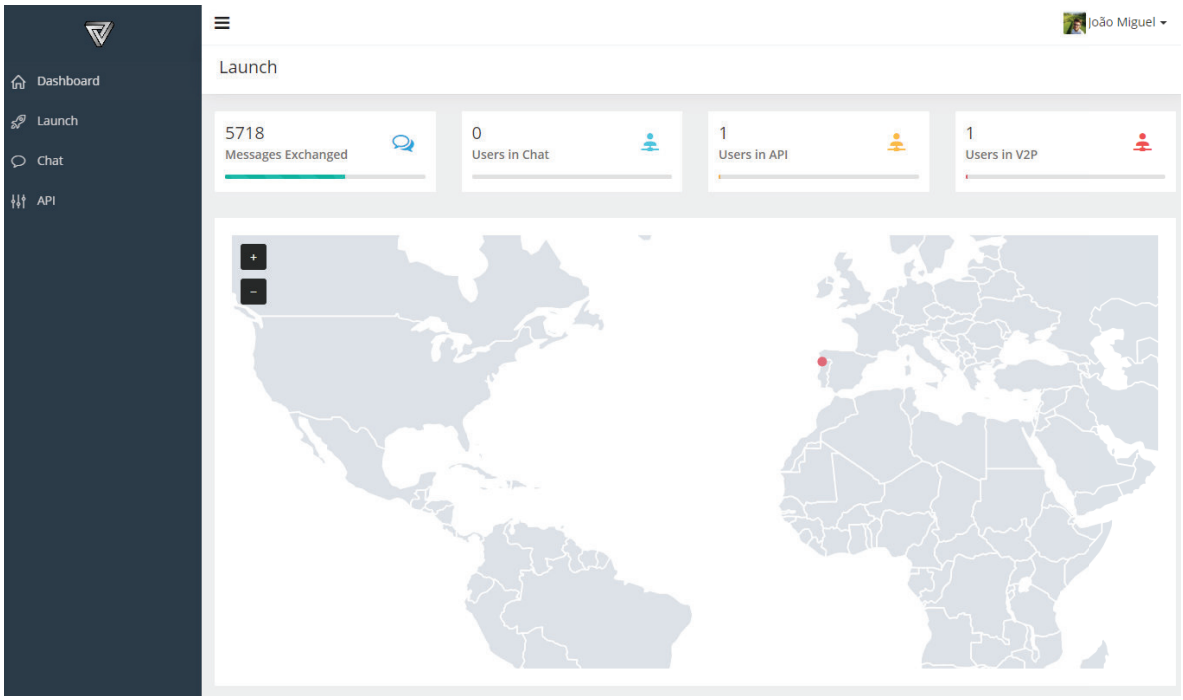


Figura 53.: Exemplo da interface *web* para utilização do sistema desenvolvido (acedido em junho de 2019)

