

Projeto de Telecomuni- cações e Informática I

SERVIÇOS DE MONITORIZAÇÃO DE TRÁFEGO BASEADO
EM COLLABORATIVE SENSING

FASE 2

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Grupo 1:

- Leandro Henrique Dantas Alves A82157
- André Martins Almeida A82211
- José Eduardo da Silva Santos..... A82350

UNIVERSIDADE DO MINHO | ANO LETIVO 2019/2020



Índice

| | |
|---|-----------|
| Índice..... | 2 |
| Índice de figuras e tabelas | 4 |
| 1 Introdução..... | 6 |
| 2 Arquitetura..... | 6 |
| 3 Aquisição de dados de GPS | 7 |
| 3.1 Aplicação Android | 7 |
| 3.2 Componente simulado | 8 |
| 3.2.1 Algoritmo para a simulação do percurso | 8 |
| 3.3 CoAP REST API..... | 9 |
| 4 Serviço de <i>gateway</i> | 10 |
| 4.1 Aplicação Android | 10 |
| 4.2 <i>Script</i> para testes..... | 11 |
| 4.3 Pedidos REST..... | 12 |
| 5 Serviço de gestão de dados de veículos | 13 |
| 5.1 Base de dados | 14 |
| 5.2 Tipos de conta..... | 15 |
| 5.3 HTTP REST API | 15 |
| 5.4 Nível de trânsito..... | 18 |
| 5.4.1 Trânsito na localização do utilizador..... | 18 |
| 5.4.2 Trânsito nos segmentos de um itinerário | 20 |
| 6 Aplicação <i>Web</i>..... | 21 |
| 6.1 Registo | 22 |

| | | |
|-----------|---|-----------|
| 6.2 | Início de sessão | 23 |
| 6.3 | Mapa..... | 24 |
| 6.3.1 | Tipo padrão | 25 |
| 6.3.2 | Tipo autoritário | 26 |
| 6.3.3 | Veículos de marcha de emergência | 27 |
| 6.3.4 | Pinos de veículos..... | 28 |
| 6.4 | Detalhes da localização | 28 |
| 6.5 | Pedidos SOS | 30 |
| 6.6 | Estado do veículo | 31 |
| 6.7 | Excesso de limite de velocidade | 33 |
| 6.8 | Atribuição de tipos de conta..... | 33 |
| 6.9 | Introdução de destinatário..... | 35 |
| 7 | Software utilizado | 37 |
| 8 | Outras funcionalidades planeadas | 37 |
| 8.1 | Mapa..... | 37 |
| 8.2 | Itinerários..... | 38 |
| 8.3 | Componente de aquisição de dados e <i>gateway</i> | 39 |
| 8.4 | Notificações | 39 |
| 9 | Conclusões | 39 |
| 10 | Referências..... | 40 |

Índice de figuras e tabelas

| | |
|--|----|
| Figura 2.1 – Arquitetura do sistema..... | 6 |
| Figura 3.1 – Aplicação Android do componente de aquisição de dados. | 7 |
| Figura 3.2 – Fluxograma do algoritmo de simulação de percurso. | 8 |
| Tabela 3.1 – CoAP REST API do componente de aquisição de dados. | 9 |
| Figura 4.1 – Diagrama de casos de uso do <i>gateway</i> | 10 |
| Figura 4.2 – Aplicação Android do serviço de <i>gateway</i> | 11 |
| Figura 4.3 – Diagrama de sequência do início de sessão no <i>gateway</i> | 12 |
| Figura 4.4 – Diagrama de sequência da recolha e envio de dados no <i>gateway</i> | 12 |
| Tabela 5.1 – HTTP REST API do Serviço Central (métodos POST e PUT). | 16 |
| Tabela 5.2 – HTTP REST API do Serviço Central (métodos GET). | 17 |
| Figura 5.1 – Fluxograma do algoritmo do cálculo do nível de trânsito na localização do utilizador. | 18 |
| Figura 5.2 – Demonstração do algoritmo do cálculo do nível de trânsito na localização do utilizador. | 19 |
| Figura 5.3 – Fluxograma do algoritmo de cálculo do nível de trânsito em cada segmento de um itinerário. | 20 |
| Figura 5.4 – Demonstração do algoritmo do cálculo do nível de trânsito num segmento de um itinerário. | 20 |
| Figura 6.1 – Diagrama de casos de uso da aplicação <i>web</i> para utilizadores do tipo padrão. | 21 |
| Figura 6.2 – Diagrama de casos de uso da aplicação <i>web</i> para funcionalidades exclusivas a cada tipo. | 21 |
| Figura 6.3 – Página de registo de conta na aplicação <i>web</i> | 22 |
| Figura 6.4 – Diagrama de sequência do registo de conta na aplicação <i>web</i> | 23 |
| Figura 6.5 – Página de início de sessão na aplicação <i>web</i> | 23 |
| Figura 6.6 – Diagrama de sequência do início de sessão na aplicação <i>web</i> | 24 |
| Figura 6.7 – Página do mapa vista num computador. | 24 |
| Figura 6.8 – Página do mapa vista num dispositivo móvel. | 25 |
| Figura 6.9 – Diagrama de sequência da obtenção do mapa e da localização do utilizador e de veículos visíveis. | 25 |
| Figura 6.10 – Veículos visíveis a utilizadores do tipo padrão. | 26 |
| Figura 6.11 – Veículos visíveis a utilizadores do tipo autoritário. | 26 |
| Figura 6.12 – Veículos em SOS visíveis a utilizadores do tipo ambulância. | 27 |
| Figura 6.13 – Veículos em SOS e em excesso de velocidade visíveis a utilizadores do tipo polícia. | 27 |
| Figura 6.14 – Legenda dos tipos de pinos. | 28 |
| Figura 6.15 – Detalhes da localização de um utilizador ativo. | 29 |
| Figura 6.16 – Detalhes da localização de um utilizador inativo. | 29 |
| Figura 6.17 – Menu de SOS. | 30 |
| Figura 6.18 – Notificação de SOS recebida por um utilizador do tipo reboque. | 30 |
| Figura 6.19 – Diagrama de sequência da mudança de estado de SOS. | 30 |

| | |
|---|----|
| Figura 6.20 – Menu de mudança de estado de veículo de um transporte público. | 31 |
| Figura 6.21 – Menu de mudança de estado de veículo de um transporte de marcha de emergência..... | 31 |
| Figura 6.22 – Itinerário de resgate para um reboque..... | 32 |
| Figura 6.23 – Diagrama de sequência da mudança de estado de veículo. | 32 |
| Figura 6.24 – Notificação de reboque a caminho para resgate. | 33 |
| Figura 6.25 – Notificação de detecção de um veículo em excesso de velocidade. . | 33 |
| Figura 6.26 – Itinerário de perseguição para um polícia..... | 33 |
| Figura 6.27 – Atribuição de tipo de conta (lista de utilizadores)..... | 34 |
| Figura 6.28 – Atribuição de tipo de conta (seleção de tipo). | 34 |
| Figura 6.29 – Diagrama de sequência da atribuição do tipo de conta. | 34 |
| Figura 6.30 – Introdução do destinatário..... | 35 |
| Figura 6.31 – Desenho e informação do itinerário. | 35 |
| Figura 6.32 – Diagrama de sequência do desenho do itinerário..... | 36 |
| Figura 8.1 – Pinos com vista de cima (não implementados). | 38 |

1 Introdução

Neste projeto desenvolvemos serviços de monitorização de tráfego baseado em *collaborative sensing* que inclui um serviço de aquisição de dados, serviço de *gateway*, serviço de gestão de dados de veículos e aplicação *Web* de monitorização individual e agregada.

Neste relatório estão especificadas a arquitetura do sistema, funcionalidades de cada componente (implementadas e planeadas), tecnologias e algoritmos usados e motivos para tais.

2 Arquitetura

A arquitetura do sistema desenvolvido consiste em componentes de aquisição de dados GPS (*Global Positioning System*) da localização do utilizador, num serviço de *gateway* que adquire esses dados e pede à Google Maps API informação geocodificada sobre os mesmos para serem enviados para um serviço central que contém uma base de dados. A aplicação *Web* é acedida pelo utilizador para visualizar informação pertinente contida no serviço central.

A comunicação dos elementos desta arquitetura é baseada em protocolos REST (*Representational State Transfer*), como CoAP (*Constrained Application Protocol*) [1] e HTTP (*Hypertext Transfer Protocol*), demonstrada na Figura 2.1.

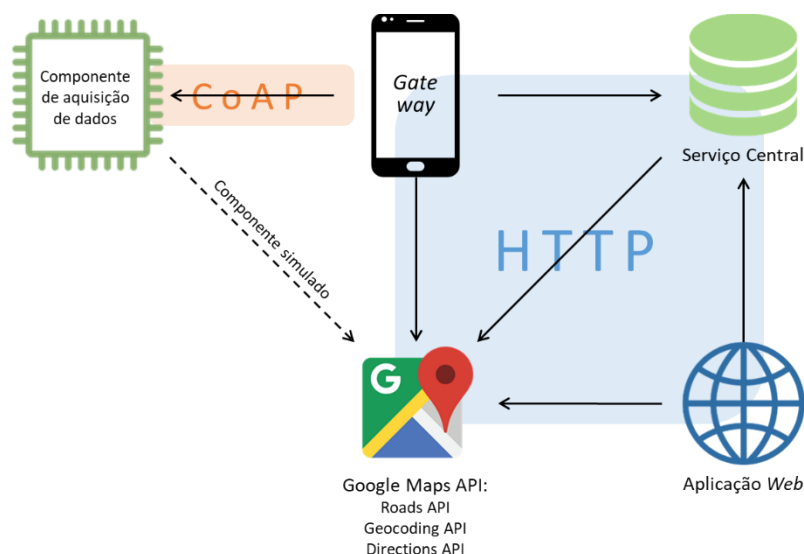


Figura 2.1 – Arquitetura do sistema.

3 Aquisição de dados de GPS

3.1 Aplicação Android

Foi desenvolvido um componente de aquisição de dados de GPS reais com as seguintes características e funcionalidades:

- Permite escolher provedor da localização [2]:
 - Satélite (*GPS PROVIDER*),
 - Rede telefónica / Wi-Fi (*NETWORK PROVIDER*);
- Obtém a cada segundo parâmetros relevantes à localização do utilizador:
 - Latitude (°),
 - Longitude (°),
 - Altitude (m),
 - Velocidade instantânea (m/s),
 - Orientação (°);
- Hospeda um servidor CoAP para conexões via Wi-Fi / LAN (*Local Area Network*) (porta 5683);
- Corre em *foreground* – a recolha da localização e o servidor CoAP continuam ativos enquanto a aplicação estiver minimizada.

Na Figura 3.1 encontra-se uma captura de ecrã desta aplicação.

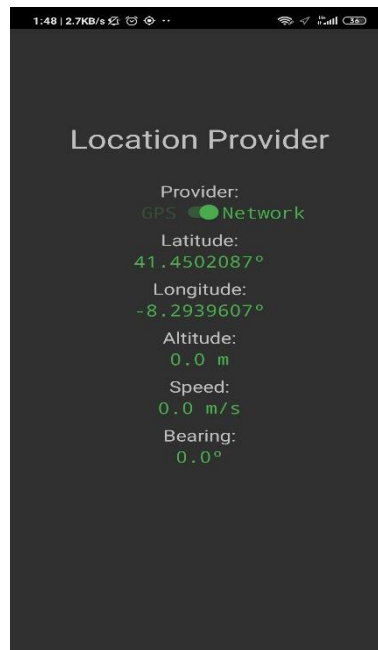


Figura 3.1 – Aplicação Android do componente de aquisição de dados.

3.2 Componente simulado

Para testes, foi desenvolvido um componente de aquisição de dados simulados por um percurso predefinido com as seguintes características e funcionalidades:

- Permite definir origem e destino do percurso e velocidade
- É pedido um itinerário à Google Maps - Directions API [3] com a origem e o destino definidos.
- Obtém a cada segundo parâmetros relevantes à localização simulada, dependendo:
 - Latitude (°),
 - Longitude (°),
 - Altitude = 200 m (valor atribuído manualmente),
 - Orientação (°);
- Hospeda um servidor CoAP numa porta à escolha – portas à escolha permitem várias instâncias deste *script* pois os testes foram feitos na mesma rede e pretendeu-se simular vários veículos.

3.2.1 Algoritmo para a simulação do percurso

A cada segundo, o veículo simulado percorre uma certa distância dependendo da sua velocidade, o algoritmo da Figura 3.2 serve para obter as coordenadas e a orientação do veículo a cada segundo ao ser percorrida tal distância.

