isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

On-demand Mobility Platform

DEFINITIVO

Autor

Carlos Pedro Marques Afonso

Orientador

Prof. Doutora Ana Cristina da Costa Oliveira Alves

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, fevereiro, 2021

isec
Engenharia

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

On-demand Mobility Platform

Relatório de Estágio de Natureza Profissional para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Desenvolvimento de Software

Autor

Carlos Pedro Marques Afonso

Orientador

Prof. Doutora Ana Cristina da Costa Oliveira Alves

Supervisor na empresa Ubiwhere

Engenheiro André Filipe Gomes Duarte

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, fevereiro, 2021

Agradecimentos

Agradeço à Ubiwhere, sobretudo ao meu orientador, André Duarte, pela oportunidade que me ofereceu em realizar este estágio e pela ajuda e constante preocupação que sempre demonstrou. Agradeço também a todos os elementos tanto da empresa de Aveiro como de Coimbra por toda a boa disposição demonstrada e pela constante preocupação em que tudo estivesse do meu agrado e que todos se sentissem confortáveis na empresa.

Agradeço à minha orientadora da parte do ISEC, a Professora Doutora Ana Alves, por todo o profissionalismo, tanto como professora, - onde nas aulas primou pela excelência pedagógica e didática -, como orientadora e acompanhamento exemplares, do início ao fim do estágio.

Agradeço à minha família, nomeadamente aos meus pais por terem acreditado sempre em mim e por todo o esforço e apoio que me forneceram para poder chegar a esta fase e concluí-la com sucesso.

Agradeço ao Instituto Superior de Engenharia de Coimbra (ISEC), mais concretamente, aos docentes do Mestrado de Informática e Sistemas do ramo de Desenvolvimento de Software pelos conhecimentos e conselhos que me transmitiram e por me terem ajudado, a aprofundar o meu conhecimento e a desenvolver as minhas capacidades cognitivas e analíticas.

Agradeço aos meus amigos e colegas de curso, por todo o companheirismo e amizade durante todas as fases.

Por fim, mas não menos importante, agradeço aos restantes estagiários, Stefan Loureiro, João Ferreiro, José Gomes e Ana Farinha Alves pelos momentos de convívio, amizade, descontração e por todas as sugestões que partilharam comigo.

Resumo

Uma grande parte da população vive nas áreas urbanas, sendo que o contínuo aumento da população leva a que as cidades tenham de utilizar os seus serviços de forma eficiente, otimizando desta forma os seus recursos. Sendo assim, exige-se uma gestão ainda mais eficiente de modo a aumentar os índices de satisfação dos seus cidadãos. O setor da Mobilidade Inteligente apresenta uma importância cada vez maior, dando resposta a uma série de necessidades. É desta forma que surge o presente trabalho, sendo proposta o desenvolvimento de uma plataforma *on-demand* que seja capaz de agregar pedidos de transporte, de forma a agrupá-los e permitir a otimização da rota, tornando assim o sistema mais eficiente.

Esta proposta irá assim apoiar as regiões que necessitem de infraestruturas e opções de mobilidade com a criação de soluções de mobilidade urbana sustentável para a população mais envelhecida, pessoas com mobilidade reduzida ou pessoas que não tenham acesso a veículo próprio.

Palavras Chave: *On-demand, mobility-platform, clustering, smart-cities,* mobilidade urbana sustentável, otimização de pontos de recolha de pessoas numa cidade

Abstract

A large part of the population lives in urban areas, and the continued increase in the population leads cities to have to use their services efficiently, thereby optimising their resources. Therefore, even more efficient management is required in order to increase the satisfaction rates of citizens. The Smart Mobility sector has become increasingly important, responding to a number of needs. This is how the present work emerges, and the development of an on-demand platform that is capable of aggregating transport requests is proposed, in order to group them and allow the optimization of the route, thus making the system more efficient.

This proposal will thus support regions that need sustainable urban mobility infrastructure and options for the senior population, people with reduced mobility or persons who do not have access to their own vehicle.

Keywords: On-demand, mobility-platform, clustering, smart-cities, sustainable urban mobility, optimisation of pick-up points in a city

Índice

Lista de Figuras

Lista de Tabelas

Definições e Acrónimos

AMPQ – *Advanced Message Queueing Protocol*

API – *Application Programming Interface*

CD – *Continuous Delivery*

CI – *Continuous Integration*

CMMI – *Capability Maturity Model Integration*

CRUD – *Create, Read, Update, Delete*

CTT – Correios de Portugal

DARP – *Dial-a-Ride Problem*

DBSCAN – *Density-Based Spatial Clustering of Application with Noise*

DEIS – Departamento de Engenharia Informática e Sistemas

DHL – *Dalsey, Hillblom* e *Lynn*

DMDBSCAN – *Dynamic Method of Density-Based Spatial Clustering of Application with Noise*

DRT – *Demand Responsive Transport*

Eps – épsilon – Dois pontos são considerados vizinhos se a distância entre os dois pontos estiver abaixo do valor de épsilon.