A.5 TEMPLATE DA PLATAFORMA CHAT DESENVOLVIDA

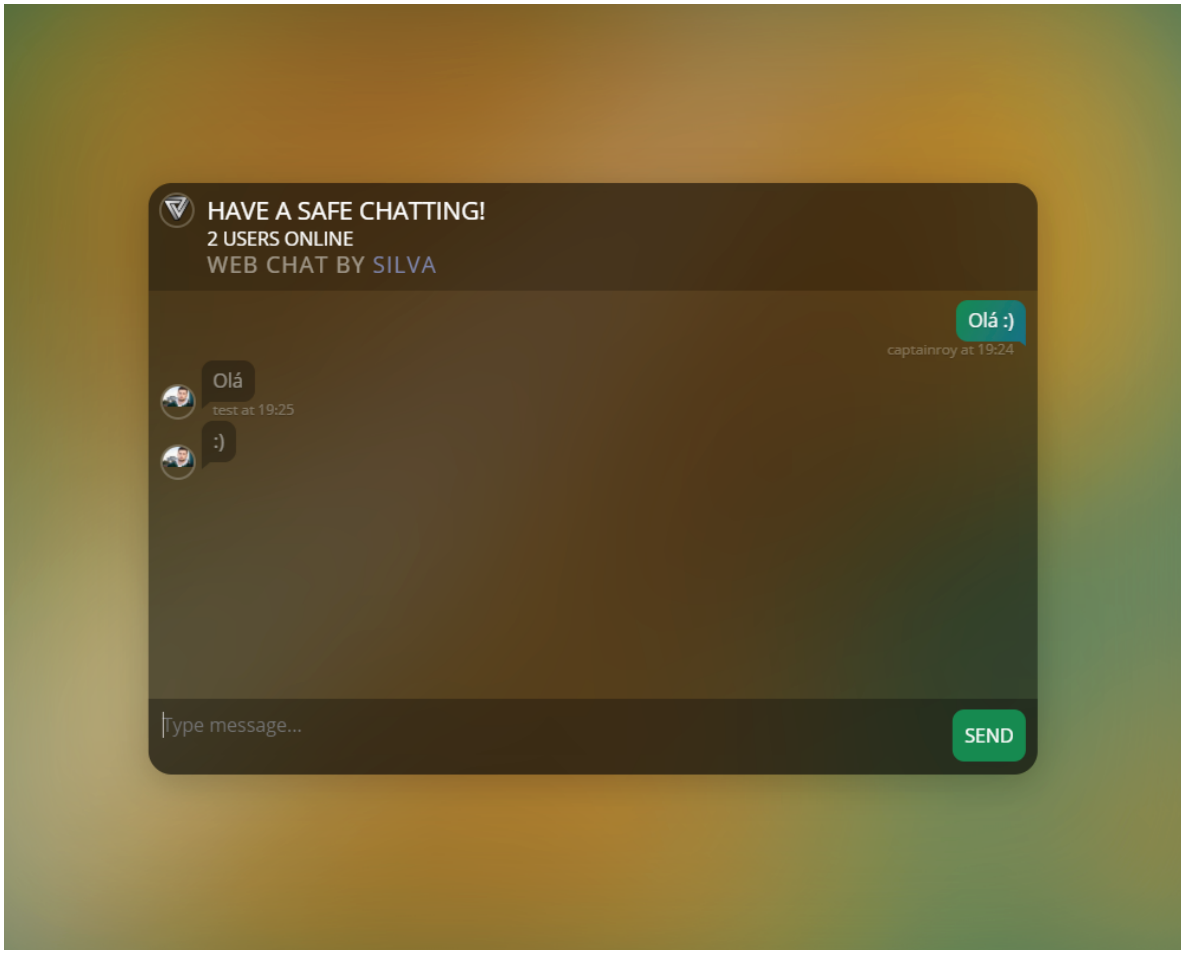


Figura 54.: Exemplo da interface *web* para utilização do chat utilizado para testes (acedido em junho de 2019)

A.6 TEMPLATE DA API INICIALMENTE DESENVOLVIDA

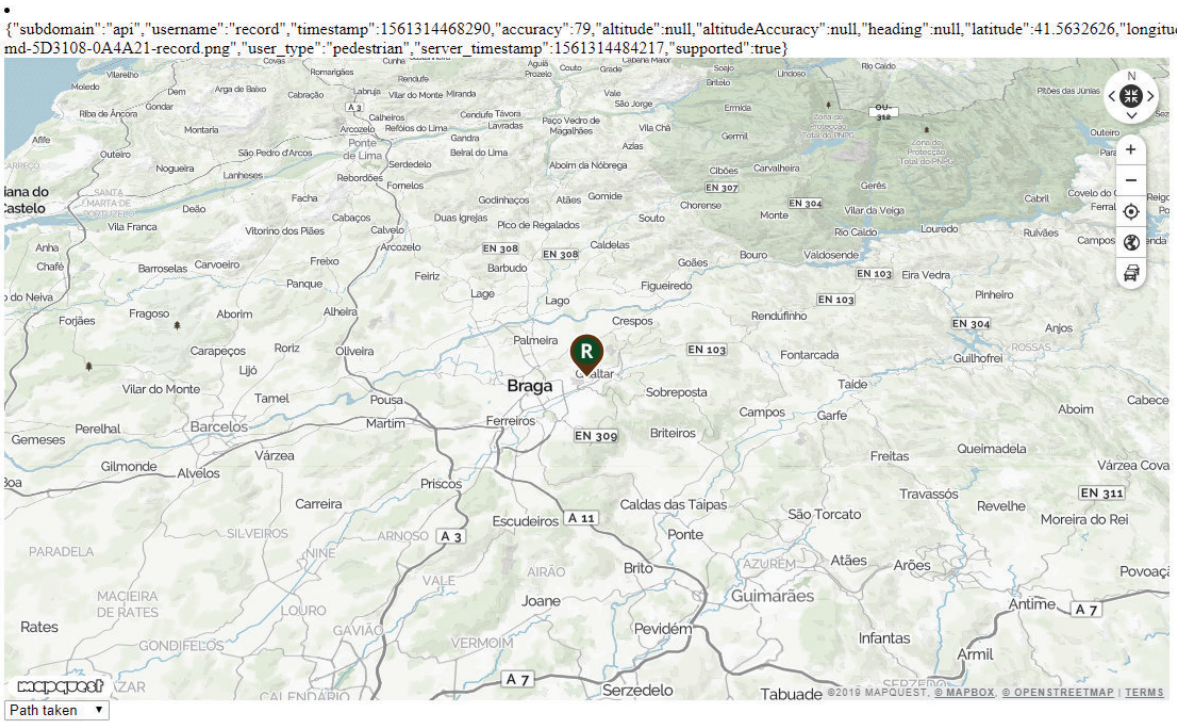


Figura 55.: Exemplo da interface *web* para utilização da API inicial utilizada para testes (acedido em junho de 2019)