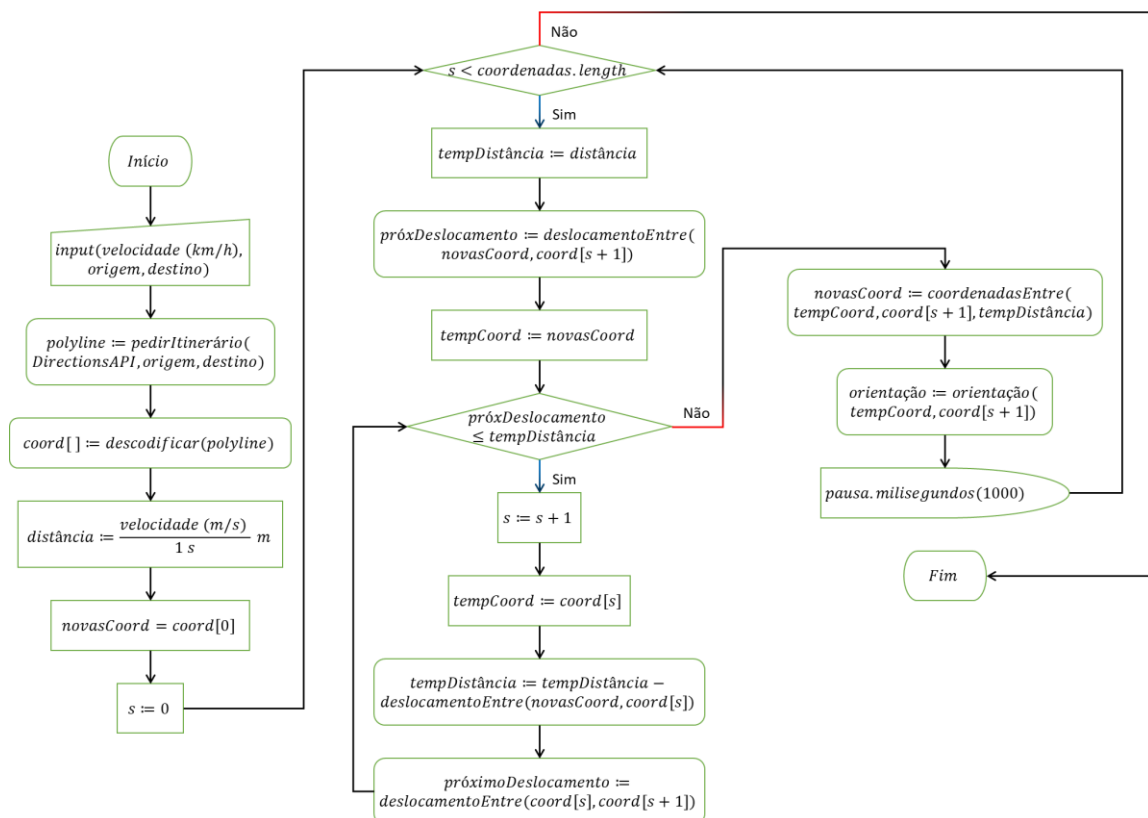


Figura 3.2 – Fluxograma do algoritmo de simulação de percurso.

A função *descodificar(polyline)* [4] descodifica o conjunto de caracteres (*polyline*) devolvido pela Directions API num conjunto de coordenadas que ao se conectarem por linhas retas, formam o desenho do itinerário.

A função *deslocamentoEntre(A, B)* [5] calcula o deslocamento (em metros) entre as coordenadas *A* e *B*, e é definida pela expressão:

$$\text{acos}(\sin(\varphi_A) \times \sin(\varphi_B) + \cos(\varphi_A) \times \cos(\varphi_B) \times \cos(\theta_B - \theta_A)) \times R$$

- φ_A = latitude (rad) de *A*;
- θ_A = longitude (rad) de *A*;
- φ_B = latitude (rad) de *B*;
- θ_B = longitude (rad) de *B*;
- R = raio da Terra (6371×10^3 m).

A função *orientação(A, B)* [5] calcula a orientação (em radianos) entre as coordenadas *A* e *B* através da expressão:

$$\text{atan2}\left(\frac{\sin(\theta_B - \theta_A) \times \cos(\varphi_B)}{\cos(\varphi_A) \times \sin(\varphi_B) - \sin(\varphi_A) \times \cos(\varphi_B) \times \cos(\theta_B - \theta_A)}\right)$$

A função *coordenadasEntre(A, B, d)* [5] calcula as coordenadas (φ , ϑ) (em radianos) pertencentes ao segmento de reta entre *A* e *B* a uma distância *d* (em metros) de *A*, definida pelas expressões:

$$\varphi = \text{asin}\left(\sin(\varphi_A) \times \cos\left(\frac{d}{R}\right) + \cos(\varphi_A) \times \sin\left(\frac{d}{R}\right) \times \cos(\text{orientação}(A, B))\right)$$

$$\theta = \theta_A + \text{atan2}\left(\frac{\sin(\text{orientação}(A, B)) \times \sin\left(\frac{d}{R}\right) \times \cos(\varphi_A)}{\cos\left(\frac{d}{R}\right) - \sin(\varphi_A) \times \sin(\varphi)}\right)$$

3.3 CoAP REST API

Para que os dados da localização adquiridos sejam acedidos pelo serviço de *gateway*, foi desenvolvida um REST API (*Application Programming Interface*) de CoAP com os pedidos da Tabela 3.1.

Tabela 3.1 – CoAP REST API do componente de aquisição de dados.

| Tarefa | Método | Caminho | Resposta (<i>String</i>) |
|-------------------|--------|-----------|---|
| Obter localização | GET | /location | {latitude} {longitude} {altitude} {velocidade} {orientação} |

Foi escolhido o CoAP como protocolo aplicacional porque nos foi recomendado pelos docentes, porque é o ideal para comunicação M2M (*machine-to-machine*) na mesma rede, tem estrutura semelhante ao HTTP e está disponível em bibliotecas de Android.

4 Serviço de *gateway*

4.1 Aplicação Android

Foi desenvolvido um serviço de *gateway* com as seguintes características e funcionalidades:

- Permite ao utilizador iniciar sessão numa conta registada para obter o seu número identificador;
- Permite introduzir o endereço IP (*Internet Protocol*) do servidor do componente de aquisição de dados de GPS (porta 5683);
- Obtém os parâmetros da localização recolhidos pelo componente através de pedidos CoAP a cada segundo;
- Obtém informação sobre a localização através de pedidos HTTP à Google Maps API:
 - Roads API [6]:
 - Coordenadas do segmento de rua mais próximo (para que a localização do utilizador fique sempre na estrada mais próxima),
 - Limite de velocidade da rua,
 - Geocoding API [7]:
 - Endereço (nome) da rua,
 - Localidade;
- Envia dados recolhidos para o Serviço Central através de pedidos HTTP;
- Corre em *foreground* – Tanto o componente de aquisição de dados como o *gateway* têm esta funcionalidade, o que permite a utilização de ambos no mesmo dispositivo.

Esta aplicação também pode se conectar com um componente de aquisição de dados simulado desde que o servidor do mesmo esteja hospedado na porta 5683.

Estas ações estão representadas no diagrama de casos de uso da Figura 4.1.

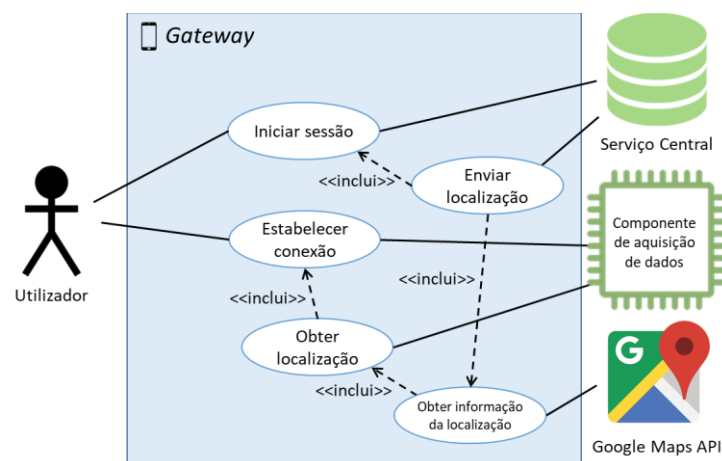


Figura 4.1 – Diagrama de casos de uso do *gateway*.

Na Figura 4.2 encontra-se uma captura de ecrã desta aplicação.

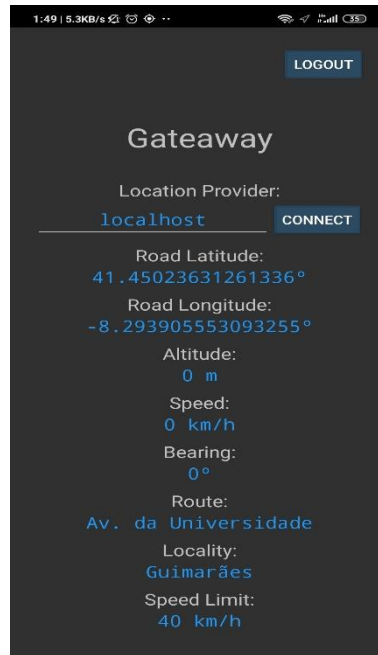


Figura 4.2 – Aplicação Android do serviço de *gateway*.

4.2 Script para testes

Para serem testados múltiplos veículos simulados, foram necessárias múltiplas instâncias do *gateway* que comunicassem com múltiplos componentes de aquisição de dados simulados, para tal, foi desenvolvido um *script* com as seguintes características e funcionalidades:

- Permite introduzir o endereço e porta do servidor do componente de aquisição de dados de GPS e ID do utilizador;
- Obtém os parâmetros da localização recolhidos pelo componente através de pedidos CoAP;
- Obtém informação sobre a localização através de pedidos HTTP à Google Maps API;
- Envia dados recolhidos para o Serviço Central.

4.3 Pedidos REST

O serviço de *gateway* para recolher, interpretar e armazenar dados, comunica com o componente de aquisição de dados, com a Google Maps API e com o Serviço Central.

Na Figura 4.3 e Figura 4.4 estão representados os diagramas de sequência dos pedidos REST feitos pelo *gateway*.

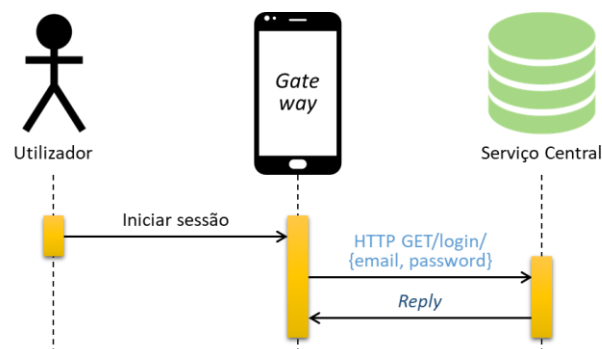


Figura 4.3 – Diagrama de sequência do início de sessão no *gateway*.

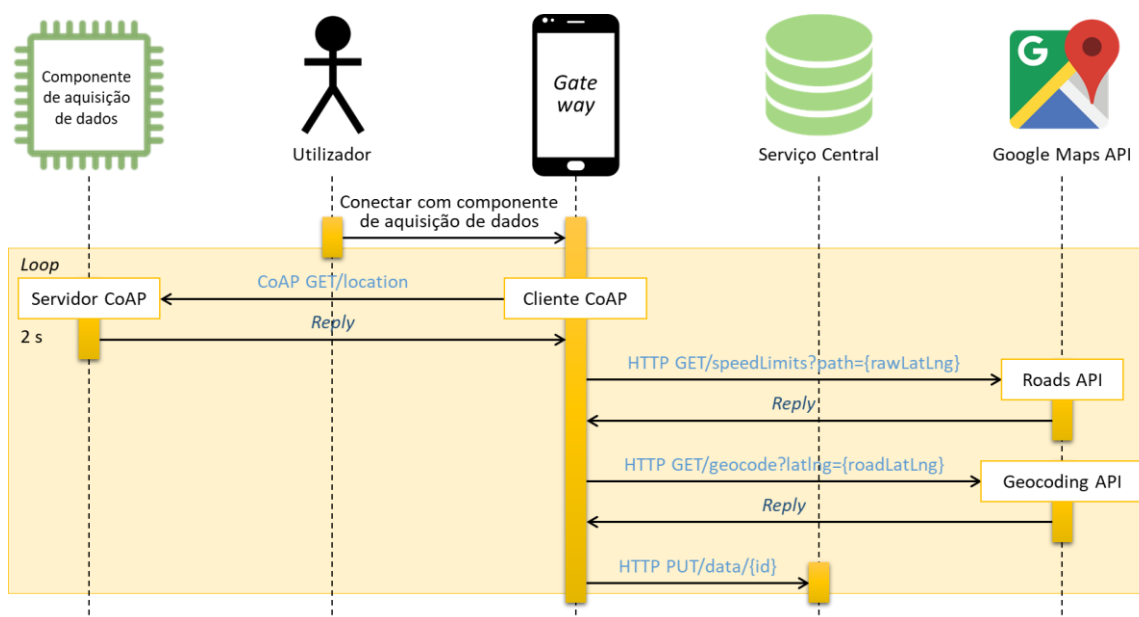


Figura 4.4 – Diagrama de sequência da recolha e envio de dados no *gateway*.

