Mobile Tolling System

David Duarte
João Duarte

Orientadores Luís Osório
Paulo Borges

Relatório final realizado no âmbito de Projecto e Seminário,
do Curso de Licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2017/2018

Setembro de 2018

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Mobile Tolling System

Autores:
39374 David Rodrigo Ferreira Rodrigues Duarte
38241 João Pedro Pina Duarte

Orientadores:
Luís Osório
Paulo Borges

Relatório final realizado no âmbito de Projecto e Seminário,
do Curso de Licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2017/2018

Setembro de 2018

Resumo

O projecto enquadra-se na unidade curricular de Projecto e Seminário e visa concluir o percurso académico da Licenciatura de Informática e de Computadores.
Para um indivíduo que pretenda utilizar auto-estradas é possível usufruir do serviço de duas maneiras diferentes. Se já for cliente Via-Verde[1] e possuir o identificador disponibilizado pela mesma, pode usar a portagem automática. Se ainda não for cliente, tem de usar uma portagem manual.
O nosso serviço visa eliminar a necessidade do cliente Via-Verde necessitar de um identificador. Através de uma aplicação móvel, instalada no *smartphone* do cliente, que detecta eventos de passagem nas portagens (entrada/saída) e que os comunica ao Backend.
O Backend para além de expor uma *Web* **API** de forma a responder às necessidades da aplicação móvel, também processa os dados fornecidos pela mesma de forma a aumentar a fiabilidade na detecção de portagens, utilizando um sistema de informação geográfico (**GIS**).

Palavras-chave: Pagamento por geolocalização, **GNSS**, **GIS**, Aplicação móvel, *Web* **API**

Abstract

The project is part of the course *projecto e Seminário* and aims to complete the academic course of the Computer and Computer Science Degree.
For an individual who wanted to use highways it is possible to enjoy the service in two different ways. If you are already a Via-Verde[1] customer and have access to the identifier provided by it, you can use the automatic tolling. If you are not a customer yet, you must use a manual toll.
Our service eliminates the need for the Via-Verde customer to have an identifier, through a mobile application, using the client's personal smartphone.
This mobile application has as main responsibility to detect events of passage in the tolls (entrance / exit) and to communicate these events to the Backend.
Besides exposing a Web API to answer the needs of the mobile app, the Backend also stores and processes the data given by it, enhancing the reliability of the toll detection using a geographic information system (GIS).

Keywords: Geolocation Tolling, GNSS, GIS, Mobile App, Web API

Índice

Lista de Figuras

Conceitos e abreviaturas

Portagem manual
Portagem em que o utilizador tem de parar na cancela para obter o bilhete à entrada ou pagar à saída antes de poder avançar.

OBU
On-Board Unit. É o dispositivo onde actualmente se baseia o sistema de portagens. Serve para identificação do veículo ao passar numa portagem.

Portagem automática
Portagem que obtém automaticamente informação de identificação dos veículos sem que estes precisem de interromper o seu movimento.
Tem-se como exemplo: portagem com câmeras que obtêm foto frontal e traseira da matrícula do veículo e/ou é capaz de comunicar com **OBUs** presentes nos veículos dos clientes.

Clearing
Entidade responsável pelo gerenciamento do sistema de portagens que recebe informação das concessionárias.

ETC
Eletronic Toll Collector. Sistema de portagem móvel ou não, que adquire informação sobre os veículos que passam pela portagem.

Enforcement
Processo que assegura a conformidade da informação adquirida pelo sistema.

GNSS
Global Navigation Satellite System, sistema que através do uso de satélites consegue disponibilizar informação de geolocalização.

Geofencing
Perímetro virtual que engloba uma área geográfica do mundo real, que permite detectar entradas e saídas do mesmo.

DSRC
Dedicated short-range communications. Tecnologia de comunicação sem-fios, localizada na banda de espectro 5,9 GHz, usada actualmente entre **OBUs** e portagens automáticas.

Evento
Um evento define uma interacção entre o utilizador e a sua passagem numa portagem.

Transacção
Informação sobre o veículo e qual o evento de entrada e de saída, quando utilizado nos serviços de portagem da Via-Verde.

Organização do documento

Este documento encontra-se organizado da seguinte forma.

Capítulos

O relatório encontra-se dividido por capítulos, sendo estes:

- Capítulo 1, Introdução - Introdução ao projecto e respectivos objectivos.
- Capítulo 2, Formulação do Problema - Detalha os problemas que serão ultrapassados;
- Capítulo 3, Soluções Propostas - São expostas as soluções e como foram implementadas;
- Capítulo 4, Resultados e Conclusões - São apresentados os resultados observados tais como as conclusões e propostas de trabalho futuro para o melhoramento do sistema.

Convenções Tipográficas

Apresentam-se seguidamente as convenções tipográficas empregues na escrita do relatório de Projeto e Seminário:

- Em todo o documento a escrita está conforme o antigo acordo ortográfico;
- Emprega-se no texto normal a fonte *Times New Roman*, tamanho 11;
- Emprega-se nos títulos a fonte *Times New Roman*, tamanho 20;
- Emprega-se nos subtítulos a fonte *Times New Roman*, tamanho 13;
- As figuras presentes no relatório são identificadas com a fonte *Times New Roman*, tamanho 9;
- O espaçamento utilizado entre linhas/parágrafos é de 1,15;
- Empregam-se [parêntesis retos] para referências bibliográficas;
- Todos os estrangeirismos estão destacados em itálico;
- As abreviaturas são destacadas a negrito.

Capítulo 1 - Introdução

1.1 Enquadramento

Actualmente os dispositivos móveis têm-se vindo a tornar cada vez mais populares, pouco dispendiosos e com uma oferta variada de funcionalidades, sendo uma delas a geolocalização. Tirando partido de que os possíveis utilizadores, muito provavelmente, possuem um destes dispositivos e que o mesmo oferece geolocalização, pretende-se desenvolver um sistema informático que visa facilitar o uso dos serviços de mobilidade disponibilizados pela Via-Verde, mais especificamente, o serviço de portagens. Modernizando o mesmo e enriquecendo a experiência de utilização de infra-estruturas de transporte e mobilidade já existente.

Os clientes Via-Verde podem usufruir das auto-estradas sem necessidade de paragem nas portagens, através do uso de um identificador **OBU**(*On Board Unit*). Tendo em conta que é necessário este identificador estar instalado no veículo e existindo um custo e tempo de registo associado ao mesmo, identificou-se que existe uma oportunidade para melhoria do serviço, assumindo que o utilizador possui um *smartphone* que fornece geolocalização. A proposta do **MTS** oferece a possibilidade de usufruir do serviço de portagens Via-Verde, facilitando a sua experiência aos clientes que não têm acesso directo ao **OBU** ou que pretendam uma alternativa ao mesmo.

1.2 Objectivos

De modo a oferecer este serviço, desenvolveu-se uma aplicação móvel capaz de detectar a passagem em portagens e comunicar essa informação a um servidor que eventualmente a faz chegar à Via-Verde. Este servidor irá receber os eventos de passagem da aplicação móvel de cada cliente, processá-los e armazená-los, até o *clearing* os validar e eventualmente ser realizado o débito do serviço. Nessa altura, o servidor receberá confirmação da informação e do respectivo débito, sendo esta comunicada com o dispositivo móvel possibilitando o utilizador de a visualizar.

Devido à possibilidade de poder existir erro na detecção de portagens, o utilizador poderá, caso seja necessário, corrigir as portagens que percorreu. O uso indevido desta funcionalidade incorre eventualmente em punição com possíveis multas.

Capítulo 2 - Formulação do problema

2.1 Evolução

Os sistemas de portagens rodoviárias começaram por instituir portagens manuais em que todos os veículos tinham de parar para tirar o bilhete e posteriormente pagar no final da viagem sendo o enforcement utilizado, apenas a barreira da portagem.
Este tipo de solução trazia vários problemas tais como o congestionamento, devido à paragem obrigatória de todos os veículos para registar a sua entrada ou efectuar o pagamento.
Surgiu então uma nova solução que evitava este tipo de problemas. Essa solução foi a **ORT** *(Open Road Tolling)*[2]. A **ORT** introduziu um novo componente ao sistema chamado de **OBU** que iria estar presente em cada veículo de cada cliente e comunicar através de uma antena com a portagem tirando partido de comunicação sem-fios **DSRC**, eliminado assim a necessidade de barreiras de portagem.

2.2 Estado da arte

Existem variados tipos diferentes de *tolling* já implementados nas estradas em todo o globo. O modelo mais usado é a combinação entre **OBU** com comunicação por rádio com uma barreira de portagem, usando como enforcement vários métodos i.e. reconhecimento de imagem de matrícula, **GNSS** através do **OBU**. Algumas das nações de interesse são:

Suíça[3] - É utilizado um sistema com um **OBU** com capacidades de geolocalização e detecção de movimento, conectado ao odómetro do veículo para ajudar a evitar fraudes;

Austrália[4] - É oferecida a oportunidade de utilizar portagens sem a necessidade de um OBU, tirando partido de uma aplicação móvel (LinktGo) que utiliza a funcionalidade de geolocalização do dispositivo móvel;

Portugal[1] - O modelo usado actualmente em Portugal para open-road tolling é um sistema cooperativo entre um dispositivo instalado no veículo (**OBU**) e uma barreira de portagem que contém uma antena, capaz de comunicar com o anterior através de **DSRC** um meio de comunicação rádio (5.8 GHz), também como câmeras que obtêm uma foto frontal e traseira da matrícula do veículo.

2.3 O nosso caso de utilização

A abordagem assemelha-se à proposta utilizada na Austrália pela *LinktGo*[4]. Este serviço vai focar-se numa aplicação móvel que tira partido das funcionalidades de GNSS, existentes no próprio dispositivo que detectada quando o utilizador passa por plazas de portagem, identificando os percursos que efectuou. Comunicando com um serviço de *leasing* neste caso a Via-Verde que recebe confirmação dos percursos efectuados. Mais tarde é comunicado por parte do *leasing* o estado do pagamento, sendo que a nossa aplicação não se responsabiliza por essa componente; apenas disponibiliza informação sobre a mesma.

2.3 Requisitos do sistema

Para que o sistema funcione como um todo, concluímos que seria necessário o sistema implementar os seguintes requisitos:

2.3.1 Requisitos funcionais

- Registo e autenticação de um utilizador no sistema (encaminhar registo da Via-Verde para o serviço apropriado);
- detecção de entrada e saída de portagens;
- Visualização da localização actual do veículo e o percurso que este está a efectuar;
- Serviço de histórico da utilização de veículos, incluindo percursos efectuados;

2.3.2 Requisitos não funcionais

- Garantir resistência a falhas por parte de detecção da geolocalização;
- Garantir as principais funcionalidades quando o dispositivo se encontra offline.

Capítulo 3 - Soluções Propostas

3.1 Visão Geral do Problema

O Mobile Tolling System (**MTS**) vai ser enquadrado no contexto da Via-Verde com o intuito de reforçar os seus serviços. Haverá partilha de informação de passagem em portagens com o sistema de *Clearing* da Via-Verde.