A.7 TEMPLATE DA DOCUMENTAÇÃO GERADA PELO PLUGIN JSDOC

Figura 56.: Exemplo da interface *web* para visualização da documentação do código do projeto

A.8 Template da dashboard desenvolvida para o simulador

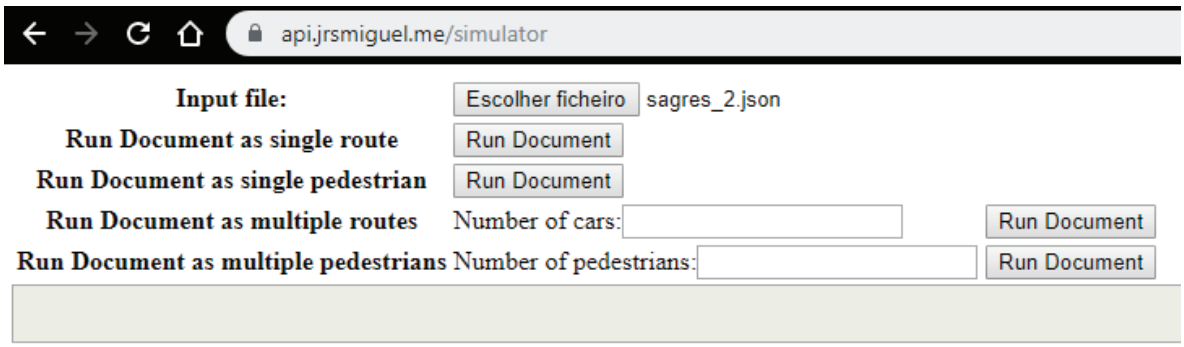


Figura 57.: Exemplo da interface *web* para execução do script de simulação de rotas