5 Serviço de gestão de dados de veículos

O serviço de gestão de dados de veículos é responsável por processar e armazenar os dados de veículos e dos utilizadores de modo a servir ao Serviço *Web* o conjunto de aplicações de monitorização individual e agregada das experiências de condução.

Este serviço foi desenvolvido num Raspberry Pi 3 Model B+ [8] pois o grupo já estava familiarizado com o mesmo pois foi usado em UCs (Unidades Curriculares) de anos anteriores para alojar bases de dados, APIs e aplicações *web*, e como estava alojado numa rede doméstica privada de um dos membros, não era preocupante o uso excessivo de memória e de pedidos pois não eram aplicadas taxas.

Como o Raspberry Pi não é o ideal para sistemas que requerem escalabilidade elevada, o *deployment* da API foi feito numa *Compute Engine* e o da base de dados numa *SQL Instance*, ambos da Google Cloud Platform [9]. Foi escolhida esta plataforma porque era associável ao projeto criado para uso das APIs da Google Maps. Além disso supusemos que o desempenho dos pedidos à Google Maps API seria superior se fossem feitos a partir de uma máquina da Google.

A aplicação *web* também foi desenvolvida no mesmo Raspberry Pi com *deployment* feito na mesma *Compute Engine*.

5.1 Base de dados

A base de dados do serviço contém as seguintes tabelas:

- **User** – Utilizadores:
 - **id** (chave primária, auto-incrementável) – Identificador numérico,
 - **email** – Endereço e-mail, único para cada utilizador,
 - **name** – Nome do utilizador,
 - **password** – Palavra-passe,
 - **typeid** – Identificador numérico do tipo de conta,
 - **latitude** – Latitude (°) do segmento de rua mais próximo ao veículo,
 - **longitude** – Longitude (°) do segmento de rua mais próximo ao veículo,
 - **altitude** – Altitude (m) de veículo,
 - **speed** – Velocidade instantânea (km/h) do veículo,
 - **bearing** – Orientação (°) do veículo,
 - **speedLimit** – Limite de velocidade da rua mais próxima,
 - **authLocalityId** (chave estrangeira **Locality.id**) – Identificador numérico da localidade registada pelo utilizador
 - **status** – Estado do veículo:
 - 0 = *off-duty* / ocupado / sem marcha de emergência,
 - 1 = livre / marcha de emergência,
 - n = marcha de emergência com alvo no veículo cujo **id** = n ,
 - **sos** – Estado de SOS:
 - 0 = sem SOS,
 - 5 = ambulância,
 - 6 = polícia,
 - 7 = reboque,
 - **routeId** (chave estrangeira **Route.id**) – Identificador numérico da rua,
 - **lastTimestamp** – *Timestamp* da última mensagem de dados recebida, usada para determinar a inatividade do utilizador;
- **Route** – Rua (contém apenas as ruas recebidas por um *gateway* ou contidas num itinerário definido por um utilizador):
 - **id** (chave primária, auto-incrementável) – Identificador numérico,
 - **name** – Designação da rua,
 - **localityId** (chave estrangeira de **Locality.id**) – Identificador numérico da localidade;
- **Locality** – Localidade (contém apenas as localidades recebidas pelo *gateway* ou contidas num itinerário definido por um utilizador):
 - **id** (chave primária, auto-incrementável) – Identificador numérico,
 - **name** – Designação da localidade.

5.2 Tipos de conta

Cada utilizador tem-lhe atribuído um tipo de conta para que este exerça funções específicas e/ou tenha acesso a certas funcionalidades na aplicação *Web*:

1. Padrão:

- Tipo padrão dos utilizadores, atribuído durante o registo da conta;

2. Autoritário:

- Acesso à localização de todos os veículos,
- Consegue alterar o tipo de outros utilizadores,
- Recebe pedidos de SOS de qualquer tipo e notificações de excesso de velocidade;

3. Táxi:

- Estados: *off-duty* / ocupado (0) e livre (1, visível para todos os utilizadores dentro da localidade);

4. Autocarro:

- Estados: *off-duty* / ocupado (0) e livre (1, visível para todos os utilizadores dentro da localidade);

5. Ambulância:

- Estados: sem emergência (0) e marcha de emergência (≥ 1 , visível para todos os utilizadores dentro da localidade),
- Recebe pedidos de SOS de ambulância;

6. Polícia:

- Estados: sem emergência (0) e marcha de emergência (≥ 1 , visível para todos os utilizadores dentro da localidade),
- Recebe pedidos de SOS de polícia e notificações de excesso de velocidade;

7. Reboque:

- Estados: sem emergência (0) e marcha de emergência (≥ 1 , visível apenas para quem estiver a ser socorrido),
- Recebe pedidos de SOS de reboque.

5.3 HTTP REST API

O Serviço Central disponibiliza uma API para receber pedidos que necessitam de acesso à base de dados e/ou de outras ações específicas. Na Tabela 5.1 e Tabela 5.2 encontram-se os pedidos disponíveis.

Tabela 5.1 – HTTP REST API do Serviço Central (métodos POST e PUT).

| Tarefa | Método | Caminho | Corpo (JSON) |
|-------------------------|--------|--------------|--|
| Registo | POST | /signup | {email, name, password, locality} |
| Enviar dados | PUT | /data/{id} | {latitude, longitude, altitude, speed, bearing, route, locality, speedLimit} |
| Mudar estado de SOS | PUT | /sos/{id} | {sos} |
| Mudar estado do veículo | PUT | /status/{id} | {status} |
| Mudar estado do veículo | PUT | /types/{id} | {typeld} |

- POST/signup – Registo de conta na aplicação *web*:
 - **email** – endereço e-mail,
 - **name** – nome do utilizador,
 - **password** – palavra-passe,
 - **locality** – localidade (este campo serve para que quando a um utilizador lhe for atribuído um tipo de conta, este tenha as suas funções exercidas na localidade escolhida);
- PUT/data/{U's id} – Envio de dados da localização do utilizador *U* pelo *gateway*:
 - **latitude** – latitude (°) do segmento de rua mais próximo ao veículo,
 - **longitude** – longitude (°) do segmento de rua mais próximo ao veículo,
 - **altitude** – altitude (m) de veículo,
 - **speed** – velocidade instantânea (km/h) do veículo,
 - **bearing** – orientação (°) do veículo,
 - **route** – rua onde o veículo se encontra,
 - **locality** – localidade onde o veículo se encontra,
 - **speedLimit** – limite de velocidade da rua;
- PUT/sos/{U's id} – Mudança de estado de SOS do utilizador *U* na aplicação *web*:
 - **sos** – identificador do estado de SOS:
 - 0 = sem SOS,
 - 5 = pedido de ambulância,
 - 6 = pedido de polícia,
 - 7 = pedido de reboque;
- PUT/status/{U's id} – Mudança de estado de veículo do utilizador *U* (exclusivo para táxis, autocarros, ambulâncias, polícias e reboques) na aplicação *web*:
 - **status** – identificador do estado do veículo:
 - 0 = *off-duty* / ocupado / sem marcha de emergência,
 - 1 = livre / marcha de emergência,
 - *n* = marcha de emergência com alvo no veículo cujo id = *n*;
- PUT/types{U's id} – Mudança de tipo de conta do utilizador *U* feito exclusivamente por um utilizador autoritário da localidade de *U* na aplicação *web*:
 - **typeld** – tipo de conta.

Tabela 5.2 – HTTP REST API do Serviço Central (métodos GET).

| Tarefa | Método | Caminho | Resposta (JSON) |
|---------------------------|--------|--|---|
| Iniciar sessão | GET | /login/{email}/{password} | {id, name, typeld, status, sos} |
| Obter dados | GET | /data/{id} | {latitude, longitude, altitude, speed, bearing, speedLimit, lastTimestamp, routeName, localityName, traffic, vehicles: [{id, name, typeld, latitude, longitude, speed, bearing, speedLimit, status, sos}, ...]} |
| Lista de utilizadores | GET | /users/{id} | {users: [{id, name, typeld}, ...]} |
| Trânsito de um itinerário | GET | /traffic/{segments:[{latitude, longitude, routeLength, routeDuration}, ...]} | {delay, traffic: [...]} |

- GET/login/{email}/{password} – Início de sessão feito no *gateway* ou na aplicação *web*:
 - **id** – identificador do utilizador,
 - **name** – nome do utilizador,
 - **typeld** – tipo de conta,
 - **status** – identificador do estado do veículo,
 - **sos** – identificador do estado de SOS,
- GET/data/{U's id} – Obtenção de dados da localização do utilizador *U* e dos veículos visíveis para *U*:
 - **latitude** – latitude (°) do segmento de rua mais próximo ao veículo,
 - **longitude** – longitude (°) do segmento de rua mais próximo ao veículo,
 - **altitude** – altitude (m) de veículo,
 - **speed** – velocidade instantânea (km/h) do veículo,
 - **bearing** – orientação (°) do veículo,
 - **speedLimit** – limite de velocidade da rua;
 - **routeName** – rua onde o veículo se encontra,
 - **localityName** – localidade onde o veículo se encontra,
 - **traffic** – nível de trânsito na localização do utilizador:
 - 0 = livre,
 - 1 = moderado,
 - 2 = congestionado,
 - **vehicles** [array] – informação e localização dos veículos visíveis;
- GET/users/{A's id} – Listagem dos utilizadores registados na localidade do utilizador autoritário *A*, pedida por *A* na aplicação *web*. Esta lista serve para o autoritário alterar o tipo de um utilizador escolhido:
 - **users** [array] – lista de utilizadores:
 - **id** – identificador do utilizador,
 - **name** – nome do utilizador,
 - **typeld** – tipo do utilizador;

- GET/traffic/{segments} – Listagem do nível de trânsito em cada segmento de um itinerário:
 - **segments** [array] – segmentos do itinerário (obtidos pela Google Maps - Directions API):
 - **latitude** – latitude (°) do centro do segmento,
 - **longitude** – longitude (°) do centro do segmento,
 - **routeLength** – comprimento do segmento,
 - **routeDuration** – duração média do percurso do segmento,
 - **delay** – atraso total calculado pelos níveis de trânsito e duração de cada segmento,
 - **traffic** [array] – nível de trânsito em cada segmento.

5.4 Nível de trânsito

5.4.1 Trânsito na localização do utilizador

O trânsito na localização do utilizador, obtido em GET/data na aplicação web, é calculado segundo o algoritmo da Figura 5.1.

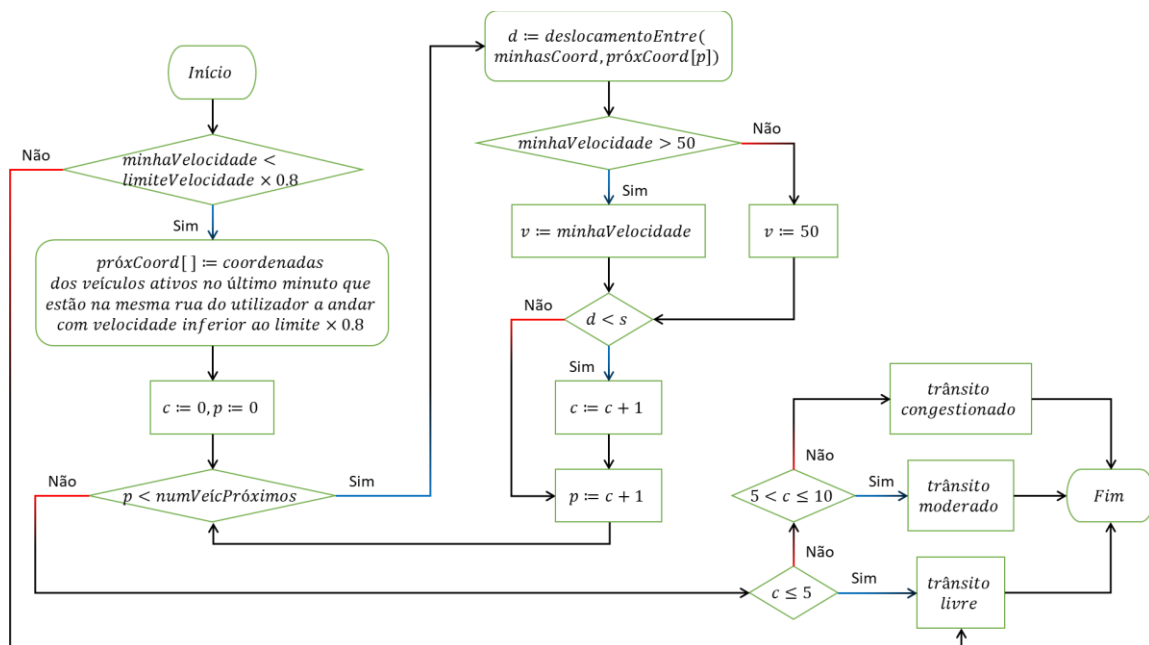


Figura 5.1 – Fluxograma do algoritmo do cálculo do nível de trânsito na localização do utilizador.