GNU – *General Public License*

GPS – *Global Positioning System*

HoV – *High-occupancy Vehicle lane*

HTTP – *HyperText Transfer Protocol*

iOS – *iPhone Operating System*

IoT – *Internet of Thing*s

ISEC – Instituto Superior de Engenharia de Coimbra

ISO – *International Organization for Standarization*

JSON – *Java Script Object Notation*

JWT – *Json Web Tokens*

MaaS – *Mobility as a Service*

MinPts – Número mínimo de pontos necessários para formar um *cluster*

MoD – *Mobility on Demand*

MUMA – *Moovit Urban Mobility Analytics*

NIF – Número de Identificação Fiscal

ORM – *Object-Relational Mapping*

OSRM – *Open Source Routing Machine*

PIN – *Personal Identification Number*

RAM – *Random Access Memory*

REST – *Representational State Transfer*

SSD – *Solid State Drive*

SSPL – *Server Side Public Licensing*

TNT – *Thomas Nationwide Transport*

ToD – *Transport on Demand*

TSP – *Traveling Salesman Problem*

UI – *User Interface*

URL – *Uniform Resource Locator*

VRP – *Vehicle Routing Problem*

WEB – *World Wide Web*

1 Introdução

Neste capítulo é apresentado um enquadramento sobre o tema da mobilidade, uma descrição sobre a entidade de acolhimento do estágio, motivação, objetivos e a estrutura que o presente documento apresenta.

À medida que as cidades vão ficando cheias de pessoas, novas soluções vão surgindo com o objetivo de resolver o desafio da mobilidade. Sensivelmente metade da população mundial vive nas áreas urbanas e até 2030 está previsto que este número seja aumentado em 10% [1]. O crescimento populacional nas cidades coloca pressão na mobilidade, trazendo diversos problemas, tais como, supressão das vias, obras, aumento do número de acidentes, tendo como consequência feridos graves ou até mortes, aumento do congestionamento de trânsito, levando a que as populações percam muito tempo em filas, apresentem dificuldades em estacionarem e a que haja consequentemente um aumento do nível de poluição, sendo este último, um fator bastante preocupante para a saúde pública. Além dos problemas mencionados anteriormente, este crescimento populacional nas cidades faz com que se abra um fosso entre a periferia e o centro da cidade. Este é o principal problema, o qual se pretende resolver com o presente trabalho, uma vez que existem muitas pessoas que vivem em meios isolados com poucos recursos e não têm possibilidade de se deslocarem para o meio citadino.

Este documento tem como objetivo apresentar todo o trabalho que foi desenvolvido ao longo do ano letivo 2019/2020 no âmbito da unidade curricular de Estágio do Mestrado de Informática e Sistemas, mais concretamente do ramo de Desenvolvimento de Software do Instituto Superior de Engenharia de Coimbra. O estágio foca-se na área das *Smart Cities* e decorreu nas instalações da Ubiwhere, mais especificamente no Instituto Pedro Nunes, em Coimbra.

1.1 Entidade de acolhimento - Ubiwhere

A **Ubiwhere** foi fundada em 2007, na cidade de Aveiro, focando-se na investigação e desenvolvimento de soluções inteligentes através das tecnologias mais recentes e inovadoras de forma a serem criados produtos que possam marcar a diferença e acrescentar assim valor aos seus clientes. Deste modo, este avanço tecnológico traz várias vantagens, tais como, a melhoria da qualidade de vida das pessoas e um aumento do crescimento da empresa, levando consequentemente a um aumento do número de trabalhadores. Esta empresa possui dois escritórios localizados em Aveiro e em Coimbra, no Instituto Pedro Nunes e apresenta como principal foco as áreas das **Cidades Inteligentes, Internet das Coisas** (IoT) e **Telecomunicações.**

Atualmente, a empresa emprega mais de 70 colaboradores, sendo que a mesma está comprometida em garantir as melhores práticas ao nível da gestão de qualidade, com certificações como a ISO 9001 e o nível 3 de CMMI (*Capability Maturity Model Integration*) nos processos de desenvolvimento interno [2].

Por parte da empresa o estagiário teve como orientador o Mestre André Filipe Gomes Duarte e como orientadora por parte do ISEC, a Professora Doutora Ana Cristina da Costa Oliveira Alves.