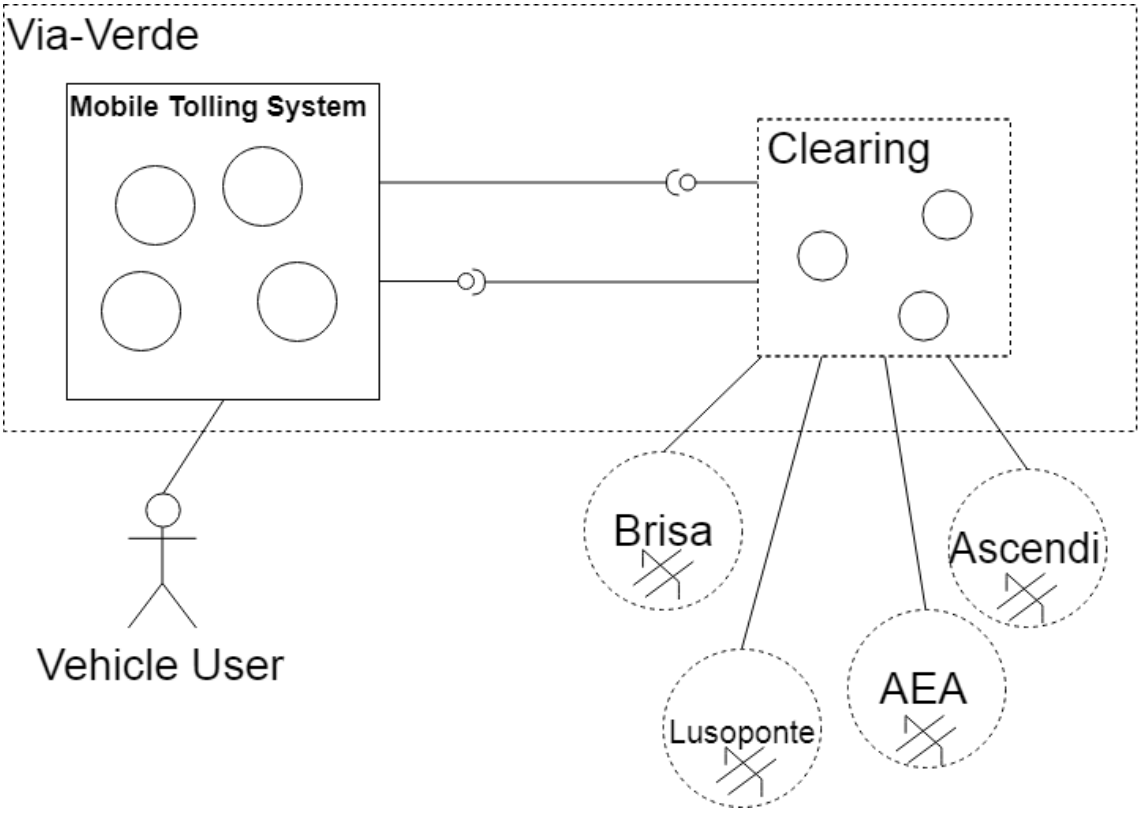


Figura 1 - Diagrama geral do sistema

Este adquire também informação de passagem em portagens por parte dos **ETCs** das concessionárias. Neste caso, a Brisa, Lusoponte, Auto-Estradas do Atlântico (**AEA**) e Ascendi. Estas formam o aglomerado de entidades que fornece informação ao Clearing, permitindo validar percursos efectuados e cobrança dos valores associados pelos utilizadores Via-Verde.

3.2 Casos de Utilização

Neste diagrama pode-se observar as relações entre os seguintes atores e os casos de utilização do sistema.

Um *Vehicle User* representa qualquer pessoa com acesso à aplicação móvel. Após a mesma efectuar o registo na Via-Verde e concordar com os termos de uso do serviço **MTS**, é considerada utilizadora do serviço e pode usufruir de todas as funcionalidades indicadas na Figura 2.

O *Clearing* representa o sistema da Via-Verde, do qual o **MTS** não é responsável.

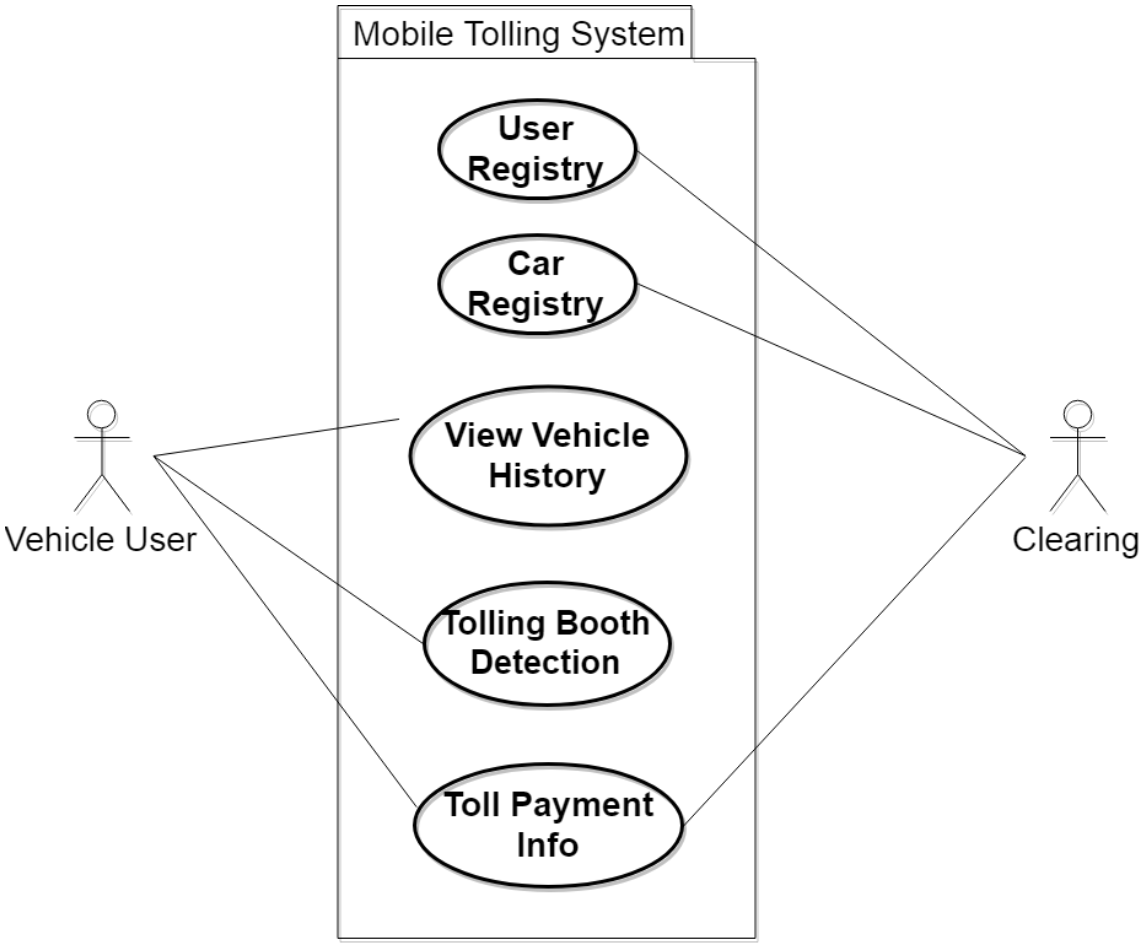


Figura 2 - Diagrama de Use-Case.

3.2.1 Funcionalidades expostas ao condutor

Utilizando a aplicação móvel disponibilizada, o condutor é capaz de ver o seu histórico de viagens, receber notificações, verificar pagamentos e iniciar viagens, designado pelo caso de utilização "*Tolling Booth Detection*" na figura 2.

3.3 Detecção de passagem em portagens

O ponto fulcral do sistema é a detecção de passagem em portagens.

Como já foi referido anteriormente, o dispositivo móvel do utilizador detecta a sua geolocalização com o fim de comparar a mesma com a das portagens.

3.3.1 Geofencing

Para responder ao problema de detecção de portagens é usado o serviço de *Geofence*[5] da Google, para a plataforma Android, permitindo representar regiões geográficas, designadas por *geofences*.

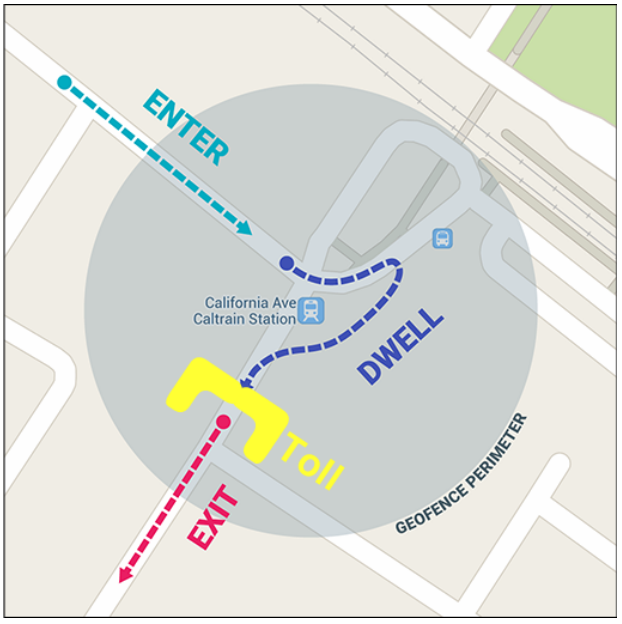


Figura 3 - Esquema do geofence

Esta tecnologia permite monitorizar a geolocalização do dispositivo e gerar eventos quando é detectado que o mesmo ultrapassa a fronteira do *geofence*, seja uma entrada ou saída do perímetro. Como ilustrado na figura 3, ao associar um geofence por portagem e mediante um evento de entrada são guardados os pontos de geolocalização dentro do *geofence* até existir o evento de saída, possibilitando saber se o percurso efectuado passa pela portagem.

Problemas associados ao *Geofencing*

- Possibilidade de erro em zonas não urbanas quando o telemóvel não dispõe de **GPS** nativo;
- Limitação de registo de 100 *geofences* por aplicação por dispositivo.

3.3.2 Solução ao limite de *geofences*

Devido ao limite de 100 geofences por dispositivo, imposto pela Google, foi necessário encontrar uma solução para apenas registar geofences para as portagens de maior interesse. Neste caso, as mais próximas do condutor quando este escolhe usufruir da funcionalidade de detecção de portagens do sistema.