O algoritmo tem em conta o limite de velocidade porque não faz sentido considerar que há trânsito quando um veículo está com velocidade superior a 80% do limite de velocidade da rua em que se encontra.

Um exemplo de cálculo de nível de trânsito está demonstrado na Figura 5.2.

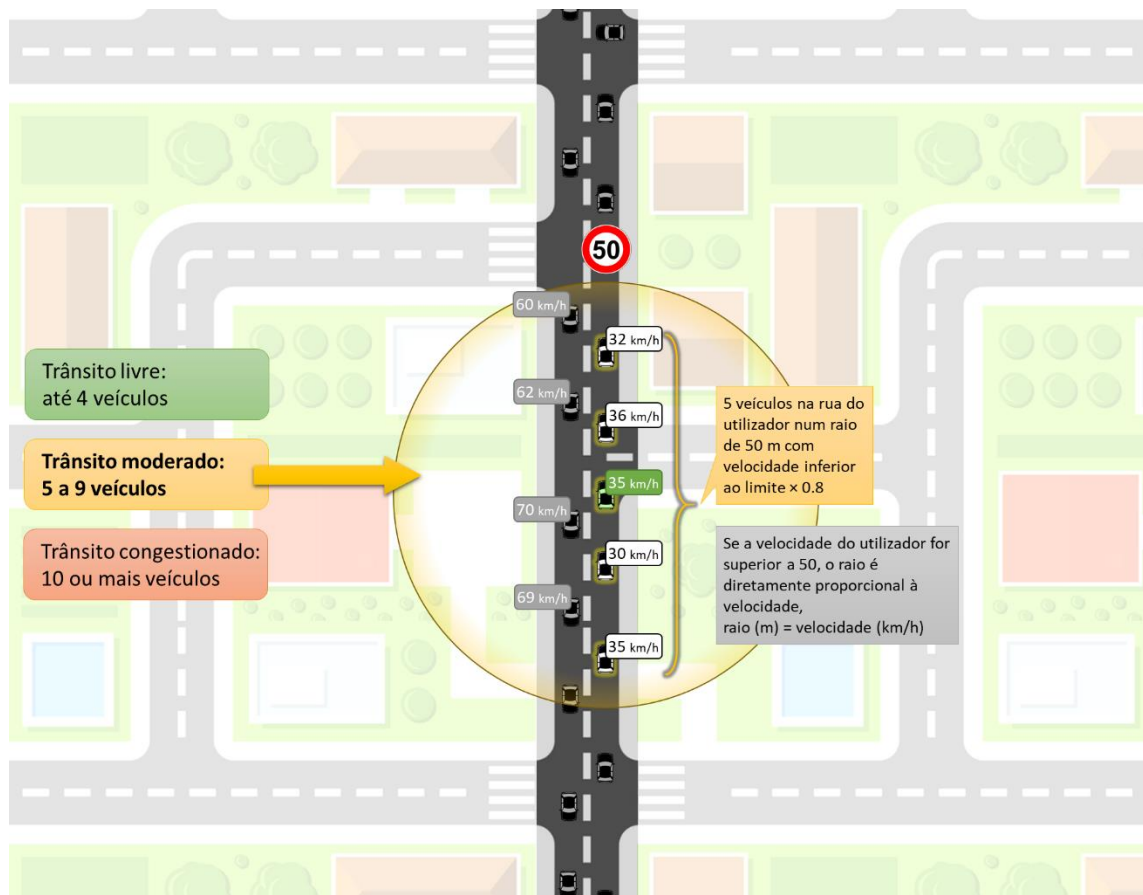


Figura 5.2 – Demonstração do algoritmo do cálculo do nível de trânsito na localização do utilizador.

5.4.2 Trânsito nos segmentos de um itinerário

O trânsito em cada segmento de um itinerário e atraso total, obtidos em GET/traffic na aplicação *web*, são calculados segundo o algoritmo da Figura 5.3.

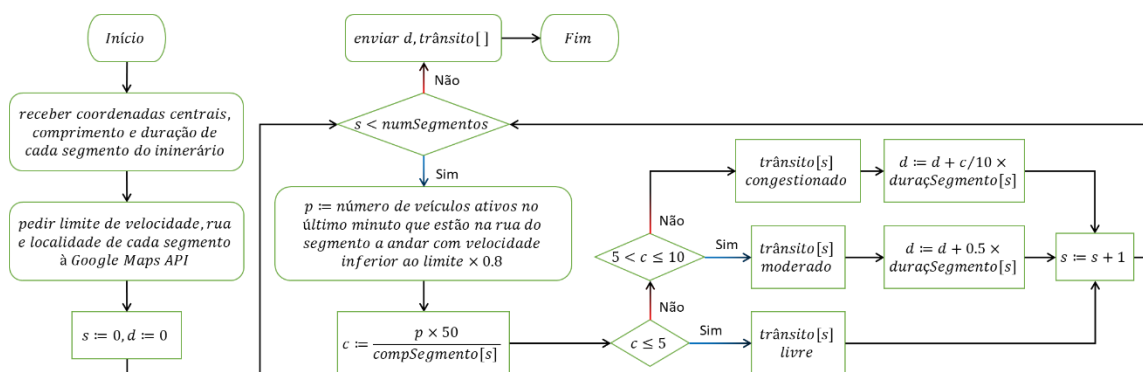


Figura 5.3 – Fluxograma do algoritmo de cálculo do nível de trânsito em cada segmento de um itinerário.

Um exemplo de cálculo do nível de trânsito nos segmentos de um itinerário está demonstrado na Figura 5.4.

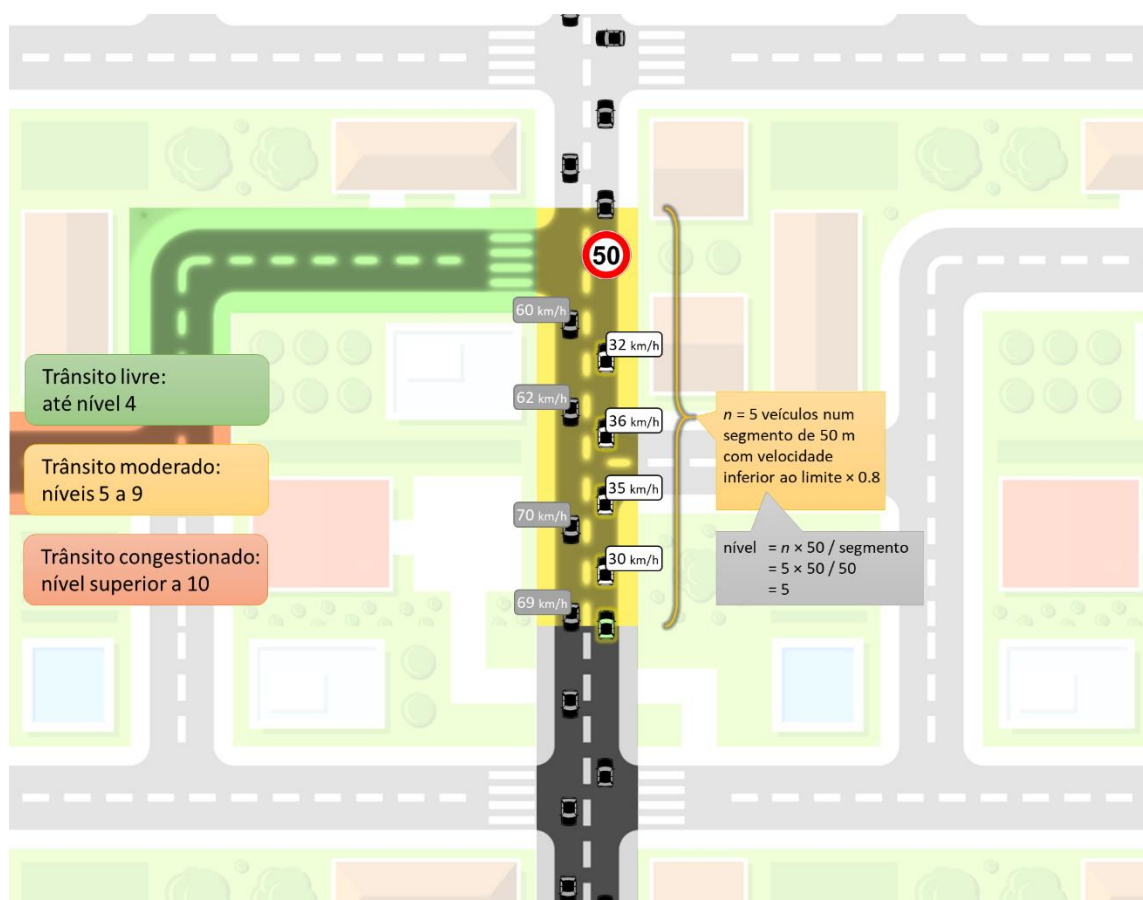


Figura 5.4 – Demonstração do algoritmo do cálculo do nível de trânsito num segmento de um itinerário.

6 Aplicação Web

A aplicação *Web* trata-se da interface disponibilizada ao utilizador para que este tenha acesso à informação da sua localização, ao trânsito nessa localização ou num itinerário que pretenda tomar, a pedidos de SOS e a funcionalidades exclusivas ao seu tipo de conta.

Na Figura 6.1 e Figura 6.2 estão representados os diagramas de caso de uso das funcionalidades de cada tipo de utilizador.

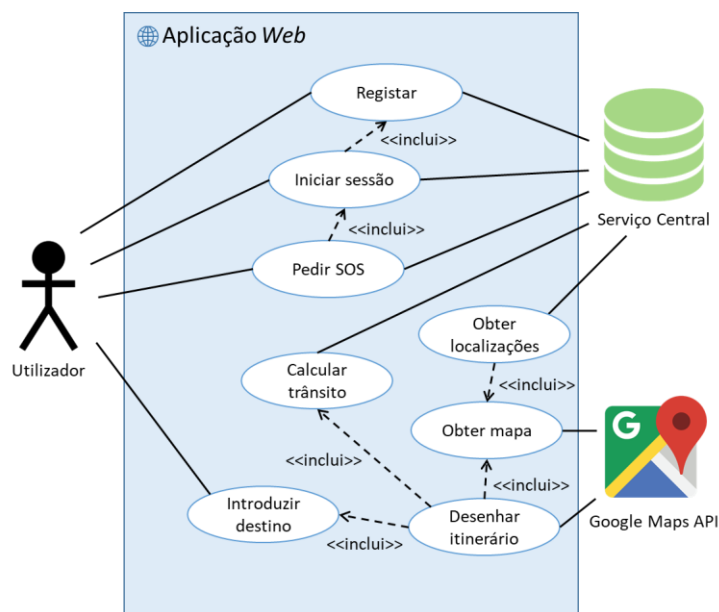


Figura 6.1 – Diagrama de casos de uso da aplicação *web* para utilizadores do tipo padrão.