B

DADOS DE FRONTEND

B.1 DADOS RELATIVOS A TODOS OS UTILIZADORES REGISTADOS NO SISTEMA


```
[{"id":1,"username":"captainroy"},{"id":2,"username":"silva"},{"id":3,"username":"mastergomes"},{"id":4,"username":"test"},
{"id":7,"username":"ss"},{"id":8,"username":"fnac_braga"},{"id":9,"username":"hyfeez"},{"id":10,"username":"record"},
{"id":16,"username":"aldcosta"},{"id":17,"username":"carlitos"}]
```


Figura 58.: Exemplo de dados em formato *JSON* retornados quando invocação de API para múltiplos utilizadores (acedido em junho de 2019)

B.2 DADOS RELATIVOS A UM ÚNICO UTILIZADOR REGISTADO NO SISTEMA


```
{"id":1,"profile_picture":"https://captainjroy.github.io/images/foto_perfil.jpg","name":"João
Miguel","username":"captainroy","email":"jrsmiguel@outlook.pt","created_on":"2019-03-31T14:59:47.942Z","last_login":"2019-06-
23T16:40:58.088Z","current_login":"2019-06-23T16:44:49.886Z"}
```


Figura 59.: Exemplo de dados em formato *JSON* retornados quando invocação de API para um único utilizador (acedido em junho de 2019)

B.3 DADOS RELATIVOS A TODOS OS AVISOS EMITIDOS NO SISTEMA


```
{"id":615,"level":1,"heading_towards":null,"distance":141,"detected_by":"captainroy","other_user":"13"},{"id":616,"level":1,"heading_towards":null,"distance":104,"detected_by":"captainroy","other_user":"17"},
{"id":617,"level":1,"heading_towards":null,"distance":58,"detected_by":"captainroy","other_user":"19"},{"id":618,"level":1,"heading_towards":null,"distance":259,"detected_by":"captainroy","other_user":"2"},
{"id":619,"level":1,"heading_towards":null,"distance":253,"detected_by":"captainroy","other_user":"6"},{"id":620,"level":1,"heading_towards":null,"distance":259,"detected_by":"captainroy","other_user":"11"},
{"id":621,"level":1,"heading_towards":null,"distance":139,"detected_by":"captainroy","other_user":"13"},{"id":622,"level":1,"heading_towards":null,"distance":145,"detected_by":"captainroy","other_user":"17"},
{"id":623,"level":1,"heading_towards":null,"distance":76,"detected_by":"captainroy","other_user":"19"},{"id":624,"level":1,"heading_towards":null,"distance":307,"detected_by":"captainroy","other_user":"2"},
{"id":625,"level":1,"heading_towards":null,"distance":257,"detected_by":"captainroy","other_user":"6"},{"id":626,"level":1,"heading_towards":null,"distance":248,"detected_by":"captainroy","other_user":"11"},
{"id":627,"level":1,"heading_towards":null,"distance":139,"detected_by":"captainroy","other_user":"13"},{"id":628,"level":1,"heading_towards":null,"distance":172,"detected_by":"captainroy","other_user":"17"},
{"id":629,"level":1,"heading_towards":null,"distance":100,"detected_by":"captainroy","other_user":"19"},{"id":630,"level":1,"heading_towards":null,"distance":298,"detected_by":"captainroy","other_user":"2"},
{"id":631,"level":1,"heading_towards":null,"distance":268,"detected_by":"captainroy","other_user":"6"},{"id":632,"level":1,"heading_towards":null,"distance":241,"detected_by":"captainroy","other_user":"11"},
{"id":633,"level":1,"heading_towards":null,"distance":138,"detected_by":"captainroy","other_user":"13"},{"id":634,"level":1,"heading_towards":null,"distance":190,"detected_by":"captainroy","other_user":"17"},
{"id":635,"level":1,"heading_towards":null,"distance":239,"detected_by":"captainroy","other_user":"19"},{"id":636,"level":1,"heading_towards":null,"distance":296,"detected_by":"captainroy","other_user":"2"},
{"id":637,"level":1,"heading_towards":null,"distance":277,"detected_by":"captainroy","other_user":"6"},{"id":638,"level":1,"heading_towards":null,"distance":235,"detected_by":"captainroy","other_user":"11"},
{"id":639,"level":1,"heading_towards":null,"distance":138,"detected_by":"captainroy","other_user":"13"},{"id":640,"level":1,"heading_towards":null,"distance":189,"detected_by":"captainroy","other_user":"17"},
{"id":641,"level":1,"heading_towards":null,"distance":377,"detected_by":"captainroy","other_user":"19"},{"id":642,"level":1,"heading_towards":null,"distance":295,"detected_by":"captainroy","other_user":"2"},
{"id":643,"level":1,"heading_towards":null,"distance":281,"detected_by":"captainroy","other_user":"6"},{"id":644,"level":1,"heading_towards":null,"distance":228,"detected_by":"captainroy","other_user":"11"},
{"id":645,"level":1,"heading_towards":null,"distance":170,"detected_by":"captainroy","other_user":"17"},{"id":646,"level":1,"heading_towards":null,"distance":263,"detected_by":"captainroy","other_user":"18"},
{"id":647,"level":1,"heading_towards":null,"distance":294,"detected_by":"captainroy","other_user":"2"},{"id":648,"level":1,"heading_towards":null,"distance":280,"detected_by":"captainroy","other_user":"6"},
{"id":649,"level":1,"heading_towards":null,"distance":446,"detected_by":"test","other_user":"0"},{"id":650,"level":2,"heading_towards":true,"distance":446,"detected_by":"test","other_user":"0"},
{"id":651,"level":2,"heading_towards":true,"distance":498,"detected_by":"test","other_user":"2"},{"id":652,"level":1,"heading_towards":null,"distance":249,"detected_by":"test","other_user":"4"},
{"id":653,"level":1,"heading_towards":null,"distance":498,"detected_by":"test","other_user":"2"},{"id":654,"level":2,"heading_towards":true,"distance":249,"detected_by":"test","other_user":"4"},
{"id":655,"level":1,"heading_towards":null,"distance":303,"detected_by":"test","other_user":"5"},{"id":656,"level":2,"heading_towards":true,"distance":303,"detected_by":"test","other_user":"5"},
{"id":657,"level":1,"heading_towards":null,"distance":240,"detected_by":"test","other_user":"10"},{"id":658,"level":1,"heading_towards":null,"distance":322,"detected_by":"test","other_user":"11"},
{"id":659,"level":1,"heading_towards":null,"distance":251,"detected_by":"test","other_user":"16"},{"id":660,"level":1,"heading_towards":null,"distance":262,"detected_by":"test","other_user":"18"},
{"id":661,"level":1,"heading_towards":null,"distance":394,"detected_by":"test","other_user":"20"},{"id":662,"level":1,"heading_towards":null,"distance":221,"detected_by":"test","other_user":"22"},
{"id":663,"level":1,"heading_towards":null,"distance":90,"detected_by":"test","other_user":"23"},{"id":664,"level":1,"heading_towards":null,"distance":0,"detected_by":"test","other_user":"26"},
{"id":665,"level":1,"heading_towards":null,"distance":290,"detected_by":"test","other_user":"27"},{"id":666,"level":2,"heading_towards":true,"distance":290,"detected_by":"test","other_user":"27"},
{"id":667,"level":2,"heading_towards":true,"distance":0,"detected_by":"test","other_user":"26"},{"id":668,"level":1,"heading_towards":null,"distance":301,"detected_by":"test","other_user":"28"},
{"id":669,"level":1,"heading_towards":null,"distance":0,"detected_by":"test","other_user":"30"},{"id":670,"level":2,"heading_towards":true,"distance":0,"detected_by":"test","other_user":"30"},
{"id":671,"level":1,"heading_towards":null,"distance":120,"detected_by":"test","other_user":"32"},{"id":672,"level":1,"heading_towards":null,"distance":323,"detected_by":"test","other_user":"33"},
{"id":673,"level":1,"heading_towards":null,"distance":301,"detected_by":"test","other_user":"39"},{"id":674,"level":2,"heading_towards":true,"distance":301,"detected_by":"test","other_user":"39"},
{"id":675,"level":1,"heading_towards":null,"distance":485,"detected_by":"test","other_user":"0"},{"id":676,"level":2,"heading_towards":true,"distance":485,"detected_by":"test","other_user":"0"},
{"id":677,"level":1,"heading_towards":null,"distance":245,"detected_by":"test","other_user":"4"},{"id":678,"level":2,"heading_towards":true,"distance":245,"detected_by":"test","other_user":"4"},
```