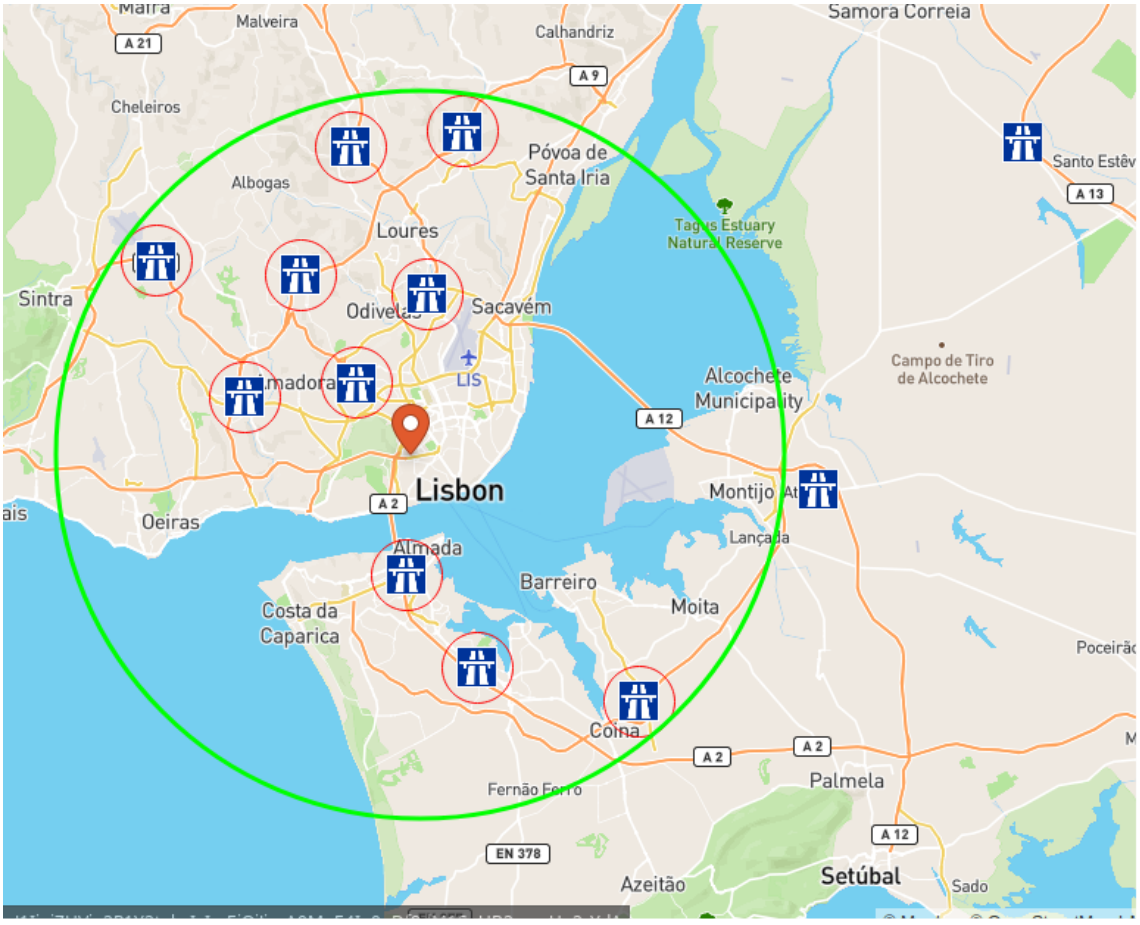


Figura 4 - Ilustração da solução para *geofencing*

A solução encontrada foi registar um *macro-geofence,* a verde na figura 4, que engloba as 90 portagens mais próximas do condutor, 10 na figura para facilitar ilustração. Ao receber o evento de saída do *macro-geofence,* sabe-se que existe a necessidade de registar de novo as portagens mais próximas. É adquirida a geolocalização do condutor e registadas as novas portagens.
É adicionado um erro de 20 km ao raio, de forma a impedir que exista o caso extremo em que no momento de saída do *macro-geocence*, o utilizador passe por uma portagem antes que esta seja registada.

3.4 Métodos de *Enforcement*

Uma das lacunas do sistema é a precisão imprevisível do **GPS** do dispositivo móvel.
Para mitigar este problema foi implementado um método de *enforcement* que coleciona os pontos geográficos durante a passagem numa portagem.
A aplicação móvel reúne os pontos geográficos e a sua direcção geográfica (azimute) e estes são posteriormente enviados ao Backend, com informação da direcção que o utilizador percorreu.

3.4.1 Intersecção entre geolocalização e área geográfica junto à portagem

Esta solução verifica se os pontos colecionados se encontram dentro de um perímetro estipulado para a portagem respectiva.
A funcionalidade de interseção de pontos geográficos e os polígonos de uma portagem será explicada em mais detalhe na secção 3.6.5, Extensão PostGIS.

3.4.2 Verificação da direcção no acto de passagem na portagem

Cada portagem contém um azimute, que representa a direção geográfica da entrada de uma portagem.
O azimute dos pontos coleccionados é comparado com o azimute da portagem, sendo possível verificar a direcção que o utilizador passou na portagem.
Esta funcionalidade é fulcral quando se trata de portagens de passagem, porque tipicamente existe uma portagem (de um só sentido), onde a estrada do sentido oposto se encontra na vizinhança, sendo difícil por vezes verificar a passagem correcta do utilizador.

3.4 Arquitectura da Solução

O ***MTS***, divide-se em duas principais componentes. A componente móvel, Mobile Application na figura 5, que se encarrega de toda a interacção com o condutor, da interface de utilização do sistema e da detecção de passagem em portagens.

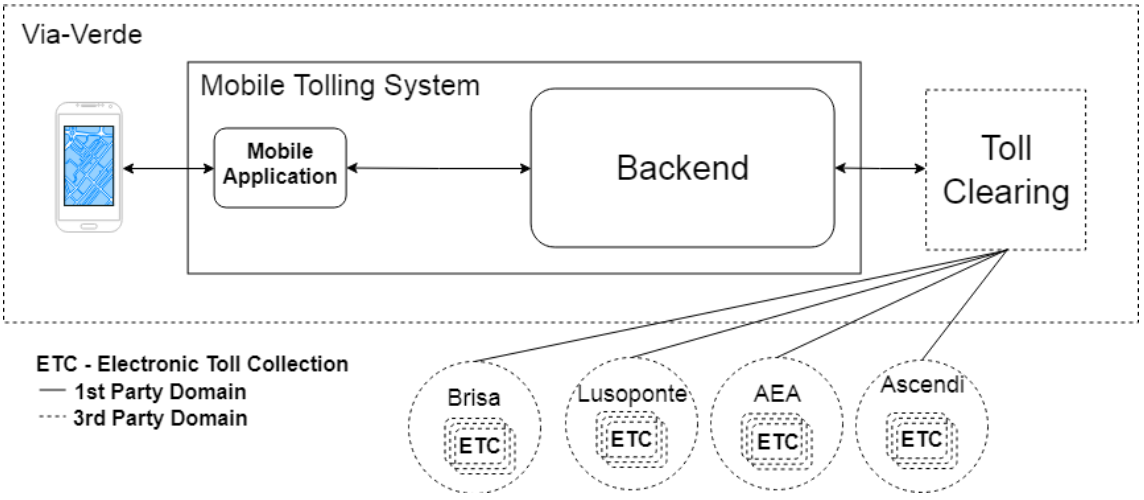


Figura 5 - Diagrama geral da arquitectura

A componente *Backend*, encarrega-se de oferecer informação relevante para a componente móvel, tal como a localização das portagens mais próximas, quais os veículos que o utilizador tem disponível e verificar se a informação de passagem numa portagem é válida. Este recebe informação de utilização de portagens por parte do condutor e envia-a para o sistema de *Toll Clearing,* para que este o valide e inicie o processo de cobrança, qual não é responsabilidade do ***MTS***.

3.5 Arquitectura da Aplicação Móvel

A aplicação móvel é implementada em *Kotlin*[6], tirando partido das mais recentes componentes de Arquitectura Android introduzidas com o *Jetpack*[7] e tendo em conta as boas práticas recomendadas pela Google, tais como implementação do padrão de desenho *Model View Presenter* (**MVP**)[8] e de injecção de dependências.

3.5.1 Interface Gráfica

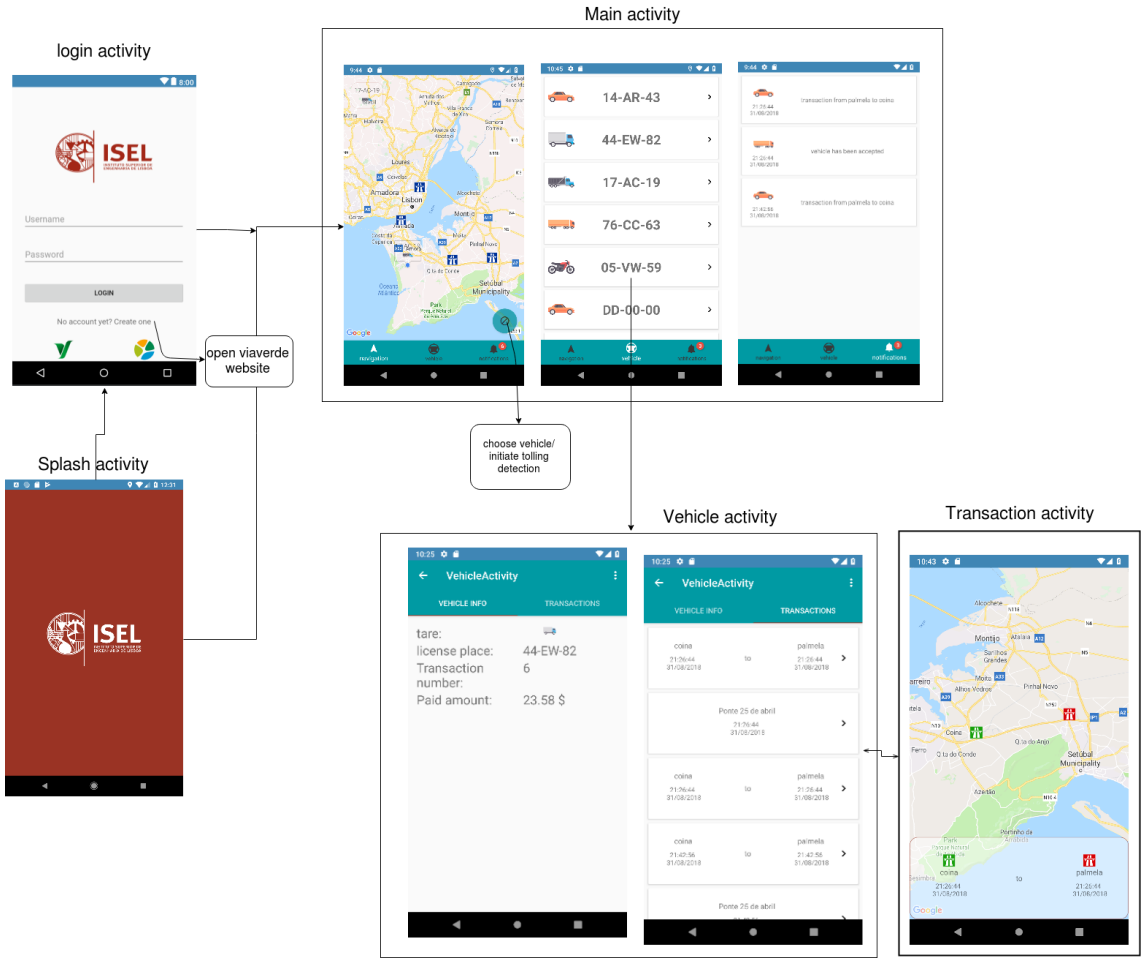


Figura 6 - Diagrama da interface gráfica da aplicação

Nesta figura encontra-se a organização e navegação na interface gráfica e modelo da apresentação de informação disponível ao utilizador, tais como as acções possíveis sobre a mesma. Na figura 6, é ilustrado o grafo de navegação e caminhos possíveis dentro da aplicação móvel do **MTS**.