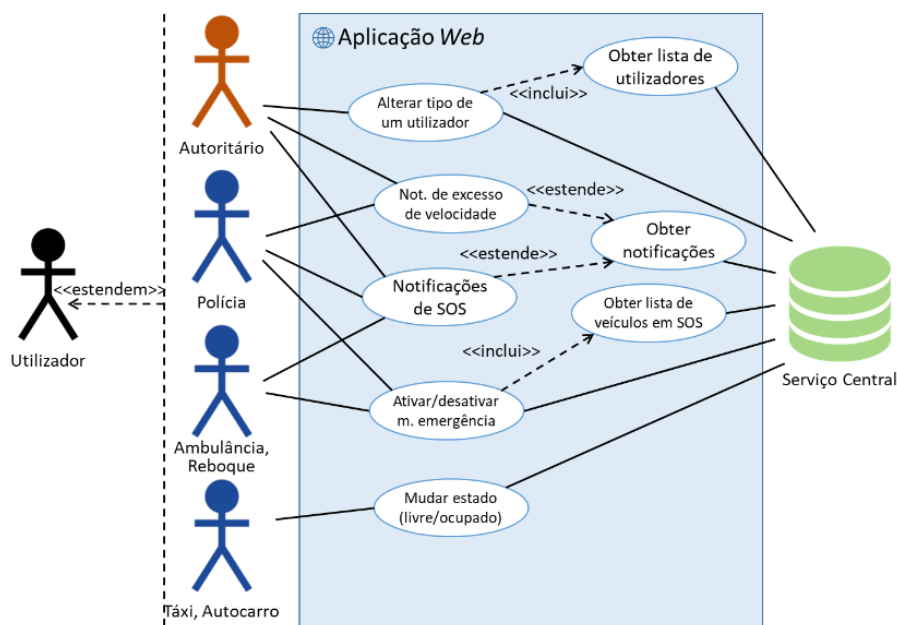


Figura 6.2 – Diagrama de casos de uso da aplicação *web* para funcionalidades exclusivas a cada tipo.

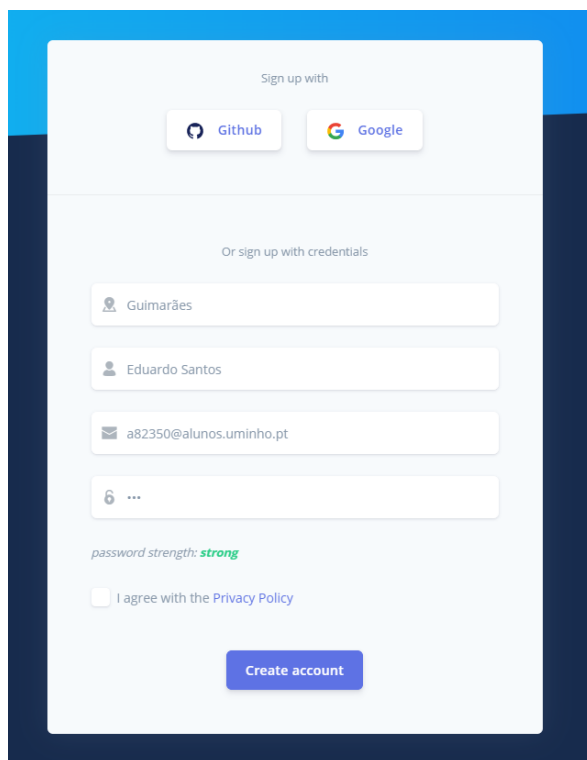
6.1 Registo

Para que o utilizador possa usufruir das funcionalidades do serviço, este deve ter uma conta registada.

Os campos do registo são:

- Endereço e-mail;
- Nome do utilizador;
- Palavra-passe;
- Localidade onde as funções exclusivas ao tipo têm efeito.

Na Figura 6.3 encontra-se a página de registo.



Sign up with

Github Google

Or sign up with credentials

Guimarães

Eduardo Santos

a82350@alunos.uminho.pt

6 ...

password strength: **strong**

☐ I agree with the Privacy Policy

Create account

Figura 6.3 – Página de registo de conta na aplicação web.

Na Figura 6.4 está representado o diagrama de sequência com os pedidos REST feitos no registo.

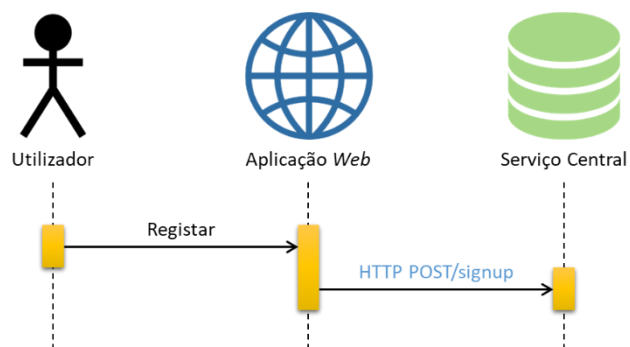


Figura 6.4 – Diagrama de sequência do registo de conta na aplicação web.

6.2 Início de sessão

Para um utilizador aceder às informações relevantes ao seu tipo de conta, este deve iniciar sessão numa conta registada, preenchendo os seguintes campos:

- Endereço e-mail;
- Palavra-passe.

Na Figura 6.5 encontra-se o a página de início de sessão.

A imagem mostra a interface de utilizador para o início de sessão. No topo, há a opção 'Sign in with' com botões para 'Github' e 'Google'. Abaixo, a opção 'Or sign in with credentials' apresenta campos para o endereço de e-mail (contendo 'a82350@alunos.uminho.pt') e a palavra-passe (representada por pontos). Há também uma opção 'Remember me' com uma caixa de seleção e um botão 'Sign in' em azul. Na base da caixa de entrada, há links para 'Forgot password?' e 'Create new account'.

Figura 6.5 – Página de início de sessão na aplicação web.

Na Figura 6.6 está representado o diagrama de sequência com os pedidos REST feitos no início de sessão.

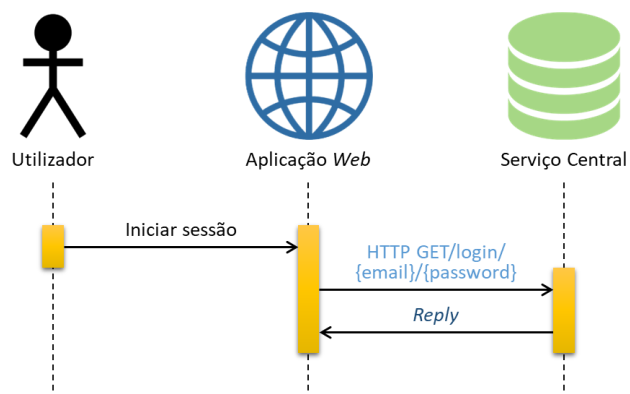


Figura 6.6 – Diagrama de sequência do início de sessão na aplicação web.

6.3 Mapa

Ao ser iniciada a sessão numa conta, é apresentado ao utilizador um mapa proveniente do Google Maps SDK [10].

Esta interface tanto é compatível para computador, demonstrada na Figura 6.7, como para dispositivos móveis, demonstrada na Figura 6.8, pois a estrutura da mesma é responsiva ao tamanho do ecrã. A interface para dispositivos móveis é útil para se usufruir das funcionalidades em tempo real numa viagem, estando as aplicações de aquisição de dados GPS e de *gateway* em *foreground*.

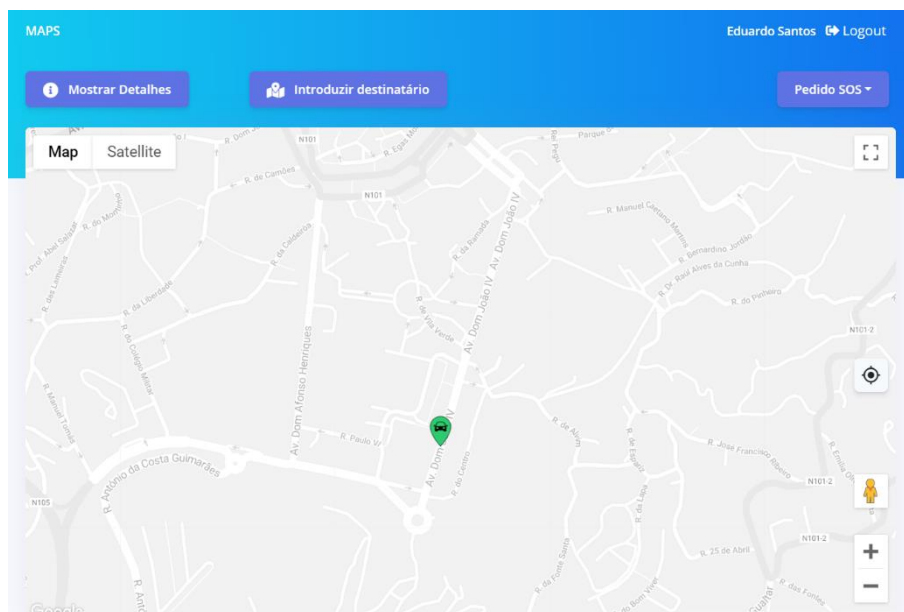


Figura 6.7 – Página do mapa vista num computador.

6.3.3 Veículos de marcha de emergência

Utilizadores do tipo ambulância, polícia e reboque, para além das funcionalidades padrão, têm também acesso à localização de veículos que fizeram pedido de SOS para o respetivo tipo, como é exemplificado na Figura 6.12.

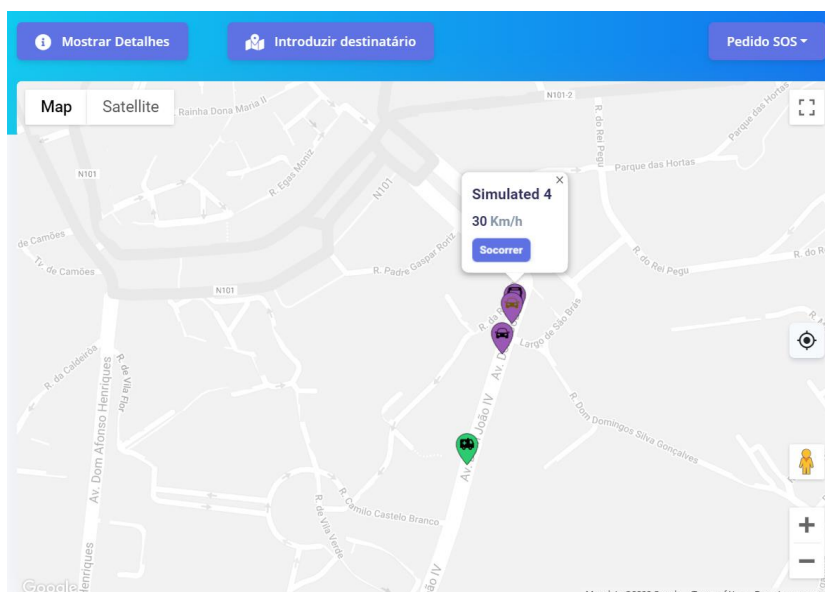


Figura 6.12 – Veículos em SOS visíveis a utilizadores do tipo ambulância.

Tipos polícia têm também acesso à localização de veículos que ultrapassem 40 km/h do limite de velocidade (contraordenações muito graves) da rua em que se encontram, demonstrado na Figura 6.13.

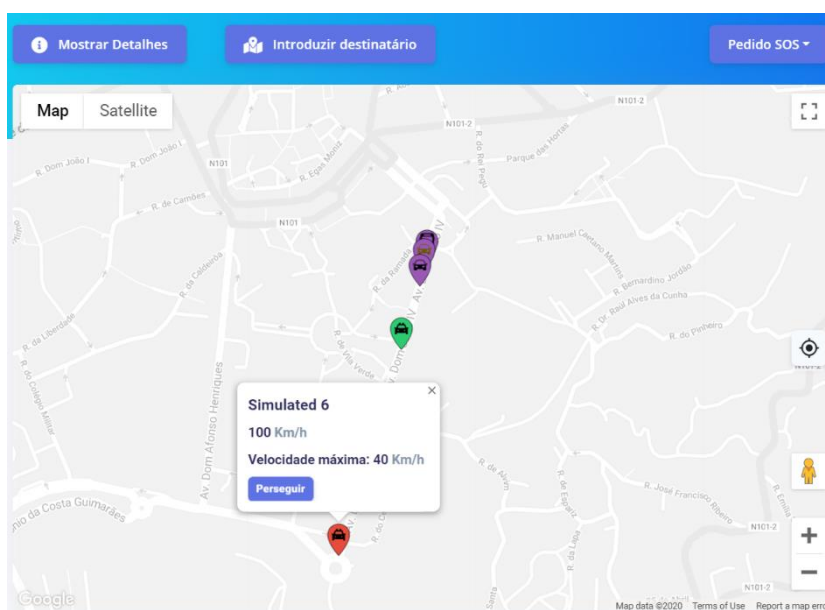


Figura 6.13 – Veículos em SOS e em excesso de velocidade visíveis a utilizadores do tipo polícia.