Figura 60.: Exemplo de dados em formato *JSON* retornados quando invocação de API para todos os avisos (acedido em julho de 2019)

B.4 DADOS RELATIVOS A AVISOS DE UM ÚNICO UTILIZADOR DO SISTEMA

{"id":786,"level":2,"heading_towards":true,"distance":294,"detected_by":"test","other_user":"39"},{"id":789,"level":1,"heading_towards":null,"distance":258,"detected_by":"test","other_user":"2"},
{"id":790,"level":2,"heading_towards":true,"distance":258,"detected_by":"test","other_user":"2"},{"id":792,"level":2,"heading_towards":true,"distance":247,"detected_by":"test","other_user":"4"},
{"id":793,"level":1,"heading_towards":null,"distance":295,"detected_by":"test","other_user":"5"},{"id":825,"level":1,"heading_towards":null,"distance":272,"detected_by":"test","other_user":"17"},
{"id":829,"level":1,"heading_towards":null,"distance":371,"detected_by":"test","other_user":"23"},{"id":830,"level":1,"heading_towards":null,"distance":41,"detected_by":"test","other_user":"26"},
{"id":832,"level":1,"heading_towards":null,"distance":297,"detected_by":"test","other_user":"27"},{"id":834,"level":1,"heading_towards":null,"distance":134,"detected_by":"test","other_user":"28"},
{"id":836,"level":2,"heading_towards":true,"distance":20,"detected_by":"test","other_user":"30"},{"id":837,"level":1,"heading_towards":null,"distance":366,"detected_by":"test","other_user":"33"},
{"id":841,"level":2,"heading_towards":true,"distance":285,"detected_by":"test","other_user":"39"},{"id":842,"level":1,"heading_towards":null,"distance":148,"detected_by":"test","other_user":"2"},
{"id":843,"level":2,"heading_towards":true,"distance":148,"detected_by":"test","other_user":"2"},{"id":845,"level":2,"heading_towards":true,"distance":246,"detected_by":"test","other_user":"4"},
{"id":847,"level":2,"heading_towards":true,"distance":285,"detected_by":"test","other_user":"5"},{"id":849,"level":1,"heading_towards":null,"distance":266,"detected_by":"test","other_user":"10"},
{"id":851,"level":1,"heading_towards":null,"distance":271,"detected_by":"test","other_user":"16"},{"id":852,"level":1,"heading_towards":null,"distance":228,"detected_by":"test","other_user":"17"},
{"id":853,"level":1,"heading_towards":null,"distance":265,"detected_by":"test","other_user":"18"},{"id":854,"level":1,"heading_towards":null,"distance":426,"detected_by":"test","other_user":"20"},
{"id":855,"level":1,"heading_towards":null,"distance":75,"detected_by":"test","other_user":"22"},{"id":856,"level":1,"heading_towards":null,"distance":47,"detected_by":"test","other_user":"26"},
{"id":858,"level":1,"heading_towards":null,"distance":292,"detected_by":"test","other_user":"27"},{"id":860,"level":1,"heading_towards":null,"distance":137,"detected_by":"test","other_user":"28"},
{"id":861,"level":1,"heading_towards":null,"distance":22,"detected_by":"test","other_user":"30"},{"id":868,"level":2,"heading_towards":true,"distance":283,"detected_by":"test","other_user":"39"},
{"id":869,"level":1,"heading_towards":null,"distance":130,"detected_by":"test","other_user":"2"},{"id":872,"level":2,"heading_towards":true,"distance":245,"detected_by":"test","other_user":"4"},
{"id":873,"level":1,"heading_towards":null,"distance":286,"detected_by":"test","other_user":"5"},{"id":875,"level":1,"heading_towards":null,"distance":377,"detected_by":"test","other_user":"8"},

Figura 61.: Exemplo de dados em formato *JSON* retornados quando invocação de API para um utilizador específico (acedido em julho de 2019)

B.5 DADOS RELATIVOS A AVISOS DE NÍVEL PRÉ ESTABELECIDO

[{"id":896,"level":3,"heading_towards":true,"distance":141,"detected_by":"captainroy","other_user":"0"},{"id":899,"level":3,"heading_towards":true,"distance":109,"detected_by":"captainroy","other_user":"1"},
{"id":906,"level":3,"heading_towards":true,"distance":156,"detected_by":"captainroy","other_user":"18"},{"id":909,"level":3,"heading_towards":true,"distance":271,"detected_by":"captainroy","other_user":"20"},
{"id":914,"level":3,"heading_towards":true,"distance":291,"detected_by":"captainroy","other_user":"24"},{"id":918,"level":3,"heading_towards":true,"distance":261,"detected_by":"captainroy","other_user":"26"},
{"id":921,"level":3,"heading_towards":true,"distance":16,"detected_by":"captainroy","other_user":"28"},{"id":929,"level":3,"heading_towards":true,"distance":138,"detected_by":"captainroy","other_user":"38"},
{"id":932,"level":3,"heading_towards":true,"distance":73,"detected_by":"captainroy","other_user":"39"},{"id":939,"level":3,"heading_towards":true,"distance":142,"detected_by":"captainroy","other_user":"0"},
{"id":947,"level":3,"heading_towards":true,"distance":145,"detected_by":"captainroy","other_user":"18"},{"id":950,"level":3,"heading_towards":true,"distance":264,"detected_by":"captainroy","other_user":"20"},
{"id":953,"level":3,"heading_towards":true,"distance":292,"detected_by":"captainroy","other_user":"24"},{"id":957,"level":3,"heading_towards":true,"distance":263,"detected_by":"captainroy","other_user":"26"},
{"id":960,"level":3,"heading_towards":true,"distance":23,"detected_by":"captainroy","other_user":"28"},{"id":968,"level":3,"heading_towards":true,"distance":138,"detected_by":"captainroy","other_user":"38"},
{"id":971,"level":3,"heading_towards":true,"distance":66,"detected_by":"captainroy","other_user":"39"},{"id":985,"level":3,"heading_towards":true,"distance":154,"detected_by":"captainroy","other_user":"18"},
{"id":988,"level":3,"heading_towards":true,"distance":273,"detected_by":"captainroy","other_user":"20"},{"id":993,"level":3,"heading_towards":true,"distance":290,"detected_by":"captainroy","other_user":"24"},
{"id":996,"level":3,"heading_towards":true,"distance":265,"detected_by":"captainroy","other_user":"26"},{"id":1001,"level":3,"heading_towards":true,"distance":28,"detected_by":"captainroy","other_user":"28"},
{"id":1009,"level":3,"heading_towards":true,"distance":138,"detected_by":"captainroy","other_user":"38"},{"id":1012,"level":3,"heading_towards":true,"distance":65,"detected_by":"captainroy","other_user":"39"}]

Figura 62.: Exemplo de dados em formato *JSON* retornados quando invocação de API para avisos de nível 3 (acedido em julho de 2019)

C

CÓDIGO FONTE

C.1 CÓDIGO RELATIVO A UMA MIGRAÇÃO

```
exports.up = (pgm) => {
  pgm.createTable("users", {
    id: 'id',
    profile_picture: {
      type    : "varchar(512)",
      notNull: true
    },
    name: {
      type    : "varchar(256)",
      notNull: true
    },
    username: {
      type    : "varchar(256)",
      notNull: true,
      unique : true
    },
    email: {
      type    : "varchar(256)",
      notNull: true,
      unique : true
    },
    password: {
      type    : "varchar(256)",
      notNull: true,
    },
    created_on: {
      type: "timestamp",
      notNull: true,
      default: pgm.func("current_timestamp")
    },
    last_login: {
      type: "timestamp",
      notNull: false
    },
    current_login: {
      type: "timestamp",
      notNull: false
    }
  })
}
```