Figura 7 - Vista Splash

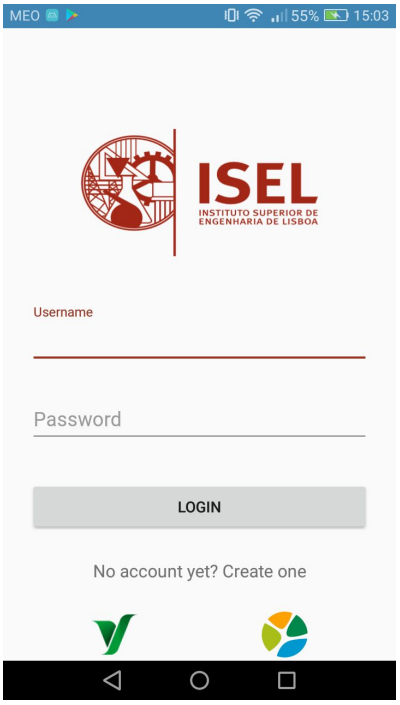


Figura 8 - Vista Login

Ponto de Entrada

O ponto de entrada na aplicação, primeira vista a ser mostrada, é a *Splash Activity,* apresentada *na figura 7.* Esta está presente enquanto a aplicação carrega e verifica se o utilizador se encontra com autenticação feita. Caso se encontre autenticado, é redireccionado para a vista principal*, apresentada na figura 9.* Caso contrário, é redireccionado para a vista de *login*, apresentada na figura 8.

A *Splash Activity* deve estar o menor tempo possível presente, visto que impede o utilizador de interagir com a aplicação.

Vista de *Login*

Nesta vista é apresentada a funcionalidade de autenticação. O utilizador pode inserir as credenciais da sua conta Via-Verde, caso sejam válidas são guardadas para o utilizador não ter de voltar a autenticar-se e este é redireccionado para a vista principal, figura 9, caso sejam inválidas, são requisitadas de novo visto não ser possível utilizar a aplicação sem uma sessão válida.

No caso de o utilizador não ter conta, este pode pressionar na opção de registar conta. O mesmo é redireccionado para o site da Via-Verde onde poderá registar uma nova conta, visto que o registo não é da nossa responsabilidade. Após ter a conta validada pode fazer autenticação na nossa aplicação.

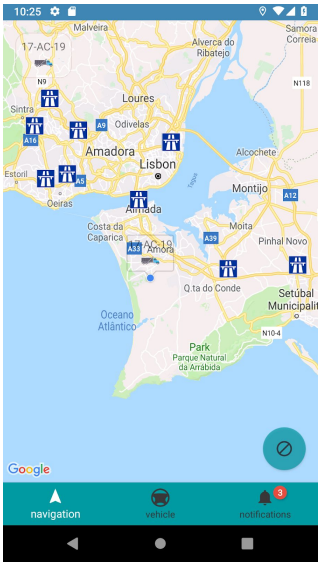


Figura 9
Vista principal-veículos

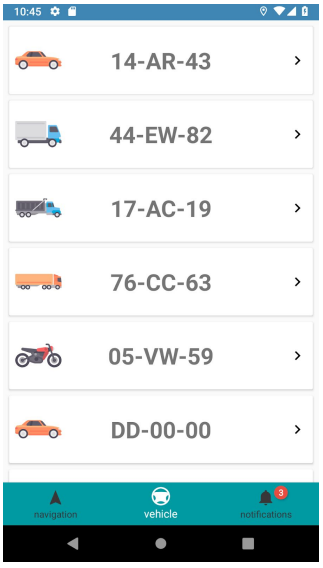


Figura 10
Vista principal-veículos

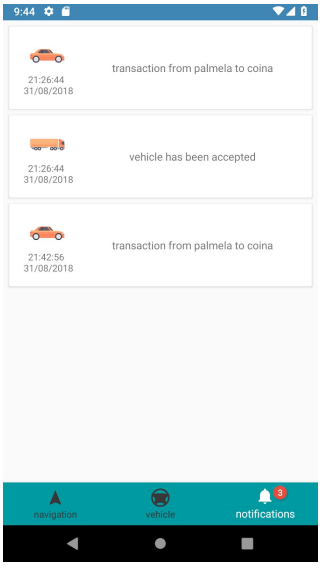


Figura 11
Vista principal-notificações

Vista Principal

Esta é a vista principal da aplicação, *Main Activity,* possibilita acesso rápido às três funcionalidades principais:

- Navegação, figura 9, acesso a detalhes de navegação com detecção de portagens, é a predefinida das três possíveis, visto ser onde o utilizador pode activar a detecção de portagens e iniciar a sua viagem sem ter de se preocupar mais com navegação na aplicação;
- Veículos, figura 10, mostra a lista de veículos que o utilizador tem disponível, e disponibiliza acesso à vista de veículo, *Vehicle Activity*;
- Notificações, figura 11, mostra a lista de notificação por confirmar leitura, disponibiliza acesso a uma vista de detalhe da notificação com respectivas acções.

Navegação

A navegação é a funcionalidade principal da aplicação, disponibiliza acesso a iniciar detecção de portagens e a escolher o veículo a usar nas mesmas através de um *floating action button(fab*), presente no canto inferior direito da figura 9. Auxilia o uso da aplicação através de um mapa, utilizando o *google maps*[9], onde é possível visualizar a localização actual, o veículo selecionado tal como as portagens mais próximas e em caso de já se encontrar numa autoestrada qual a portagem que foi detectada.
É ainda possível saber detalhes e corrigir detecções de portagem pressionando a portagem no mapa e escolhendo uma opção de correcção.
Na situação em que o utilizador se encontra previamente autenticado, o início de detecção de portagens são apenas três passos, abrir a aplicação, pressionar o *fab* e escolher o veículo a utilizar, proporcionando assim uma experiência de utilização rápida e agradável.

Veículos

Nesta vista é possível visualizar a lista de veículos registados na aplicação, e navegar para a vista de detalhes dos mesmos pressionando um deles.

Notificações

Nas notificações é possível visualizar a lista de notificações por confirmar leitura, estas têm vários tipos, como por exemplo notificação de novo veículo disponível, passagem em portagens detectadas, ou notificação de pagamento efectuado. O pressionamento em uma das notificações permite ao utilizador confirmar leitura, ou outras acções de interesse sobre a mesma.

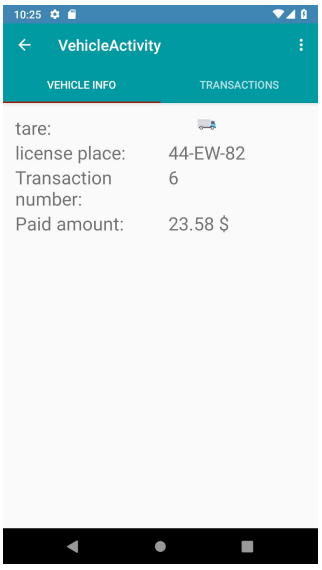


Figura 12
vista de veículo - detalhe

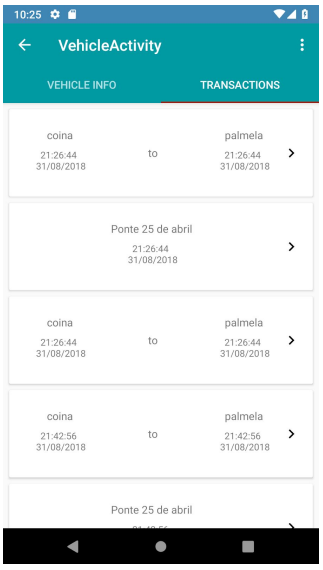


Figura 13
vista de veículo - transacções

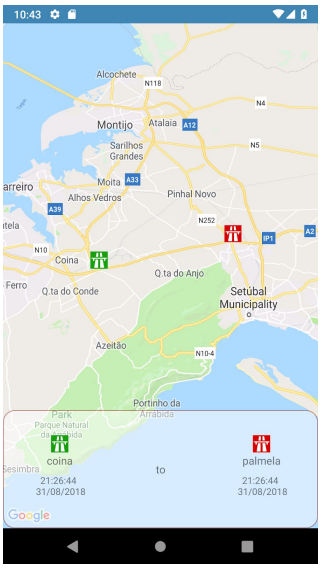


Figura 14
vista de detalhe de transacções

Vista de Veículo

Nesta vista é possível aceder a duas funcionalidades principais sobre o veículo escolhido:

- Informação detalhada do veículo;
- Colecção das passagens em portagens efectuadas com o veículo em questão.

Detalhe de veículo

Nesta vista, presente na figura 12, é possível visualizar informações detalhadas do veículo tais como a classe, a matrícula, número de transacções efectuadas e o dinheiro gasto com as mesmas.

Transacções de Veículo

Nesta vista, presente na figura 13, é possível visualizar as passagens em portagens efectuadas com o veículo ordenadas por data de forma decrescente, mostra a data, quais as portagens utilizadas, ao pressionar um dos elementos da lista o utilizador é redireccionado para uma a vista de detalhe de transacção, figura 14.

Vista de detalhe da transacção

Presente na figura 14, no detalhe de transacção é possível visualizar o montante pago, o veículo e a posição geográfica das portagens utilizadas na transacção, sendo que o utilizador as pode pressionar numa das portagens para fazer *zoom* na sua localização.

3.5.2 Lógica aplicacional

A aplicação móvel tem certos problemas de maior complexidade a serem resolvidos, dos quais a sincronização de dados com o backend, a detecção de portagens e a possibilidade de dar suporte em modo offline, foram desenvolvidas as seguintes soluções para os mesmos.

Funcionamento *offline*

Visto que a ligação pode não estar disponível durante uma viagem em que o utilizador esteja a tirar partido do nosso serviço, a solução é agendar a verificação das portagens por onde ele passou tal como restantes comunicações com o *backend* para quando o dispositivo móvel tenha conexão.