6.3.4 Pinos de veículos

Cada veículo tem-lhe associado um pino que represente o seu tipo e estado. Os tipos de pino estão legendados na Figura 6.14.

| | | | | | | | |
|---|--------|-------------|------|-----------|------------|---------|---------|
| Utilizador ativo | | | | | | | |
| Utilizador inativo | | | | | | | |
| Transporte público livre ou veículo em marcha de emergência | | | | | | | |
| Veículo com pedido de SOS | | | | | | | |
| Veículo com excesso de velocidade | | | | | | | |
| Veículos vistos por Autoritários | | | | | | | |
| | Padrão | Autoritário | Táxi | Autocarro | Ambulância | Polícia | Reboque |

Figura 6.14 – Legenda dos tipos de pinos.

6.4 Detalhes da localização

Os detalhes da localização do utilizador podem lhe ser revelados, contendo:

- Nome da rua;
- Nome da localidade;
- Velocidade instantânea;
- Limite de velocidade da rua;
- Nível de trânsito (explicado na secção anterior 5.4.1).

Um exemplo destes detalhes está na Figura 6.15.

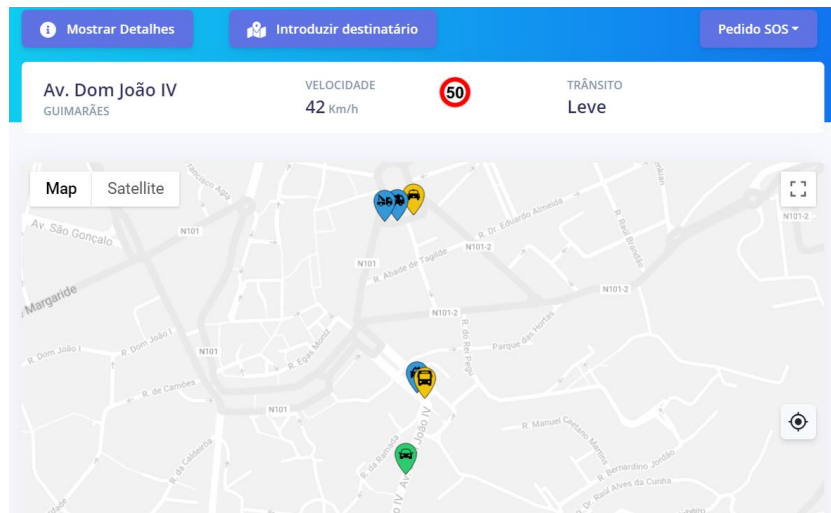


Figura 6.15 – Detalhes da localização de um utilizador ativo.

Se o utilizador estiver inativo (última recolha de localização feita à mais de um minuto), será-lhe apresentado nos detalhes o tempo de inatividade em vez da velocidade instantânea e o limite de velocidade, como é exemplificado na Figura 6.16.

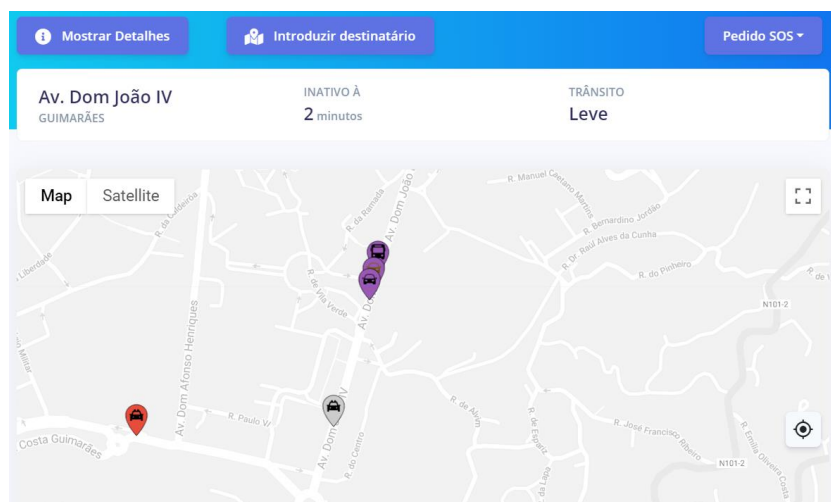


Figura 6.16 – Detalhes da localização de um utilizador inativo.

Veículos inativos são excluídos dos veículos visíveis a outros utilizadores.

6.5 Pedidos SOS

Todos os utilizadores têm acesso a um menu para fazeres pedidos de SOS com opções de ambulância, polícia e reboque, demonstrado na Figura 6.17.

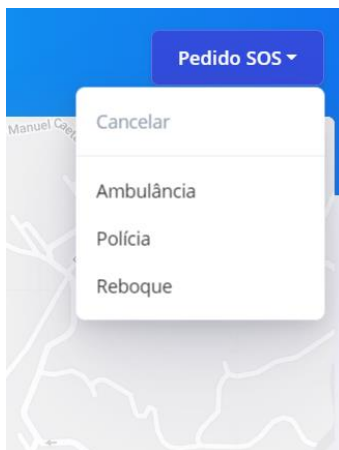


Figura 6.17 – Menu de SOS.

Quando é feito um pedido, utilizadores do respetivo tipo são notificados, como o exemplo da Figura 6.18.

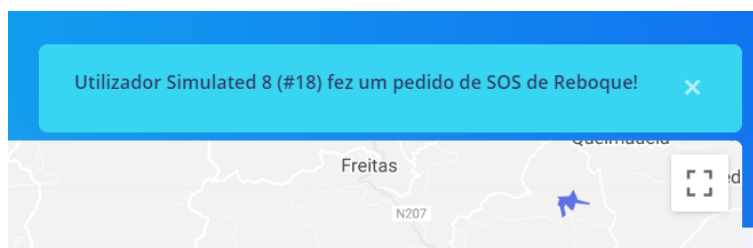


Figura 6.18 – Notificação de SOS recebida por um utilizador do tipo reboque.

Tipos autoritários recebem notificações de pedidos de SOS de qualquer tipo.

Os pedidos de SOS seguem o diagrama de sequência da Figura 6.19.

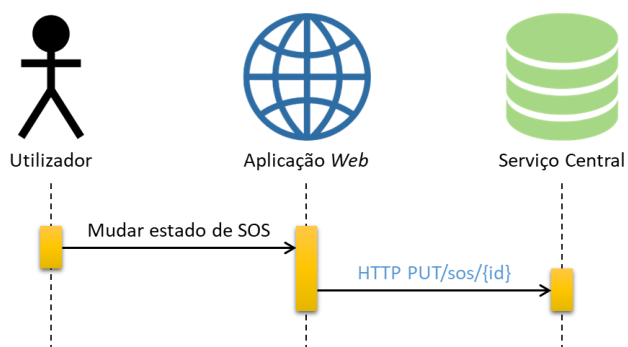


Figura 6.19 – Diagrama de sequência da mudança de estado de SOS.

As notificações são geradas a partir dos pedidos GET/data quando na resposta, os veículos visíveis incluem novos veículos com pedido de SOS.

6.6 Estado do veículo

Transportes públicos (táxi e autocarro) têm acesso ao menu de mudança de estado da Figura 6.20 para revelarem a sua localização nos veículos visíveis a outros utilizadores.

As opções deste menu são:

- Ocupado / *off-duty* – localização invisível para outros utilizadores;
- Livre – localização visível para outros utilizadores.

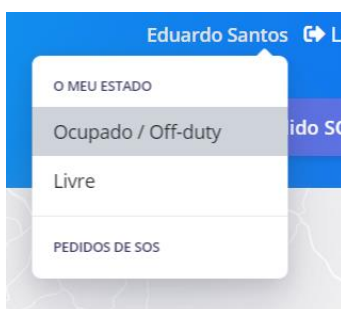


Figura 6.20 – Menu de mudança de estado de veículo de um transporte público.

Transportes de marcha de emergência têm acesso a um menu semelhante (Figura 6.21) mas com as seguintes opções:

- Sem marcha de emergência – localização invisível para outros utilizadores;
- Em marcha de emergência – localização visível para outros utilizadores.
- Pedidos de SOS – lista de utilizadores que fizeram pedidos de SOS do respetivo tipo.

Ao ser selecionado um utilizador da lista de pedidos de SOS, o veículo entra em marcha de emergência e é traçado um itinerário, como o da Figura 6.22, até ao utilizador que fez o pedido, assim como os detalhes desse utilizador (nome e rua) e do itinerário traçado (distância e tempo de viagem).

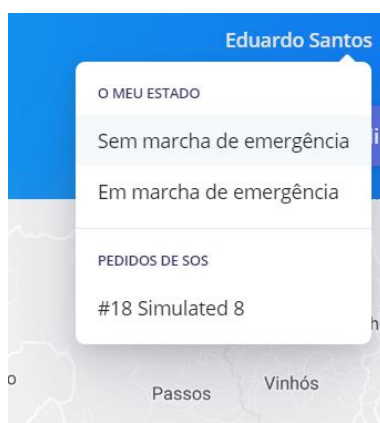


Figura 6.21 – Menu de mudança de estado de veículo de um transporte de marcha de emergência.

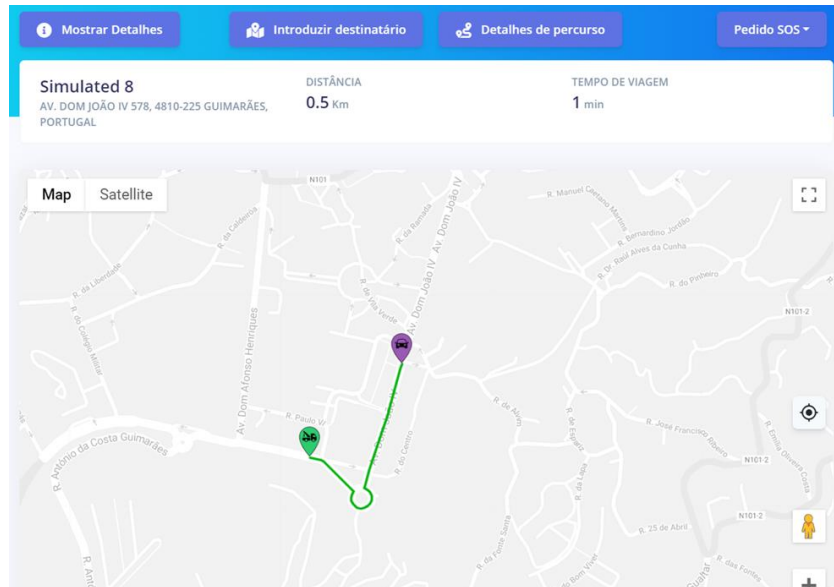


Figura 6.22 – Itinerário de resgate para um reboque.

O diagrama de sequência da mudança de estado encontra-se na Figura 6.23.

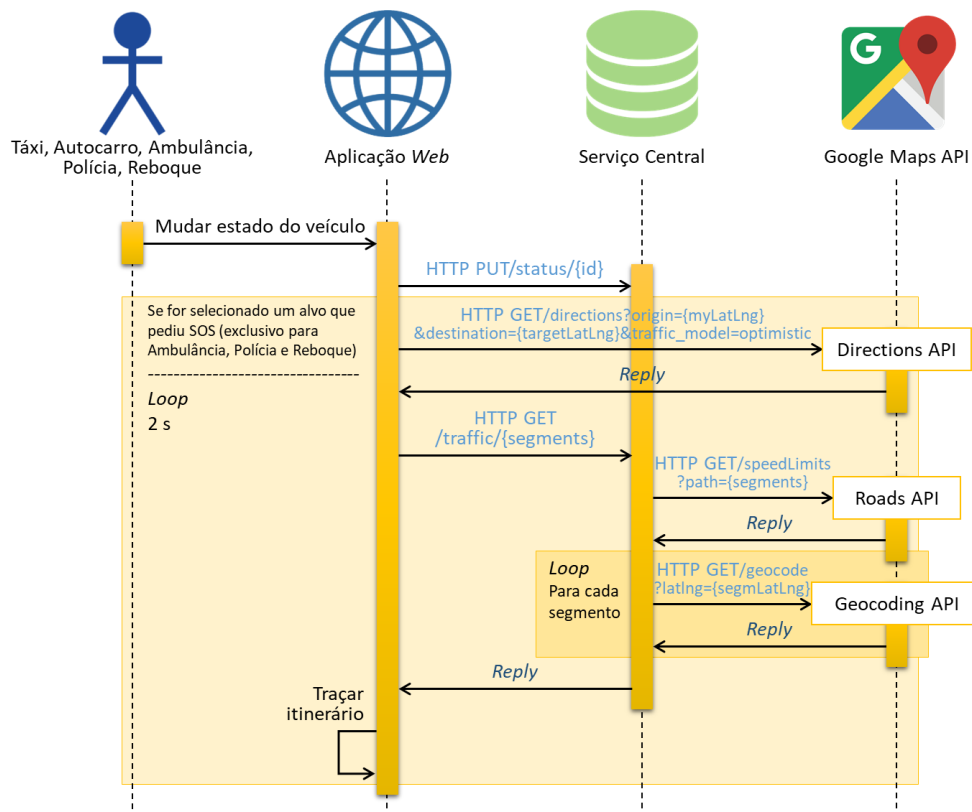


Figura 6.23 – Diagrama de sequência da mudança de estado de veículo.