Figura 63.: Exemplo de uma migração utilizando pg-migrate

C.2 CÓDIGO RELATIVO À API DE WEB BLUETOOTH

```
//Request the browser for selection of any device following filters
navigator.bluetooth.requestDevice({
    acceptAllDevices: true,
    optionalServices: [0x1800, 0x1811, 0x1815, 0x180F, 0x1810, 0x181B, 0x181E, 0x181F, 0x1805, ...]
})
    //Request connection for the selected device
        .then( device => device.gatt.connect() )
        //Get services available for selected device
        .then( server => server.getPrimaryServices() )
        //Read and print characteristics of the device
        .then( services =>
            services.forEach( service =>
                        service.getCharacteristics()
                            .then( charact => console.log(caract.readValue()) )
        )
        )
        //Handle eventual error that might occur
        .catch( e => console.error(e) )
```

Figura 64.: Exemplo de código utilizado na *API* de *web-bluetooth*

C.3 FORMATO DE COMENTÁRIO UTILIZADO PELO PLUGIN JSDOC

```
/**
*   Save a new warning emited by a user on the PostgreSQL database running on RaspberryPi3B+,
* if possible. If no warning is created, alerts browser
* @func
* @name saveWarning
* @param {Integer} level - The warning level
* @param {Boolean} heading_towards - The users heads towards collision
* @param {Integer} distance - The users distance
* @param {String} detected_by - The user whom detected
* @param {String} other_user - The user whom was detected
* @return The ID of the created warning
*/
const saveWarning = ({ level, heading_towards, distance, detected_by, other_user }) => {
  return db.any('INSERT INTO warnings (level, heading_towards, distance, detected_by, other_use
    .then( data => data )
    .catch( err => null )
}
```

Figura 65.: Exemplo de código utilizado para gerar *templates HTML* com especificações de métodos

C.4 API DISPONIBILIZADA PELO BROWSER PARA GEOLOCALIZAÇÃO

```
function start() {
  if (navigator.geolocation) {
    initializeColors()
    if (selected == 'blank') alert('Please select display route mode')
    navigator.geolocation.watchPosition(preparePosition, showError, {enableHighAccuracy: true})
  }
  else alert("Geolocation is not supported by this browser.")
}
```

Figura 66.: Exemplo de código utilizado para efetuar leitura de valores de localização de clientes (Mozilla, julho 2019)

C.5 EXEMPLO DE UTILIZAÇÃO DE TESTES UNITÁRIOS

```
describe('queries', () => {
  const db_conn = require('./connection.js')('TEST')
  const db      = require('./queries.js')(db_conn)


  it('should have queries available', async () => {
    const res = await db.queries

    expect(res).not.toBeNull()
    expect(res).toBeDefined()
  })


  it('should create new user', async () => {
    const res = await db.queries.createUser({
      name: 'test_user_name',
      username: 'test_user_username',
      profile_picture: 'test_user_picture',
      email: 'test_user_email',
      password: 'test_user_password'
    })

    expect(res).toBeDefined()
    expect(res).not.toBeNull()
  })
```

Figura 67.: Exemplo de código utilizado para executar testes unitários através de biblioteca Jest

C.6 EXEMPLO DE CONFIGURAÇÃO DO CIRCLECI

```
version: 2
jobs:
  build:
    working_directory: ~/project/Webserver/https/ # directory where steps will run

    docker:
      # CircleCI maintains a library of pre-built images
      # documented at https://circleci.com/docs/2.0/circleci-images/
      - image: circleci/node:9.11 # ...with this image as the primary container; this is where all `steps` will run
        environment:
          PGHOST: 127.0.0.1
          PGUSER: server
          RUN_MODE: DEVELOPMENT
      - image: circleci/postgres:9.6.11
        environment:
          POSTGRES_USER:
          POSTGRES_PASSWORD:
          POSTGRES_DB: postgres

    steps:
      - checkout

      - run:
          name: update-npm
          command: |
            cd Webserver/https/
            sudo npm install -g npm@latest

      - restore_cache:
          key: dependency-cache-{{ checksum "Webserver/https/package.json" }}

      - run:
          name: Installing node packages
          command: |
            cd Webserver/https/
            npm install -d

      - save_cache:
          key: dependency-cache-{{ checksum "Webserver/https/package.json" }}
```

Figura 68.: Exemplo de código de configuração utilizada para executar testes de integração contínua

C.7 EXCERTO DE SCRIPT DE ADAPTAÇÃO DE CÓDIGO PARA SIMULADOR

```
def readLines():
    file = open(sys.argv[1], "r+")
    lines = file.readlines()
    json = {}
    csv = []
    csv.append('elevation,latitude,longitude')
    last_timestamp = JSON.loads(lines[0])['timestamp']
    index = 0

    for line in lines:
        line = JSON.loads(line)
        line_values = {
            'time_delta': line['timestamp'] - last_timestamp,
            'accuracy':   line['accuracy'],
            'altitude':   line['altitude'],
            'altitudeAccuracy': line['altitudeAccuracy'],
            'heading':    line['heading'],
            'latitude':   line['latitude'],
            'longitude':  line['longitude'],
            'speed':      line['speed']
        }
        last_timestamp = line['timestamp']
        csv.append(str(line['altitude']) + ',' + str(line['latitude']) + ',' + str(line['longitude']))

        json.update({index: line_values})
        index += 1

    file.close()
    return json, csv



def writeContent(json, csv):
    file = open('output_data/' + sys.argv[2] + '.json', "w+")
    file.write(JSON.dumps(json))
    file.close()

    file = open('route_csv/' + sys.argv[2] + '.csv', "w+")
    file.write('\n'.join(csv))
    file.close()
```