Detecção de geofences

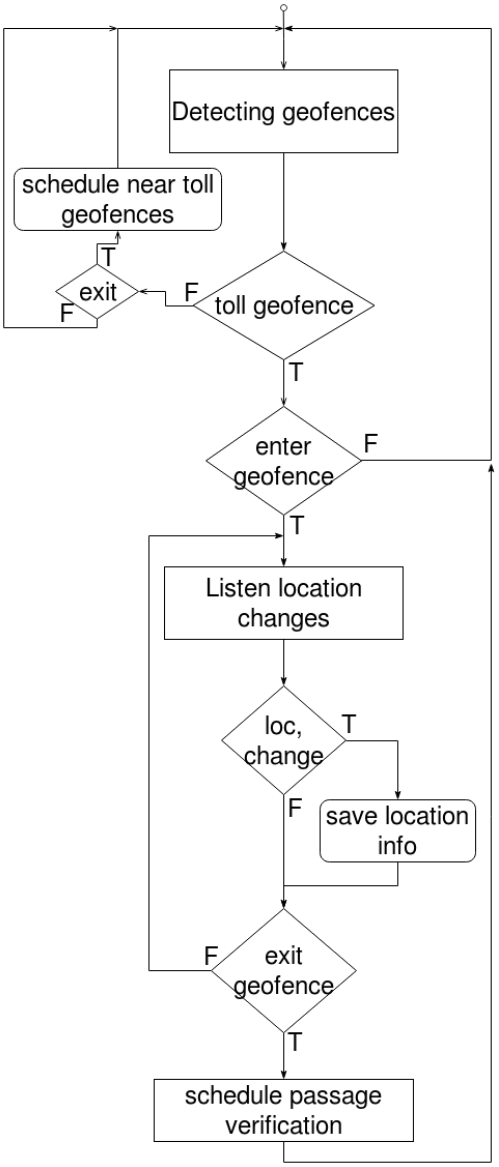


Figura 15 - Diagrama de fluxo detecção de portagens

A detecção de *geofences* é iniciada quando o utilizador escolhe um veículo para utilizar na sua viagem. É registado um *geofence* por portagem com um raio de 150 metros, valor recomendado pela Google, e um com o raio que engloba todas as portagens activas, permite-se assim detectar quando o dispositivo móvel se aproxima de uma portagem ou quando este se afasta das portagens activas. Quando detectado a entrada num dos *geofences* é seguida a lógica do diagrama de fluxo apresentado na figura 15, caso seja um *geofence* de uma portagem se for de entrada é iniciado um serviço que recebe as mudanças de localização do dispositivo, de forma a saber o trajecto percorrido pelo mesmo dentro do *geofence*, quando é detectada a saída é agendado um serviço de verificação de passagem na portagem com o backend. Caso o *geofence* seja de saída do perímetro de portagens activas, é reagendado uma activação das portagens mais próxima.

Verificação de passagem em portagens

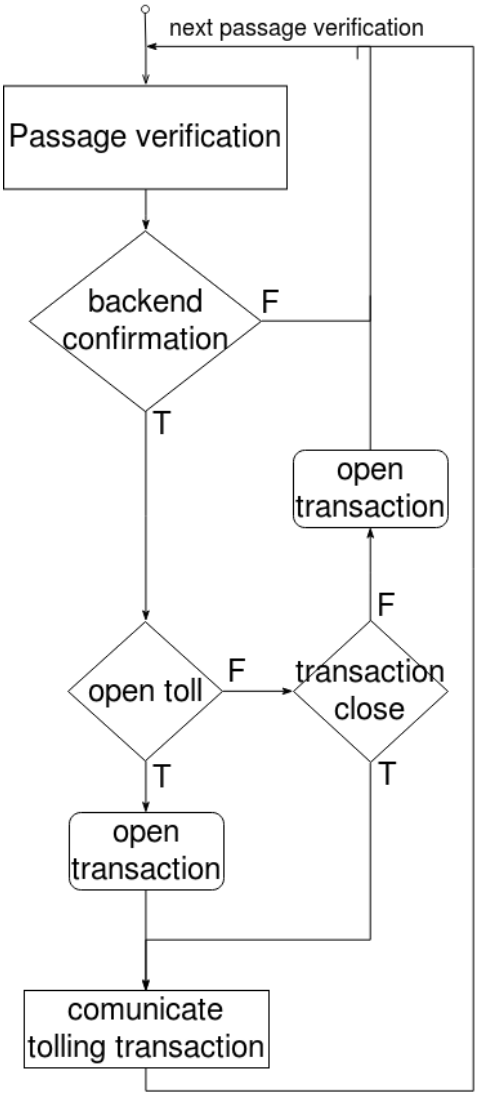


Figura 16 - Diagrama de fluxo verificação de passagem

A verificação de passagem em portagens é iniciada quando o serviço agendado é iniciado pelo sistema, neste caso apenas quando existe conexão. Como ilustrado no diagrama da figura 16 é enviado para o backend a informação da portagem e dos pontos de geolocalização de passagem pela mesma, incluindo não só as coordenadas como também a orientação e a precisão dos mesmos.
Caso seja passagem seja válida é verificado se a portagem é uma portagem aberta ou se esta passagem fecha uma transacção, se sim é comunicado com o backend a mesma, caso contrário é aberta uma transacção que será fechada pela próxima passagem.

Sincronização de dados

Para a sincronização de dados a solução escolhida foi a de tirar partido de um dos novos componentes de Arquitectura do Android, neste caso do *WorkManager*[10], este possibilita o agendamento de tarefas com várias restrições, tais como só executar caso exista conexão à internet, ou o dispositivo não esteja com pouca bateria, tal como correr periodicamente.
É agendada uma tarefa que executa quatro vezes por dia a sincronizar dados do *backend*, tais como novos veículos ou pagamentos de transacções. Para permitir funcionamento *offline* todas as comunicações que deveriam ter sido feitas quando o dispositivo se encontrava *offline* são feitas assim que exista conexão à internet, existindo uma tarefa para cada uma.

3.5.3 Arquitectura interna

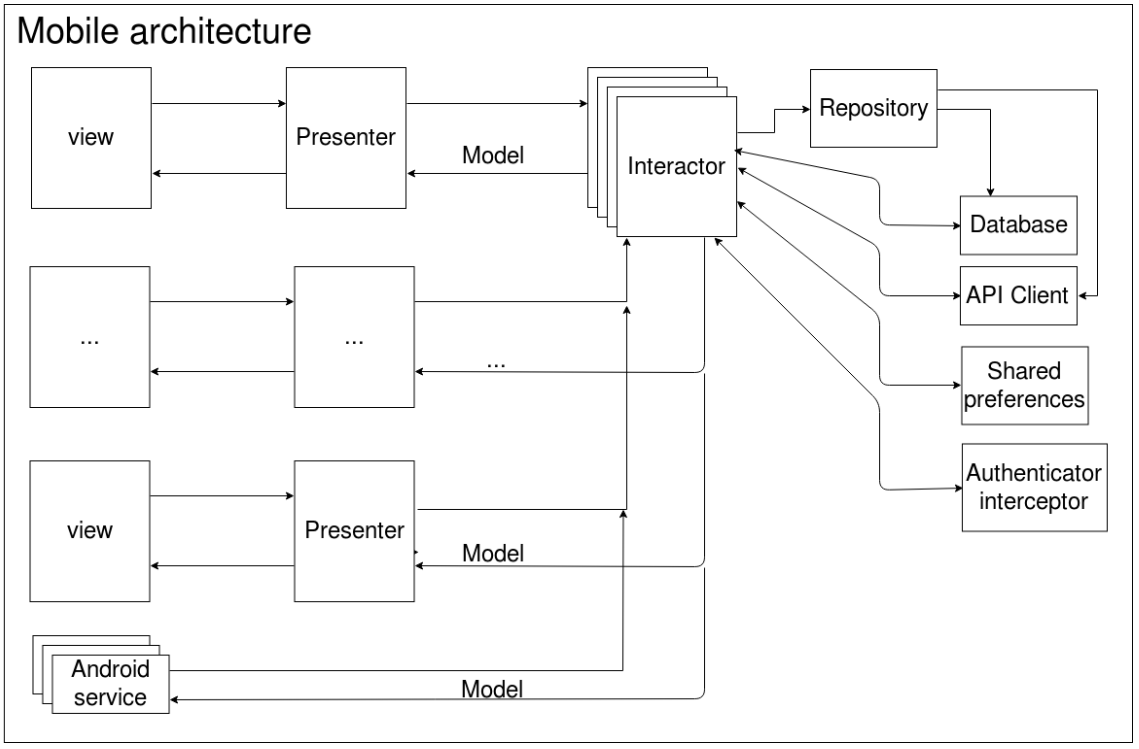


Figura 17 - Diagrama implementação **MVP**

Neste diagrama é possível observar a organização interna da aplicação que segue o modelo de arquitectura *Model View Presenter*[7] ***(MVP)***, que visa desacoplar a vista, *View*, do acesso a dados, *Model*. Segue o princípio de desenho da separação de responsabilidades, facilitando não só o desenvolvimento, como a testabilidade do código.

Estes benefícios são possíveis devido a um novo intermediário, *Presenter*, que se encarrega de receber pedidos da *view* e implementar toda a lógica de apresentação relacionada com a mesma, invocando funções existentes na *view*, para que esta altere a interface gráfica adequadamente.

É ainda utilizado uma abstracção sobre os dados, denominada de *Interactor*. Um *Interactor* representa um conjunto de *use cases* que são representados por um conjunto de acções sobre os dados que geralmente é comum em vários *presenter,* este está encarregue de aceder a serviços tais como a base de dados ou fazer pedidos a serviços externos por exemplo uma **API** *web*, deve ser reutilizado por múltiplos *presenters* ou até serviços do Android que não precisam de lógica de apresentação, desta forma não é repetido código de lógica de acesso a dados, facilitando a sua reutilização e crescimento da aplicação.

Injecção de dependências

Visto que existe um nível significativo de dependências entre componentes, foi escolhido introduzir um outro padrão denominado de injecção de dependências, que desacopla os componentes das suas dependências, tirando partido a biblioteca Dagger[11] disponibilizada pelo Google.