1.2 Motivação

Tal como já foi referido anteriormente, a mobilidade apresenta vários problemas, no âmbito da qual surge o presente estágio, que servem como fonte de motivação, tais como:

- Elevado número de veículos privados a circular diariamente que causa grandes dificuldades na mobilidade;
- Elevada percentagem de veículos com apenas um passageiro levando a uma reduzida eficiência, sendo que isto provoca:
    - Impacto negativo na fluidez das vias de trânsito;
    - Dificuldades no estacionamento;
    - Aumento da poluição;
    - Elevados custos;
- Existência de diversas pessoas em meios rurais que possuem poucos recursos;
- Necessidade de apoiar as regiões rurais onde há poucas infraestruturas e opções de transporte para a população mais envelhecida, pessoas com mobilidade reduzida ou pessoas que não tenham acesso a veículo próprio ou transporte público.

Desta forma, surgiu a oportunidade de se desenvolver uma proposta de solução no âmbito da mobilidade inteligente que visa a criação de um serviço de mobilidade *on-demand* capaz de agregar pedidos de transporte, agrupá-los, registar ofertas de boleia e garantir a otimização da rota com recurso a um serviço que se encontra alojado internamente na empresa, tornando assim o sistema eficiente.

1.3 Objetivos

A proposta de estágio (constante do anexo E) elaborada pela Ubiwhere teve como principal objetivo o desenvolvimento de um serviço de mobilidade *on-demand* que fosse capaz de agregar

pedidos de transporte, agrupar as pessoas e garantir a otimização da rota e respetivos pontos de recolha.

O processo de desenvolvimento durante o estágio consistiu nas seguintes fases:

- Produção de um documento de estágio que inclui estado da arte e decisões tecnológicas para o desenvolvimento da solução;
- Desenvolvimento de uma aplicação *web* para registar ofertas de transporte por parte de passageiros e ofertas de boleia por parte de condutores;
- Desenvolvimento de um simulador de dados nomeadamente para o registo de pedidos de transporte e registo de ofertas de boleia com recurso à API que foi desenvolvida;
- Desenvolvimento de um módulo de *clustering* de agrupamento de pedidos de transporte;
- Desenvolvimento de um mecanismo automático de atribuição de ofertas de boleias a grupos de pedidos de transporte através do módulo de *clustering*.

Os dados do *clustering* são referentes a pedidos que são numa primeira fase simulados, mas posteriormente estarão a ser alimentados pela plataforma. Esta simulação ocorreu através de uma API REST que se encontra integrada na mesma.

1.4 Estrutura do relatório

Este relatório pretende detalhar o trabalho desenvolvido durante o estágio. No capítulo 2, é apresentado o estado da arte, onde são abordados vários conceitos relacionados com o presente trabalho. Neste capítulo são ainda abordadas várias técnicas de incentivo ao agrupamento de pessoas e várias técnicas de *clustering* com a análise e comparação das mesmas. Ainda são descritas várias soluções comerciais que já se encontram implementadas no mercado e que servem de concorrentes à aplicação desenvolvida. O capítulo termina com as conclusões obtidas na análise do estado da arte.

No capítulo 3, é apresentada a metodologia e planeamento que foram aplicados ao longo dos dois semestres de estágio.

No capítulo 4, é dado a conhecer as ferramentas que permitiram gerir o trabalho assim como as tecnologias utilizadas ao longo do estágio para implementar a solução. Este capítulo inclui uma comparação entre vários motores de base de dados tendo em conta diferentes características relevantes ao domínio geográfico.

No capítulo 5, são descritos os requisitos tanto da aplicação de *backoffice*, como do módulo de *clustering* e otimização da rota. No que toca à aplicação de *backoffice*, são abordados os requisitos

funcionais, não funcionais e os casos de uso. Para o módulo de *clustering* e otimização da rota são apresentados os requisitos gerais e as regras de associação de boleias a condutores.

No capítulo 6, é feita uma análise à arquitetura desenvolvida, seguida de uma descrição sobre cada módulo que compõe a solução. Este capítulo termina com a descrição da base de dados e da *pipeline* de processamento referente ao módulo de *clustering* e otimização da rota.

No capítulo 7, é descrita a implementação da aplicação *web* de *backoffice*, do serviço de simulação de dados e do módulo de *clustering* e otimização da rota. Segue com a descrição da API que foi desenvolvida e conclui com a análise sobre a determinação do valor ótimo de cada parâmetro para a técnica de *clustering* escolhida.

O capítulo 8, expõe a conclusão final, onde é realizada uma reflexão sobre o cumprimento dos objetivos, bem como sobre o trabalho desenvolvido e a desenvolver.

Em complemento e como suporte de informação a estes capítulos, são listadas as referências bibliográficas e um conjunto de anexos onde estão incluídos documentos complementares para consulta, designadamente:

Anexo A: Especificação e levantamento de requisitos;

Anexo B: Diagrama de *Workflow* do módulo de *clustering* e otimização de rota;

Anexo C: Diagrama da pipeline de processamento;

Anexo D: *Screenshots* do *backoffice*;

Anexo E: Proposta de estágio.