Figura 69.: Exemplo de código de código utilizado para gerar informação consumida pelo simulador

D

FERRAMENTAS ADICIONAIS

D.1 GESTÃO AUTOMATICA DE UPTIME

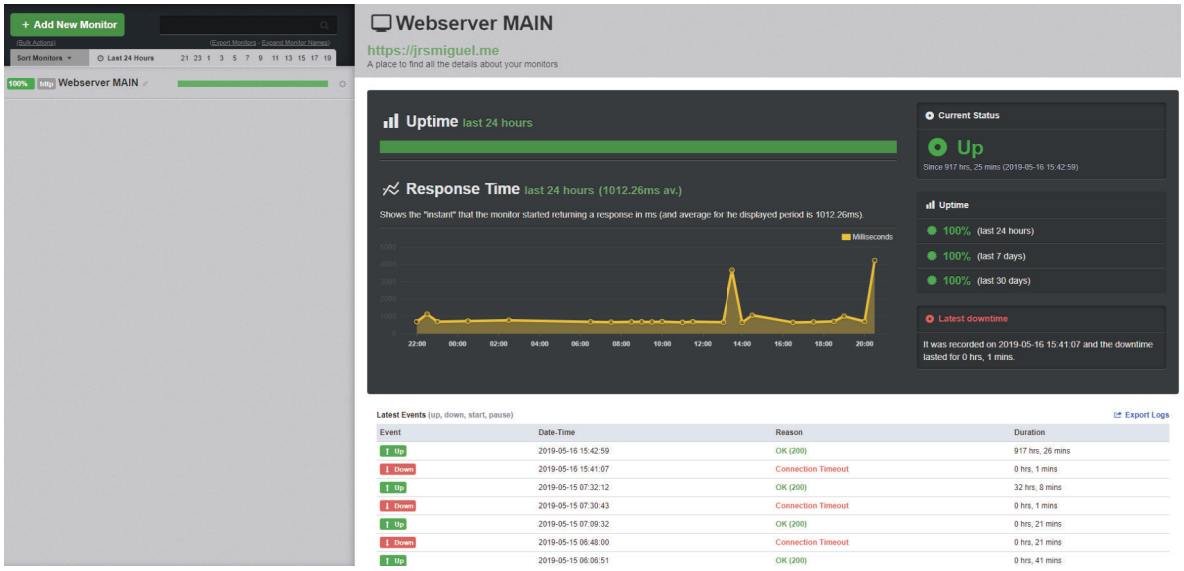


Figura 70.: Dashboard do sistema de gestão automática de uptime do UptimeRobot (acedido em junho de 2019)

D.2 DASHBOARD DE GESTÃO DE ENDEREÇAMENTO DO SERVIDOR

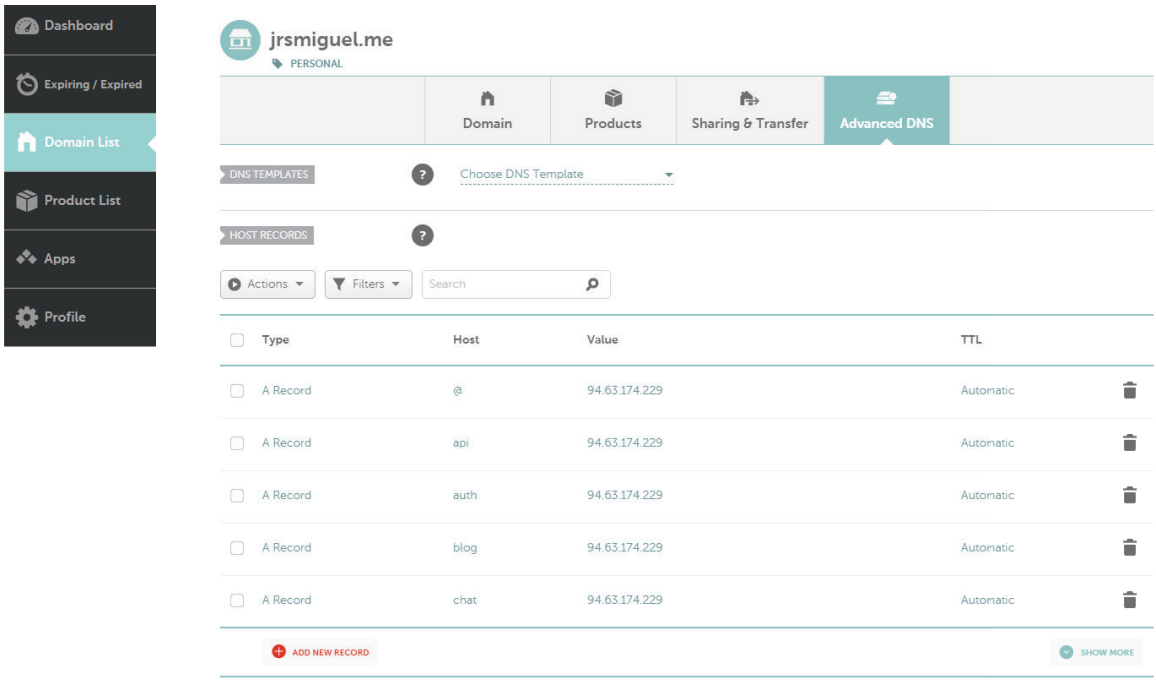


Figura 71.: Dashboard do sistema de gestão de endereçamento do servidor (acedido em junho de 2019)