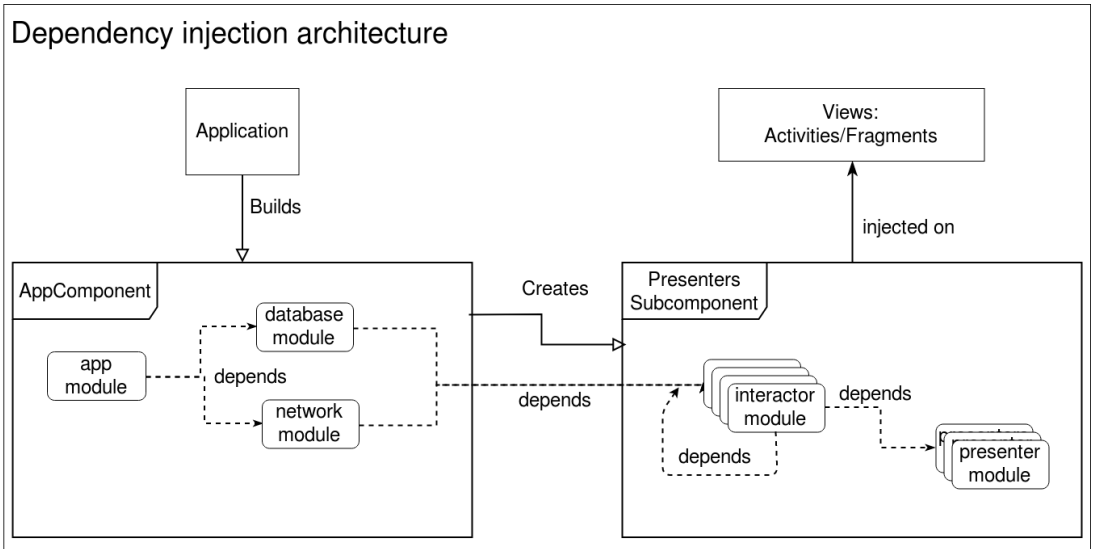


Figura 18 - Diagrama da arquitectura de injecção de dependências

Neste diagrama, figura 18, pode-se observar a organização da injecção de dependências, a aplicação constrói o componente principal, *AppComponent,* que contém os módulos principais a injectar, opcionalmente é possível criar um subcomponente, que expõe módulos que podem estar sujeitos a ter características específicas, tais como o tempo de vida ser o mesmo de uma *view.*

Para utilização foi necessário desenvolver classes *Module* com módulos onde está presente a lógica de como construir certas dependências, e classes *Componente/Subcomponent*, anotadas com os módulos que expõem e onde estes podem ser injectados. Assim todas as dependências, ilustradas na figura com *depends*, são geridas pelo dagger, sendo que as *views* só precisam de anotar a sua dependência de *presenter* com a anotação *@Inject* e invocar o *Appcomponent* requisitando o subcomponente de *presenters.*

Persistência de dados

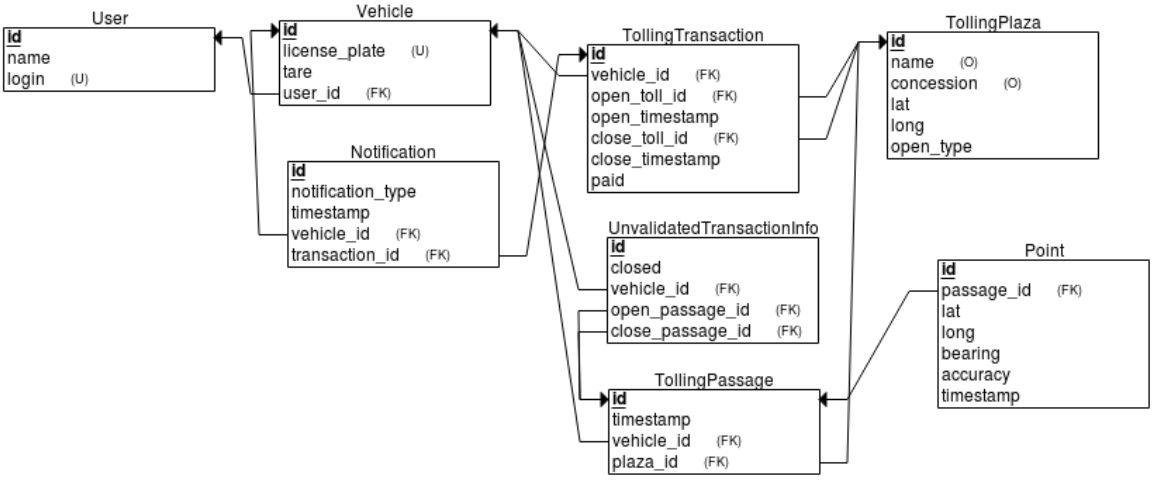


Figura 19 - Modelo de dados gerado pelo *Room*.

Foi identificada a necessidade de persistência de dados na aplicação móvel para poder oferecer funcionamento *offline,* a solução escolhida foi a de utilizar um dos os novos componentes de Arquitectura do Android, neste caso o *Room*[12], este oferece uma abstracção sobre uma base de dados em *SQLite*[13], facilitando a criação e o acesso aos dados, sendo que é apenas necessário implementar um modelo de dados anotado com *@Entity* e com *@PrimaryKey* para identificar a chave primária a utilizar na tabela e *@ForeignKey* para as possíveis chaves estrangeiras permitindo que o *schema seja* gerado automaticamente pelo *Room*, No caso de o modelo de dados da aplicação o *schema* gerado é o presente na figura 19. O acesso aos dados é feitos através de interfaces com funções anotadas com *@Query,* incluindo a *query* ***SQL*** a usar e o modelo retornado que tem de ser compatível com a tabela gerada com o pedido. Este oferece ainda a possibilidade de aceder aos dados tirando partido de um *LiveData* que permite ser observado e evocar acções quando os dados são modificados, facilitando por exemplo actualizações relevantes na interface gráfica.

Comunicação HTTP

```
@GET( value: "tolls/{position}")
fun getNearPlazas(position : LatLong): Deferred<List<TollingPlaza>>

@GET( value: "tolls")
fun getAllPlazas(): Deferred<List<TollingPlaza>>

@GET( value: "vehicles")
fun getVehicleList(): Deferred<List<Vehicle>>

@POST( value: "transaction/verify")
fun verifyTollPassage(@Body passageInfo: TollPassageInfo): Deferred<Boolean>

@POST( value: "transaction/create")
fun createTollingTransaction(@Body transaction: TransactionInfo): Deferred<TollingTransaction>

@PUT( value: "transaction/{id}/cancel")
fun cancelTollingTransaction( @Path( value: "id") id: Int): Deferred<TollingTransaction>
```

Figura 20 - *Snipet* interface de comunicação ***REST*** para *retrofit*

A comunicação com o *backend* é sucedida através do cliente ***HTTP*** *retrofit 2.0*[14], esta foi escolhida derivado à sua robustez e configurabilidade, possibilitam a utilização de *interceptor,* que permitem interceptar um pedido ***HTTP*** e modificá-lo, utilização de cache e de adapters para diferentes opções de *threading.* Como é possível observar na figura 20 para tirar partido do Retrofit é necessário implementar uma interface com as funções de acesso anotadas com o método ***HTTP*** e o seu *path*, a função deve retornar um modelo compatível com o modelo retornado pela API. O Retrofit deve ser instanciado com o host da ***API*** do *backend* e com a interface anteriormente referida, os pedidos são geridos pelo mesmo.

Para usufruir de uma api assíncrona e com sintaxe *Async/Await*[15]*,* é utilizado um adapter para tirar partido das *coroutines*[16] do Kotlin, este permite retornar *Deferred*'s que funcionam como um *future* de suspensão, em que se pode fazer *await,* não bloqueando a thread invocante apenas suspendendo o trabalho a executar até que o pedido ***HTTP*** se suceda.

Visto que o *backend* fornece autenticação com o esquema Basic[17] é utilizado um *interceptor* que intercepta os pedidos ***HTTP*** e injecta os mesmos com o campo *Authorization* e credenciais do utilizador facilitando autenticação na aplicação móvel.

Serviços de Background

Serviços implementados na aplicação móvel responsáveis por executar tarefas que não estão directamente associadas à interacção do utilizador com a interface gráfica, estes podem ser despoletados mesmo quando a aplicação não está a correr, sendo que a infra-estrutura do android se encarrega de instanciar a mesma

serviço de processamento de eventos de geofence

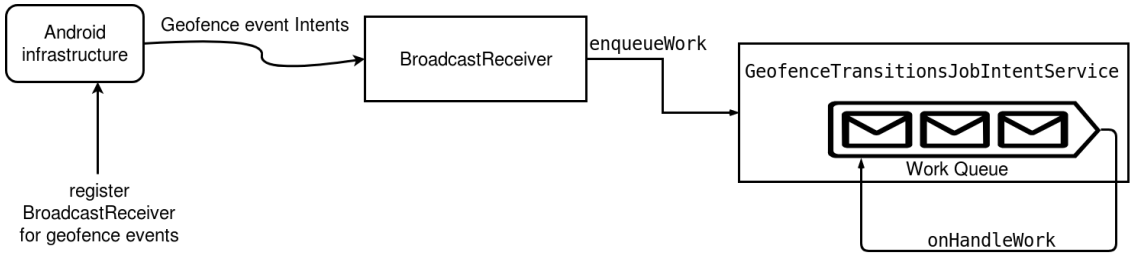


Figura 21 - Esquema do serviço para eventos de geofence.

Os eventos de entrada e saída de um geofence, visto que podem ocorrer em qualquer altura mesmo não estado a aplicação a correr no dispositivo, são processados por um serviço de background, neste caso um *JobIntentService*[18], a infraestrutura do android encarrega-se de instanciar a aplicação e delegar o *Intent*[19] do evento, entrega de mensagem assíncrona do android, em um *BroadcastReceiver*[20] previamente registado para este efeito, este adiciona-o à fila de intents a serem processados, estes são posteriormente processados numa *thread* dedicada para este efeito, onde o se obtém a informação de entrada ou saída do evento e que portagem este estava associado, no caso de ser o geofence que engloba as portagens activas é agendado uma comunicação para saber as próximas portagens a activar.

Serviços de comunicação com Backend

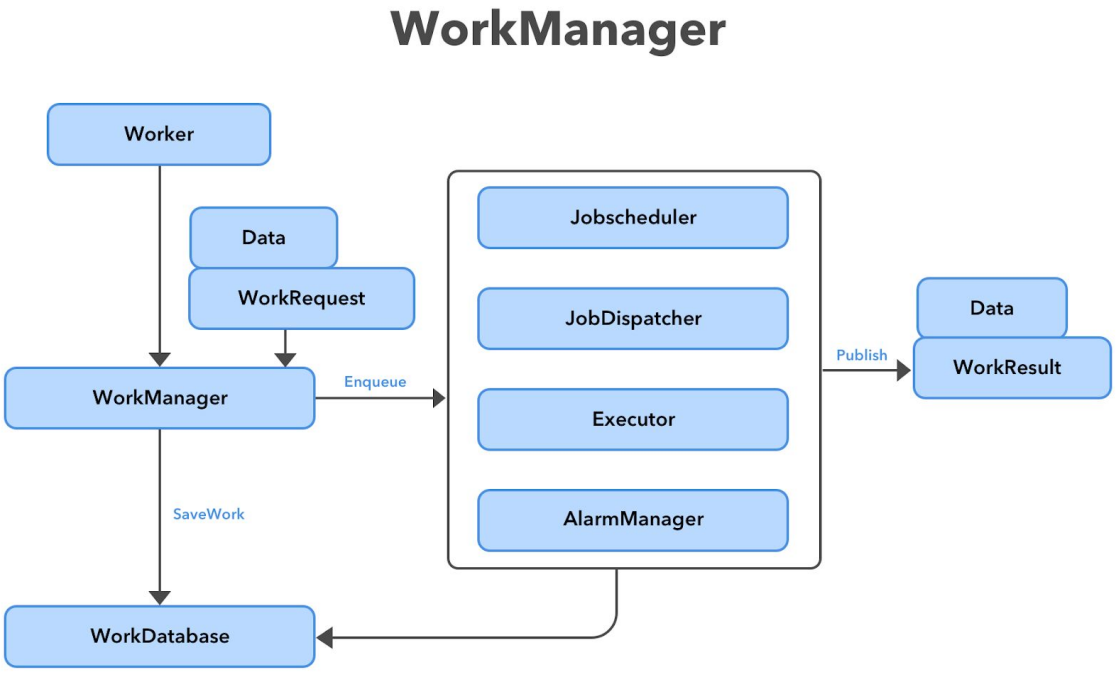


Figura 22 - Arquitectura do WorkManager.