Quando um utilizador é seleccionado para ser resgatado, esse veículo é também notificado, como o exemplo da Figura 6.24.

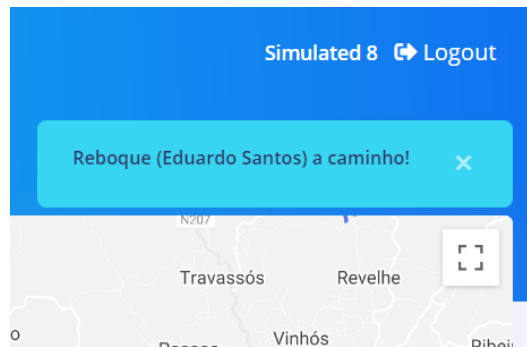


Figura 6.24 – Notificação de reboque a caminho para resgate.

6.7 Excesso de limite de velocidade

Quando um veículo ultrapassa 40 km/h do limite de velocidade da rua em que se encontra, utilizadores do tipo autoritário e polícia são notificados com a notificação da Figura 6.25.

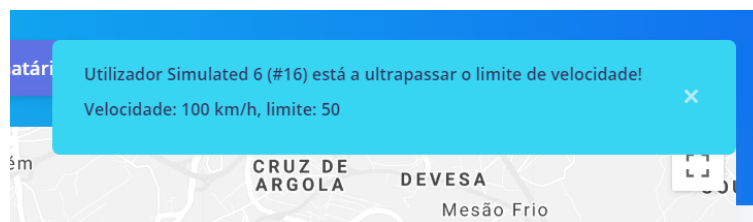


Figura 6.25 – Notificação de deteção de um veículo em excesso de velocidade.

No caso dos polícias, estes podem optar por perseguir o veículo em contraordenação, sendo traçado um itinerário até ao mesmo, como o da Figura 6.26 e é ativada marcha de emergência.

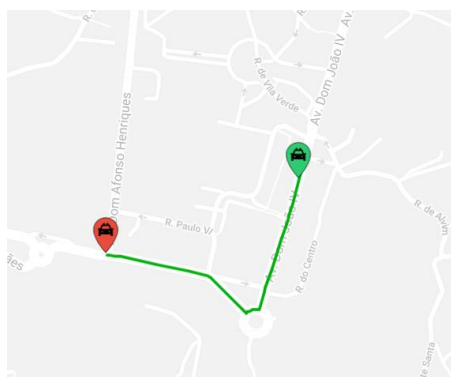


Figura 6.26 – Itinerário de perseguição para um polícia.

6.8 Atribuição de tipos de conta

Como todos os utilizadores são registados com o tipo padrão, a alteração desse tipo apenas pode ser feita por alguém com autoridade.

Apenas utilizadores do tipo autoritário têm acesso ao menu da Figura 6.27 e da Figura 6.28 para alterarem o tipo de conta de utilizadores registados na sua localidade.

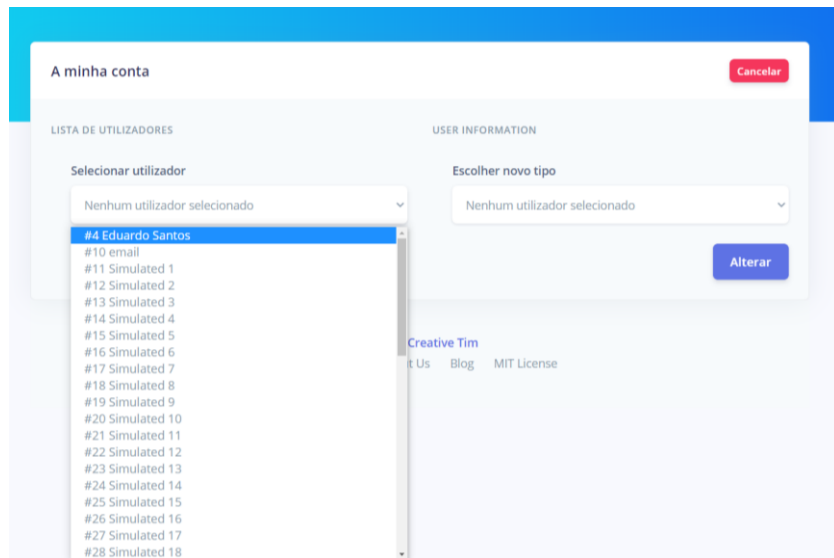


Figura 6.27 – Atribuição de tipo de conta (lista de utilizadores).

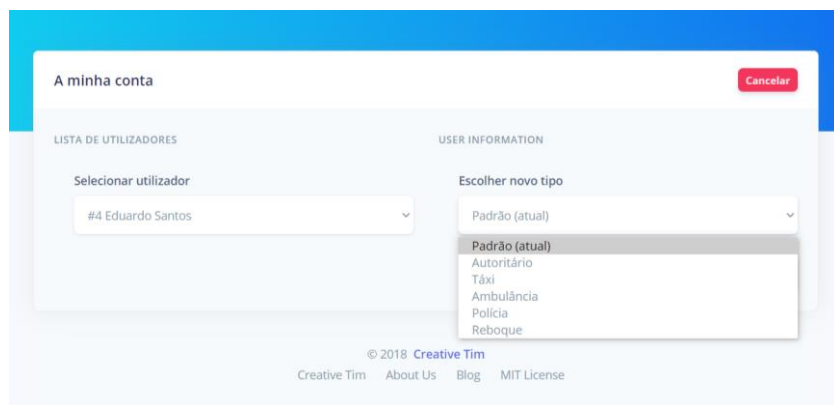


Figura 6.28 – Atribuição de tipo de conta (seleção de tipo).

Na Figura 6.29 encontra-se o diagrama de sequência destas ações.

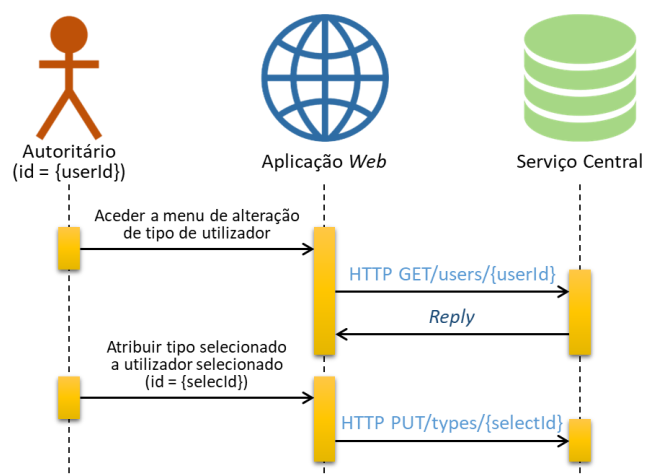


Figura 6.29 – Diagrama de sequência da atribuição do tipo de conta.

6.9 Introdução de destinatário.

Todos os utilizadores podem também pedir para ser traçado um itinerário até um local à escolha, sendo esse itinerário pintado de acordo com o trânsito em cada segmento e revelado nome dessa localidade, a distância desse percurso e do tempo médio de viagem, demonstrado na Figura 6.30 e na Figura 6.31.

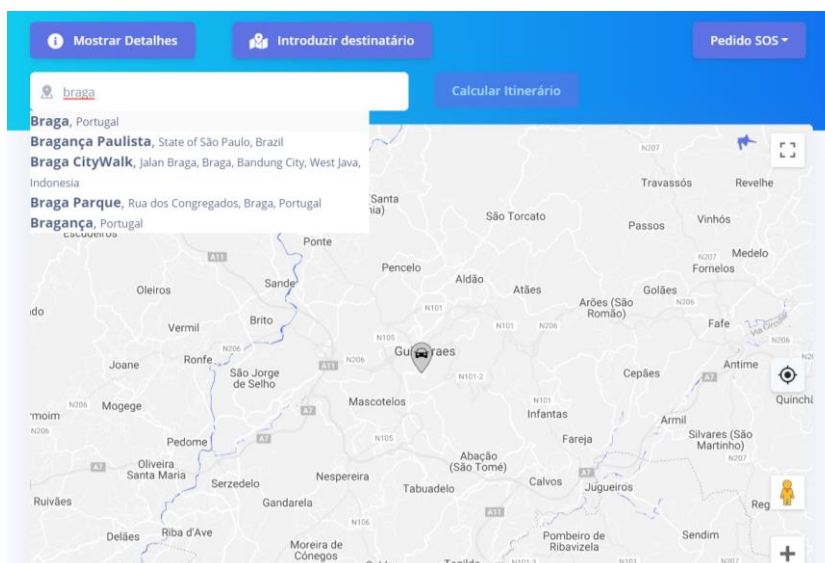


Figura 6.30 – Introdução do destinatário.

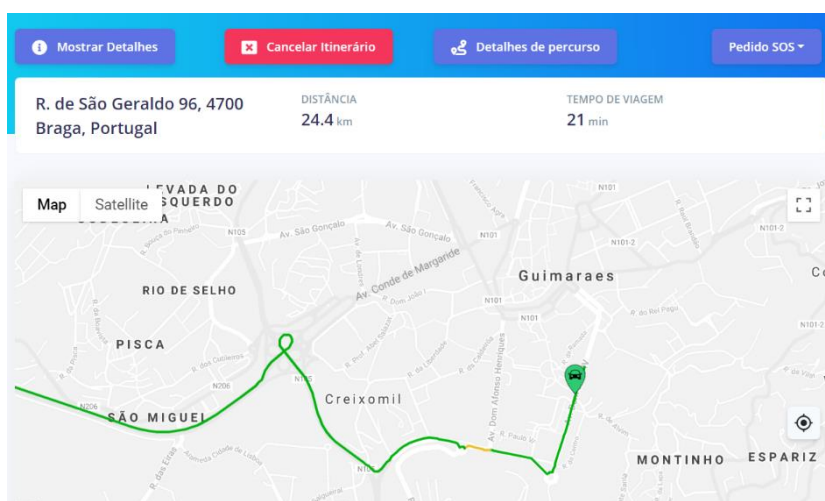


Figura 6.31 – Desenho e informação do itinerário.

O desenho do itinerário e a distância são feitos inteiramente pela Google Maps – Directions API. Contudo a cor de cada segmento depende inteiramente do trânsito calculado pelo Serviço Central.

O pedido do itinerário à Directions API é feito com a opção de trânsito “otimista” (parâmetro *traffic_model = optimistic*) para que o trânsito armazenado na base de dados da Google tenha o mínimo de influência possível. Este pedido também devolve uma duração de viagem com as tais condições “otimistas”. O Serviço Central tem também como

função calcular o atraso total baseado no trânsito desse serviço, sendo este somado à duração obtida pela Directions API para ser apresentado ao utilizador como tempo de viagem.

O diagrama de sequência encontra-se na Figura 6.32.

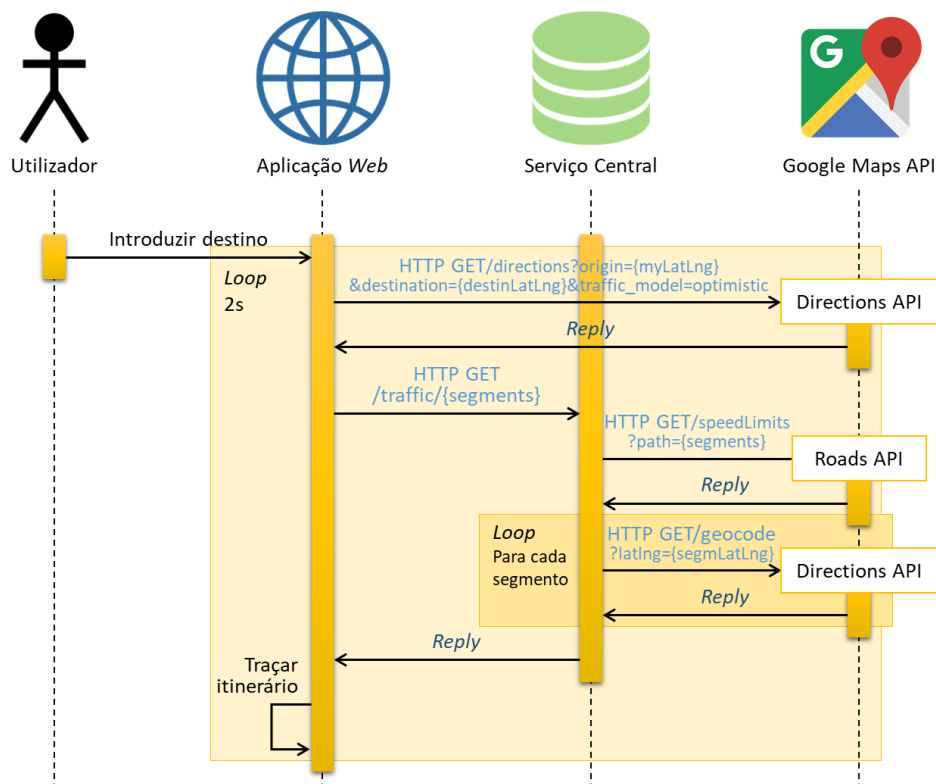


Figura 6.32 – Diagrama de sequência do desenho do itinerário.