D.3 DASHBOARD DE EMISSÃO DE CERTIFICADOS

Figura 72.: Dashboard do sistema de emissão de certificados para o servidor (acedido em junho de 2019)

D.4 DASHBOARD DE VERIFICAÇÃO DE TESTES UNITÁRIOS

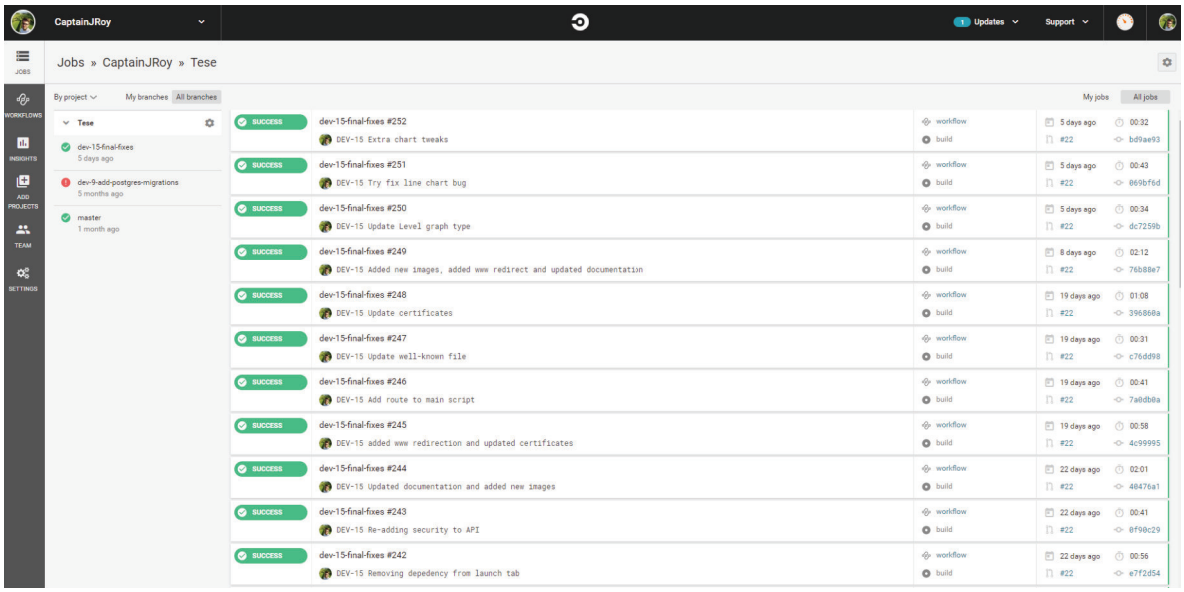


Figura 73.: Dashboard do sistema de integração continua do servidor (acedido em agosto de 2019)

D.5 DASHBOARD DE VERIFICAÇÃO DE ROTA PARA SIMULADOR

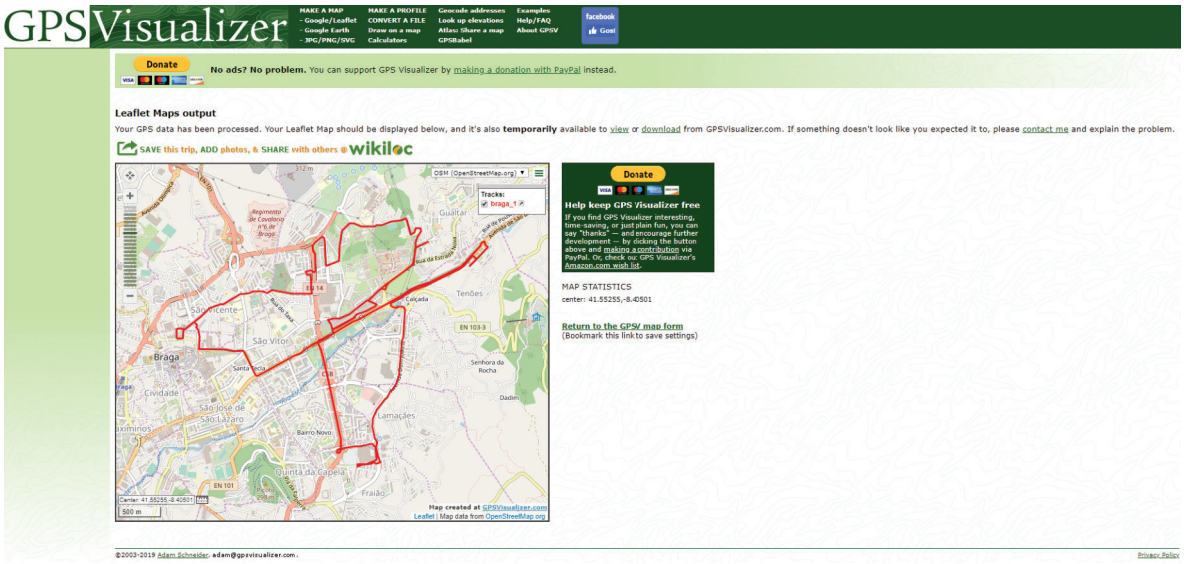


Figura 74.: Dashboard do sistema de mapeamento de rotas para o simulador (acedido em agosto de 2019)