As comunicações com o Backend são feitas tirando partido de um dos novos componentes de arquitectura do Android, neste caso do *WorkManager*[9], este oferece uma abstracção sobre os vários métodos de agendamento de tarefas tais como *Jobscheduler*, *AlarmManager*, entre outros, permitindo compatibilidade com várias versões do Android e com dispositivos que não usufruem do *Google Play Services*[21] do android. O *WorkManager* permite agendar tarefas que ocorrem periodicamente ou apenas uma vez, que quando a tarefa falhe esta seja reagendada, e que esta tenha restrições, tais como existência de conexão, ou o dispositivo não tenha pouca bateria, impedindo assim de utilizar processamento do dispositivo consumindo bateria desnecessariamente. Todos os *Works,* trabalhos agendados, são processados em uma *thread* dedicada, não bloqueando a *thread* principal da aplicação, e são armazenados persistentemente em uma *WorkDatabase,* sobrevivendo a *reboot* do sistema.

3.6 Arquitectura do *Backend*

O Backend é implementado em Kotlin, usando a plataforma SpringBoot[22] e pode ser dividido nos seguintes componentes, presentes na figura abaixo.

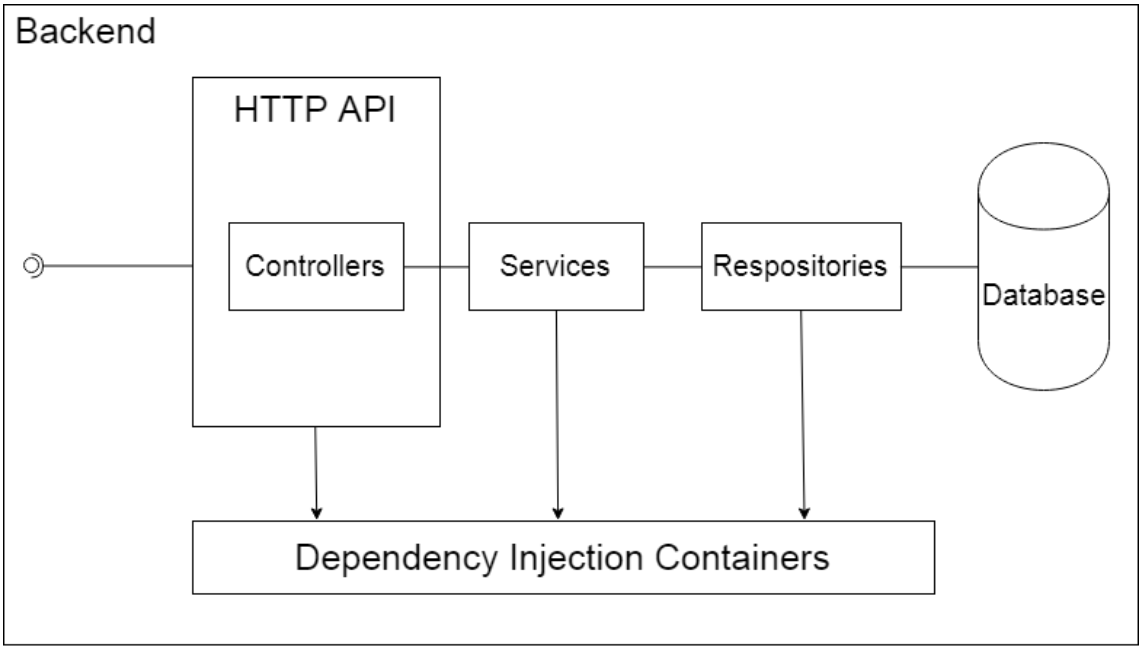


Figura 23 - Arquitectura do Backend

O Backend expõe uma *Web* **API,** definida nos controladores. Estes recebem os pedidos da camada **HTTP** e são encaminhados para os respectivos serviços.
Estes serviços são responsáveis por aplicar a lógica de negócio e gerir a persistência dos dados.
A persistência dos dados é conseguida através de uma camada de abstracção à base de dados.
Esta abstracção está implementada pelos repositórios **JPA** (Java Persistence **API**) fornecidos pelo módulo Spring Data. Estes repositórios são responsáveis por realizar *queries* à base de dados.
O sistema de base de dados usado é o PostgreSQL[23] com extensão espacial PostGIS, oferecendo suporte geográfico. O PostGIS é uma expansão ao PostgreSQL que permite registar tipos que representam objectos geoespaciais e usá-los para realizar *queries* **SQL**.

3.6.1 Inversão de Controlo (IoC) e Injecção de dependências (DI)

O contentor de inversão de controlo na plataforma Spring é representada pela interface ApplicationContext. Este contentor é responsável por instanciar, configurar, construir e gerir os objectos chamados de Beans.
A técnica de injecção de dependências usada foi a injecção por construtor.

3.6.2 Negociação de conteúdo

O tipo de conteúdo suportado pelo Backend é Javascript Object Notation *(***JSON***)*.
Para converter **JSON** em entidades e vice-versa, foi usada a biblioteca Jackson, um *parser* de JSON. É também com esta biblioteca que é feita a preparação da representação das entidades

como, por exemplo, a funcionalidade *JsonIgnore* para retirar informação sensível do conteúdo das respostas.
Outro método de conversão usado é *DateTimeFormat*, fornecido pela plataforma Spring.
As respostas com informação de erro usam o formato *"*application/problem+json*"* descrito na especificação *"Problem Details for* **HTTP APIs***"*.
Foi explorada a opção de usar *hypermedia* como interface para a negociação de conteúdo, mas concluiu-se que era prescindível, tendo em conta o facto de expôr exclusivamente funcionalidade para uma aplicação móvel e outro serviço externo. Apesar das entidades do modelo de dados estarem bem definidas, não se encontrou uma razão forte para existirem relações entre as mesmas que auxiliassem as aplicações clientes.

3.6.3 Autenticação e Autorização

A autenticação suportada está implementada segundo o esquema básico, descrito na especificação do RFC 7617, " The 'Basic' HTTP Authentication Scheme".
Este é suportado pelo módulo Spring Security, que gere o mecanismo de autorização através do *header* **HTTP** *Authorization* e numa autenticação com sucesso, mantendo uma *store* de *cookies* de sessão.
A autorização pode então ser feita através das credenciais do utilizador pelo *header* de autorização, como também pela cookie de sessão.

3.6.4 Camada de dados

Para a implementação da camada de dados foi usado o Hibernate, como abstracção sobre uma base de dados PostgreSQL.
Para suportar os objectos de tipo espacial **GIS** nas *queries* de **SQL**, foi necessário configurar o Hibernate para suportar o dialecto de PostGIS, "org.hibernate.spatial.dialect.postgis.PostgisDialect".

3.6.5 Extensão *PostGIS*

GetNClosestTolls - Procura as N portagens mais próximas

Um dos problemas do sistema, é a incapacidade de registar um número arbitrário de *geofences* de portagem na aplicação móvel, sendo de facto, um limite de 100 na plataforma Android. Devido a esta limitação, foi implementada uma solução tirando partido da extensão *PostGIS*[24], sendo possível registar a geolocalização de uma portagem como um objecto *PostGIS*, Point.
A vantagem desta implementação é que se pode tirar partido de funções fornecidas pelo PostGIS optimizadas para cálculos espaciais. Neste caso em específico, a função *ST_DISTANCE*, calcula a distância cartesiana entre duas geometrias.
A funcionalidade GetNClosestTolls tira partido da função ST_DISTANCE para calcular a distância entre um ponto pretendido e todas as portagens existentes na base de dados, capturando as N mais próximas do ponto pretendido.

CountPointsWithinPolygon - Contagem de pontos dentro de um polígono

Estes polígonos representam a área geográfica da estrada antes e após a portagem, como se pode observar na figura abaixo.

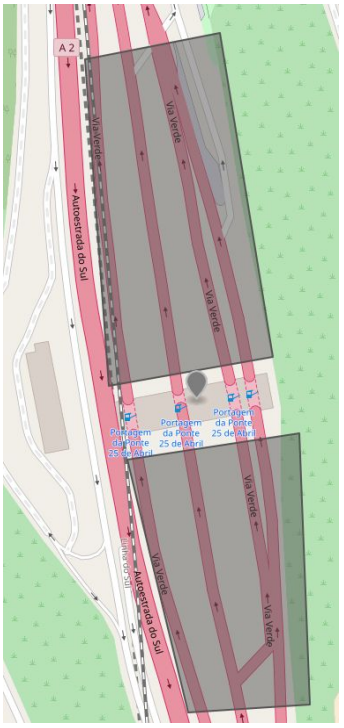


Figura 24 - Polígonos de entrada e de saída da portagem da Ponte 25 de Abril.

Cada portagem define estes dois polígonos, do tipo *Polygon*, definido na plataforma PostGIS.
Sendo *Polygon* um tipo geográfico do PostGIS, tal como *Point*, é possível verificar se uma geolocalização se encontra dentro de um polígono.
A função de PostGIS para este efeito é *ST_INTERSECTS*, que retorna verdadeiro se houver intercepção entre duas geometrias.
Tirando partido desta funcionalidade é possível, dado uma colecção de geolocalizações, verificar quantas delas se encontram dentro do polígono, oferecendo um grau de confiança superior, em vez de apenas depender do *geofence* do dispositivo móvel.
Para a realização desta funcionalidade, quando a aplicação móvel detecta um *geofence* de uma portagem, são momentaneamente coleccionados pontos de geolocalização, que posteriormente serão enviados ao servidor para realizar este método de *enforcement*.

3.6.6 Endpoints

Todos os endpoints necessitam de autenticação e são relativos ao utilizador autenticado.
Para tal, é necessário usar o *Header Authorization* com o esquema de autenticação básica e as credenciais.

TollController

- GET /tolls/{id}
    - Descrição: Obtém a representação de uma portagem, recebendo o id como *path parameter*.
    - Parâmetros de caminho:
        - id (Long): Identificador único de uma portagem.

- GET /tolls
    - Descrição: Obtém a representação de uma colecção de portagens, sendo possível filtrar o resultado pelas actualizações feitas após a data recebida por variável de consulta.
    - Variáveis de consulta:
        - date (Date): Data de início para filtro de portagens

- GET /tolls/nearest
    - Descrição: Obtém a representação da colecção de *number* portagens mais próximas, recebendo como variável de caminho a geolocalização composta por latitude e longitude. Caso o *number* seja omitido, será usado o valor por omissão 90.
    - Variável de consulta:
        - lat (Double): Latitude do ponto de referência.
        - lon (Double): Longitude do ponto de referência.
        - number (Int): Número máximo de portagens solicitadas (Opcional)

- POST /tolls/{toll_id}/verify
    - Descrição: Retorna quantas geolocalizações recebidas no corpo do pedido interceptam os polígonos da portagem recebida por parâmetro de caminho.
    - Parâmetros de caminho:
        - id (Long): Identificador único de uma portagem.
    - Corpo do pedido:
        - colecção de geolocalizações com um azimute associado.