O cálculo do nível de trânsito de cada segmento foi explicado na secção anterior 5.4.2.

Este processo termina quando o utilizador se encontra num raio de 20 metros do destino, ou quando o percurso é cancelado manualmente.

7 Software utilizado

No desenvolvimento deste projeto, foram utilizadas diversas ferramentas de *software*, incluindo:

- **Android Studio** [11] (Java) – Componente de aquisição de dados real e serviço de *gateway* (biblioteca usada: WS4D-jCoAP [12]);
- **Visual Studio** [13]:
 - Javascript (Node [14]) – componente de aquisição de dados simulados e respetivo serviço de *gateway*, API do Serviço Central,
 - React [15] – aplicação *web* (*template* usada: Argon Dashboard React [16]),
 - C – *script* de execução de múltiplas simulações de componentes de aquisição de dados e *gateways*;
- **phpMyAdmin** [17] (MySQL) – Monitorização da base de dados do Serviço Central no Raspberry PI;
- **FileZilla** [18] e **PuTTY** [19] – Comunicação e controlo do Serviço Central e da aplicação *web* no Raspberry PI;
- **SSH for Google Cloud Platform** [20] – Comunicação e controlo do Serviço Central e da aplicação *web* na Google Cloud Platform;
- **Microsoft Office PowerPoint** [21] – Apresentação final do projeto e desenho de diagramas e de pinos;
- **Microsoft Office Word** [22] – Escrita da planificação e do relatório final.

8 Outras funcionalidades planeadas

Para o desenvolvimento do projeto, foram também planeadas certas funcionalidades que apesar de não serem necessárias, melhorariam a experiência dos utilizadores e dar-nos-iam mais liberdade na implementação de certas funcionalidades.

8.1 Mapa

No mapa da aplicação *web*, planeamos implementar rotação, ampliação e foco dinâmico na localização do utilizador. Contudo, descobrimos que a sua implementação era muito difícil e não ideal para ambiente *web*. Uma solução para este problema seria desenvolver uma aplicação móvel com as seguintes funcionalidades:

- Foco dinâmico para que o mapa esteja sempre centrado na localização do utilizador;
- Rotação dinâmica para acompanhar a orientação (*bearing*) do utilizador;
- Ampliação dinâmica inversamente proporcional à velocidade instantânea do utilizador – em velocidades elevadas, é importante que a interface abranja uma área maior do mapa.

Esta solução não foi implementada por falta de tempo, contudo tentamos outra solução alternativa em que bastaria o uso da aplicação *web* – em vez de se rodar o mapa de acordo com a orientação dos veículos, rodar-se-ia os pinos dos mesmos. Para tal, necessitar-se-ia de pinos com desenho de veículos em vista de cima, como os pinos da Figura 8.1.

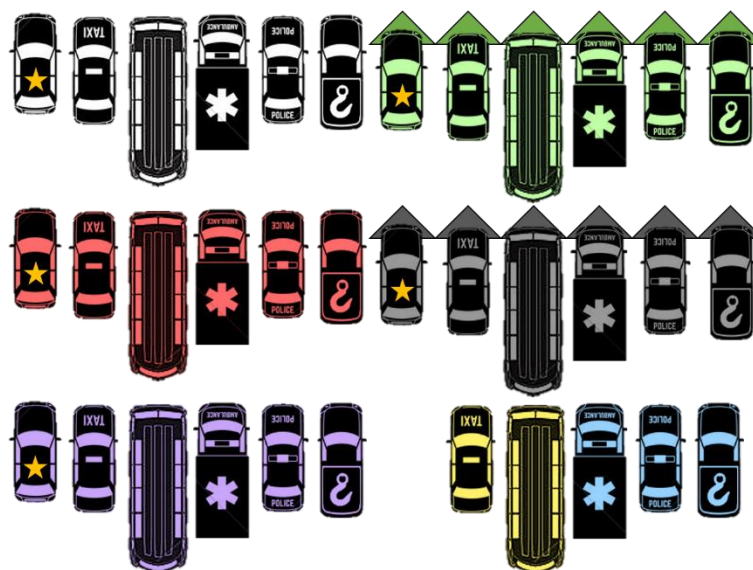


Figura 8.1 – Pinos com vista de cima (não implementados).

Não foram implementados porque não conseguimos implementar rotação nos pinos. Nesta versão dos pinos também pensamos em usar formato GIF nos veículos em marcha de emergência com alternância de duas cores a simbolizar sirenes.

8.2 Itinerários

Para o cálculo e desenho dos itinerários, gostaríamos de ter desenvolvido um algoritmo próprio para tal cálculo, baseado nos dados de trânsito do Serviço Central. Contudo, a Google Maps não disponibilizava nenhuma API com as interseções de ruas (nós) e comprimento das mesmas para que seriam o necessário para se aplicar, por exemplo, o Algoritmo de Dijkstra [23], sendo o número de veículos e o comprimento de cada segmento um fator de custo para o mesmo.

Apesar do desenho de um itinerário não se basear no trânsito, as suas cores baseiam-se.

Uma solução alternativa seria pedir rotas alternativas à Directions API (parâmetro *alternatives = true*) para o destino escolhido para que fosse escolhida a melhor rota. Seria então escolhida a rota com menor soma de duração de viagem (obtida na Directions API) com o atraso total (calculado pelo Sistema Central consoante o trânsito em cada segmento).

Esta solução não foi implementada porque já se faz um número elevado em cada 2 segundos de pedidos às APIs da Google Maps, nomeadamente um pedido à Directions API, um pedido à Roads API e N pedidos à Geocoding API (N = número de segmentos), como está demonstrado na Figura 6.32. Implementar rotas alternativas multiplicaria o número de pedidos pelo número de alternativas e o tamanho dos mesmos ao Serviço Central.

8.3 Componente de aquisição de dados e *gateway*

Nas aplicações Android e *scripts* desenvolvidos para o componente de aquisição de dados e *gateway*, a comunicação entre os mesmos é feita exclusivamente por Wi-Fi/LAN pois só estavam disponíveis bibliotecas para Android Studio com esta tecnologia que suportassem CoAP. Devido a esta limitação, não se implementou suporte para a tecnologia Bluetooth que estava planeada.

8.4 Notificações

No desenvolvimento das notificações, planeou-se também notificações de entrada de veículos em marcha de emergência na rua do utilizador. Esta funcionalidade não foi implementada pois não conseguíamos saber se a entrada desses veículos na rua era à frente ou atrás do utilizador, e saber que um veículo em marcha de emergência entra na rua do utilizador à frente do mesmo é irrelevante pois ambos não se irão cruzar.

9 Conclusões

Neste projeto conseguimos aprimorar conhecimentos aprendidos em UCs de anos anteriores, nomeadamente desenvolvimento de arquiteturas de cliente-servidor, aplicações Android, *web*, APIs e bases de dados, aplicados e aprendidos Sistemas Distribuídos, Paradigmas da Programação II, Laboratórios de Telecomunicações e Informáticas II e Tecnologias de Bases de Dados.

Foram também aprendidos novos conceitos, nomeadamente no desenvolvimento e CoAP, APIs em Node, *web* em React, e no uso do extenso leque de ferramentas disponibilizadas pela Google, nomeadamente Google Maps e Google Cloud Platform.

Achamos que implementamos todas as funcionalidades propostas e que concluímos este projeto com sucesso.

10 Referências

- [1] Coap.technology. (n.d.). *CoAP — Constrained Application Protocol | Overview*. [online] Disponível em: <https://coap.technology> [Acedido a 07/10/2019].
- [2] Android Developers. (n.d.). *LocationManager | Android Developers*. [online] Disponível em: <https://developer.android.com/reference/android/location/LocationManager.html> [Acedido a 20/09/2019].
- [3] Google Developers. (n.d.). *Developer Guide | Directions API | Google Developers*. [online] Disponível em: <https://developers.google.com/maps/documentation/directions/intro> [Acedido a 30/09/2019].
- [4] Ji, Y. (2016). *decode-google-map-polyline*. [online] npm. Disponível em: <https://www.npmjs.com/package/decode-google-map-polyline> [Acedido a 13/11/2019].
- [5] Veness, C. (2013). *Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript*. [online] Movable-type.co.uk. Available at: <https://www.movable-type.co.uk/scripts/latlong.html> [Acedido a 18/10/2019].
- [6] Google Developers. (n.d.). *Speed limits | Roads API | Google Developers*. [online] Disponível em: <https://developers.google.com/maps/documentation/roads/speed-limits> [Acedido a 23/09/2019].
- [7] Google Developers. (n.d.). *Get Started | Geocoding API | Google Developers*. [online] Disponível em: <https://developers.google.com/maps/documentation/geocoding/start> [Acedido a 23/09/2019].
- [8] Raspberry Pi. (n.d.). *Buy a Raspberry Pi 3 Model B+*. [online] Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/> [Acedido a 20/09/2019].
- [9] Google Cloud. (n.d.). *Cloud Computing Services | Google Cloud*. [online] Disponível em: <https://cloud.google.com/> [Acedido a 23/09/2019].
- [10] Google Developers. (n.d.). *Overview | Maps SDK for Android | Google Developers*. [online] Disponível em: <https://developers.google.com/maps/documentation/android-sdk/intro> [Acedido a 23/09/2019].
- [11] Android Developers. (n.d.). *Download Android Studio and SDK tools | Android Developers*. [online] Disponível em: <https://developer.android.com/studio> [Acedido a 20/09/2019].
- [12] Ws4d.org. (n.d.). *Web Services for Devices » Stack: WS4D-jCoAP (Java)*. [online] Available at: <http://ws4d.org/ws4d-jcoap/> [Acedido a 19/10/2019].
- [13] Code.visualstudio.com. (n.d.). *Visual Studio Code - Code Editing. Redefined*. [online] Disponível em: <https://code.visualstudio.com/> [Acedido a 30/09/2019].
- [14] Foundation (n.d.). *Node.js*. [online] Node.js. Disponível em: <https://nodejs.org/en/> [Acedido a 30/09/2019].
- [15] Reactjs.org. (n.d.). *React – A JavaScript library for building user interfaces*. [online] Disponível em: <https://reactjs.org/> [Acedido a 20/09/2019].

- [16] Creative Tim (2019). *Argon Dashboard React by Creative Tim*. [online] Creative-tim.com. Disponível em: <https://www.creative-tim.com/product/argon-dashboard-react> [Acedido a 23/09/2019].
- [17] phpMyAdmin contributors (n.d.). *phpMyAdmin*. [online] phpMyAdmin. Disponível em: <https://www.phpmyadmin.net/> [Acedido a 30/09/2019].
- [18] Filezilla-project.org. (n.d.). *FileZilla - The free FTP solution*. [online] Disponível em: <https://filezilla-project.org/> [Acedido a 23/09/2019].
- [19] Putty.org. (n.d.). *Download PuTTY - a free SSH and telnet client for Windows*. [online] Disponível em: <https://www.putty.org/> [Acedido a 23/09/2019].
- [20] wilford (2015). *SSH for Google Cloud Platform*. [online] Disponível em: <https://chrome.google.com/webstore/detail/ssh-for-google-cloud-plat/ojil-lmhjhbpInppnamldakhpmndnibd> [Acedido a 09/01/2020].
- [21] Products.office.com. (n.d.). *Slide Presentation Software, PPT - Microsoft PowerPoint*. [online] Disponível em: <https://products.office.com/en/powerpoint> [Acedido a 06/12/2019].
- [22] Products.office.com. (n.d.). *Microsoft Word - Word Processing Software / Office*. [online] Disponível em: <https://products.office.com/en/word> [Acedido a 14/10/2019].
- [23] Dijkstra, E. (1959). *A note on two problems in connexion with graphs*. 23rd ed. Numerische Mathematik.