TransactionController

- GET /transactions/{id}
    - Descrição: Obtém a representação de uma transacção, recebendo o id como variável de caminho.
    - Parâmetro de caminho:
        - id (Long): Identificador único da transacção.

- GET /transactions
    - Descrição: Obtém a representação de uma colecção de transacções, sendo possível filtrar o resultado pelas actualizações feitas após a data recebida por variável de consulta.
    - Variáveis de consulta:
        - date (Date): Data de ínicio para filtro de transacções.

- POST /transactions/create
    - Descrição: Cria uma transacção com estado *"INCOMPLETE"* com os dois eventos de passagem de portagem, realiza o algoritmo de enforcement e retorna a nova transacção criada.
    - Corpo do pedido:
        - inputTransaction: objecto com a informação necessária para registar uma transacção.
- PUT /transactions/{transaction_id}/confirm
    - Descrição: Neste *endpoint* é feita a confirmação da transacção, passando o estado da mesma a *"AWAITING_CONFIRMATION"*. Caso, haja correcções relativamente às portagens transpassadas, estas alterações serão registadas através da informação presente no corpo do pedido. É retornada a respectiva transacção actualizada.
    - Parâmetros de caminho:
        - transaction_id (Long): Identificador único da transacção.
    - Corpo do pedido:
        - inputTransaction: objecto com a informação necessária para registar uma transacção.

- PUT /transactions/{transaction_id}/cancel
    - Descrição: Representação de uma colecção de transacções, sendo possível filtrar o resultado pelas actualizações feitas após a data recebida por variável de consulta.
    - Parâmetros de caminho:
        - transaction_id (Long): Identificador único da transacção.
    - Corpo do pedido:
        - inputTransaction: objecto com a informação necessária para registar uma transacção.

- GET /transactions/status
    - Descrição: Obtém uma representação da última transacção associada ao veículo respectivo.
    - Variáveis de consulta:
        - vehicle_id (Long)
    - Corpo do pedido:
        - inputTransaction: objecto com a informação necessária para registar uma transacção.

UserController

- GET /users/authentication
    - Descrição: Obtém a representação de um utilizador.

VehicleController

- GET /vehicles
    - Descrição: Obtém uma representação de uma colecção de veículos, sendo possível filtrar o resultado pelas actualizações feitas após a data recebida por variável de consulta.
    - Variáveis de consulta:
        - date (Date): Data de início para filtro de veículos.

- GET /vehicles/{id} -
    - Descrição: Representação de um veículo, recebendo o id como parâmetro de caminho.
    - Parâmetros de caminho:
        - id (Long): Identificador único do veículo.

- POST /vehicles
    - Descrição: Registo de um veículo.
    - Corpo do pedido:
        - vehicle (InputModelVehicle): objecto que contém uma matrícula e a respectiva tara.

Capítulo 4 - Resultados e conclusões

4.1 Resultados

A aplicação foi testada em contexto real, numa viagem pela Ponte 25 de Abril.
Foi acionada a detecção de portagens através da aplicação móvel a alguma distância, sendo que mediante passagem na portagem da ponte a aplicação alertou o dispositivo com a notificação de passagem na mesma, sendo assim bem sucedida na detecção da portagem. É de notar que o dispositivo móvel utilizado não usufrui de **GPS** nativo pelo que a probabilidade da existência de erro era avultada, mesmo assim a funcionalidade foi bem sucedida.

4.2 Conclusões

Neste projecto investigou-se a viabilidade de usar a tecnologia **GPS** com a finalidade de detectar passagens em portagem, conclui-se que a solução apresentada é viável, apesar da existência de algumas desvantagens, para cada portagem é necessário informação de geolocalização específica que caso não exista em formato digital tem de ser inserida manualmente, tal como a possibilidade de falha por parte do serviço de geofence, o que implica que o serviço de enforcement presente no clearing continua a ser necessário para os casos extremos.

4.3 Trabalho Futuro

Tendo em conta a complexidade do problema que este projecto visa resolver, alguns aspectos ficam por melhorar. Apesar de as principais dificuldades terem sido superadas, os casos extremos necessitam ainda de alguma atenção e existem bastantes variáveis que podem dificultar o funcionamento adequado da aplicação, tais como diferentes dispositivos terem funcionalidades de geolocalização diferentes, diferentes localizações terem níveis de confiança baixos para funcionamento de geofencing ou até casos de falha por parte do backend. Ficam então por testar e mitigar possíveis falhas nestes casos extremos.

4.3.1 Delegação de permissão de veículos

A delegação de permissão de veículos de terceiros pode vir a provar ser bastante enriquecedora, tendo em conta o número de aplicações que potencialmente pode ter, como por exemplo:

- Firmas de aluguer de veículos;
- Empréstimo de viaturas entre indivíduos.

4.3.2 Fortalecer a fiabilidade do método de *enforcement*

Esta técnica passa por fazer a intersecção entre uma colecção de pontos geográficos e os polígonos de uma portagem. A proposta de melhoria seria que para cada ponto, fosse gerado um círculo com o ponto respectivo no seu centro, e então efectuar a intersecção de geometrias entre cada círculo e os polígonos da portagem, oferecendo uma margem de erro, que pode melhorar significativamente a fiabilidade do método de *enforcement*.
Para a criação de um círculo, seria usada a função ST_Buffer, que recebe como parâmetro uma geometria e o raio do círculo pretendido, retornando uma geometria. Se por exemplo, a

geometria dada como parâmetro for um ponto, a geometria resultante será um polígono próximo de um círculo.

Referências

[1] - Via Verde– um exemplo Português de criatividade e inovação ao serviço do Cliente por João Pecegueiro, visitado a 7 de Setembro de 2018

[2] - Open Road Tolling, visitado a 7 de Setembro de 2018

[3] - Technology options for the European Electronic Toll Service, visitado a 20 de Abril de 2018

[4] - LinktGO, visitado a 20 de Abril de 2018

[5] - Android Geofences, visitado a 7 de Setembro de 2018

[6] - Kotlin, visitado a 7 de Setembro de 2018

[7] - Android Jetpack, visitado a 7 de Setembro de 2018

[8] - MVP architecture, visitado a 7 de Setembro de 2018

[9] - Android Maps Android API, visitado a 7 de Setembro de 2018

[10] - WorkManger, visitado a 7 de Setembro de 2018

[11] - Dagger DI, visitado a 7 de Setembro de 2018

[12] - Room, visitado a 12 de Março de 20118

[13] - SQLite, visitado a 7 de Setembro de 20188

[14] - Retrofit 2.0, visitado a 7 de Setembro de 2018

[15] - Async/Await, visitado a 7 de Setembro de 2018

[16] - Kotlin coroutines, visitado a 7 de Setembro de 2018

[17] - Basic scheme, visitado a 7 de Setembro de 2018

[18] - JobIntentService, visitado a 7 de Setembro de 2018

[19] - Intent, visitado a 7 de Setembro de 2018

[20] - BroadcastReceiver, visitado a 7 de Setembro de 2018

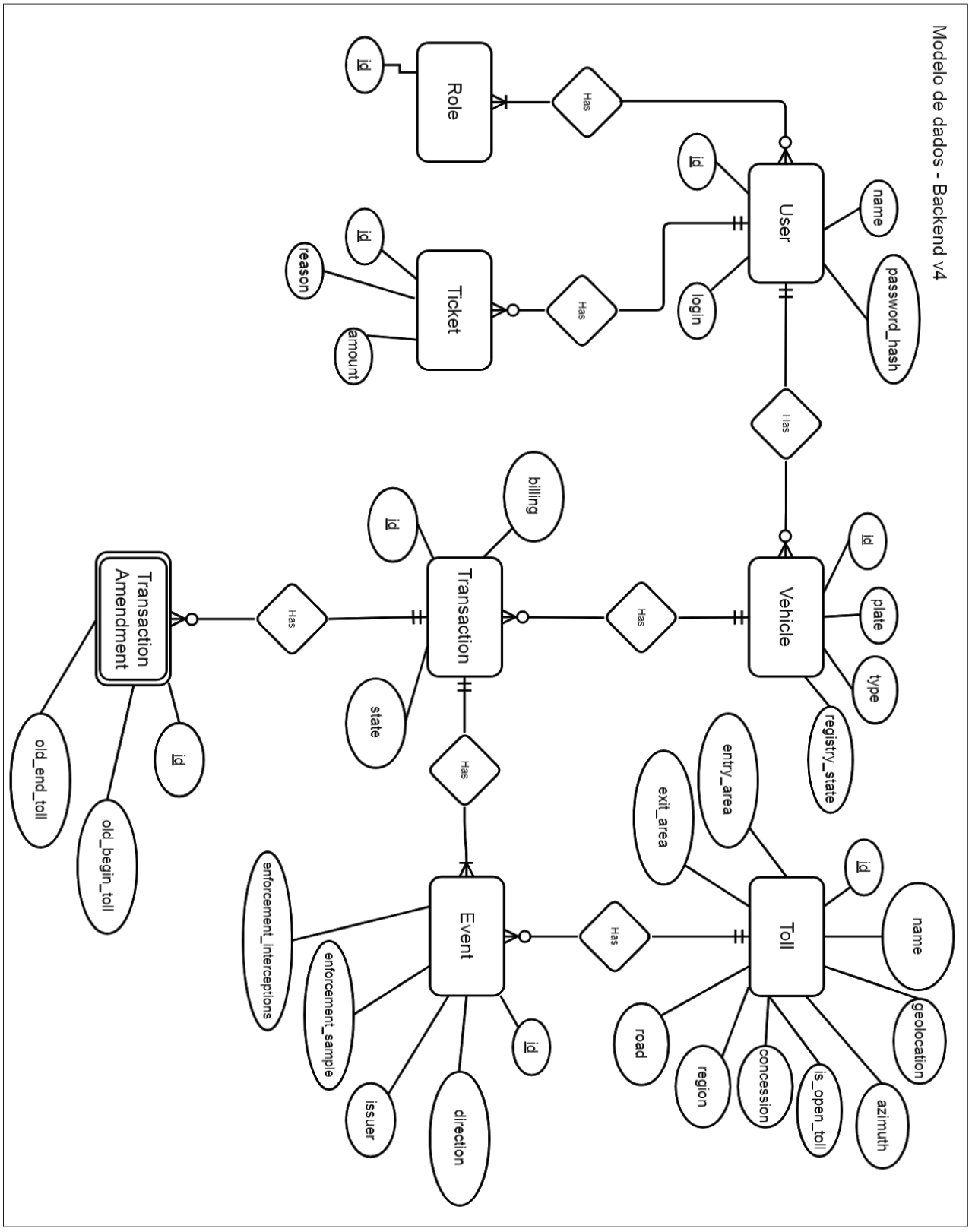
[21] - Google Play Services, visitado a 7 de Setembro de 2018

[22] - Spring Framework, visitado a 7 de Setembro de 2018

[23] - PostgreSQL, visitado a 7 de Setembro de 2018

[24] - PostGIS, visitado a 7 de Setembro de 2018

A.1 Modelos de dados



A.2 Grafo de vistas