2 Estado da arte

Neste capítulo serão descritos alguns conceitos relevantes para o desenvolvimento do projeto e algumas soluções já existentes no mercado sobre o tema *Transport On Demand.*

2.1 Mobilidade Urbana

Mobilidade é a forma como nos deslocamos, eliminando fronteiras e criando maior acessibilidade às pessoas, melhorando desta forma a qualidade de vida das mesmas. Torna-se assim num fator de progresso e desenvolvimento económico. Para que atinja esses objetivos deverá ser implementada de uma forma sustentável através de medidas decisivas, tais como, a adoção de taxas de carbono para os combustíveis, a utilização de biocombustíveis, a sensibilização da população para a utilização de transportes alternativos, a atribuição de incentivos para a utilização de transportes públicos, entre outras [3].

A diversidade da oferta de mobilidade nacional, europeia e internacional constitui uma fonte de incentivos para que a sociedade possa ter um comportamento amigo do ambiente pretendendo a melhoria contínua das condições de deslocação, a diminuição dos impactos no ambiente e o aumento da qualidade de vida das pessoas, razões pelas quais este conceito apresenta cada vez mais uma maior relevância [4].

A **Mobilidade Urbana** é vista como a facilidade de deslocamento das pessoas e bens, com o objetivo de desenvolver atividades económicas e sociais no perímetro urbano de cidades, aglomerações urbanas e regiões metropolitanas [5]. O principal objetivo da Mobilidade Urbana é melhorar a qualidade de vida das pessoas de forma sustentável, sendo que isso inclui questões económicas, sociais e políticas. A forma de atingir este objetivo passa por um plano de mobilidade urbana, ou seja, um conjunto de procedimentos para melhorar o deslocamento sustentável das pessoas, facilitando o acesso a trabalho e serviços. Os procedimentos nesse plano incluem melhorar o planeamento dos transportes públicos, melhorar aspetos de ordenamento do território, reduzir o impacto dos transportes na saúde pública, melhorar aspetos de acessibilidade e adotar planos para reduzir a sinistralidade rodoviária, melhorando assim a segurança e capacidade de deslocação das pessoas [6].

2.1.1 Mobility On Demand

Mobility On Demand (MoD) – em Português, Mobilidade a Pedido, é uma abordagem inovadora e focada no utilizador, que utiliza serviços de mobilidade emergentes, redes e operações de trânsito

integradas, dados em tempo real, viajantes conectados e sistemas inteligentes de transportes cooperativos para permitir uma abordagem de sistemas de transporte mais focada no viajante, oferecendo melhores opções de mobilidade a todos os viajantes e utilizadores do sistema de maneira eficiente e segura [7].

Hoje em dia, existem várias empresas que estão muito presentes na parte da mobilidade a pedido e que atualmente estão a ganhar cada vez mais relevância no mercado, designadamente, a **Uber** [8], **Cabify** [9] **e o Lyft** [10]. Estes serviços incluem um pedido de transporte porta a porta (normalmente pedido através de uma aplicação móvel) com visualização do condutor mais próximo do ponto de partida, a informação do condutor e veículo e também informação do custo da viagem. As vantagens da utilização deste serviço incluem maior transparência no custo para quem pede o serviço, uma maior eficiência por redução do número de veículos na cidade e a rentabilização de um veículo, além da criação de algumas oportunidades de trabalho.

Uma variante destes serviços é designada de ***car-sharing***. Enquanto nos casos do Uber e outros semelhantes, o pedido de viagem inclui um veículo e o seu condutor, no ***car-sharing***, há a requisição apenas do veículo para efetuar a viagem. A vantagem deste tipo de serviço está focada essencialmente na rentabilização de um veículo que é reaproveitado para múltiplas viagens, reduzindo também o número de veículos que consequentemente melhora a mobilidade na cidade. Este conceito estende-se a outros tipos de veículos, nomeadamente, *scooters* e trotinetes. Algumas soluções presentes no mercado utilizando este conceito incluem: **DriveNow** [11]**, GetAround** [12] **e o Car2Go** [13]**,** no que diz respeito a veículos de 4 rodas, **Lime** [14] **e Circ** [15] para trotinetes elétricas, e **eCoolTra** [16] para scooters. De salientar que tanto a **Lime, Circ** e **eCoolTra** são para problemas que utilizam o conceito de ***last-mile*** [17].

Cada vez mais, o processo de pedido de viagem está muito ligado às tecnologias de informação. A facilidade de acesso a dispositivos móveis permite a existência de técnicas de pedidos de veículos ou viagens, a partir da aplicação móvel. O utilizador indica o seu local de origem e a aplicação fornece as opções mais relevantes, nomeadamente, os veículos ou condutores mais próximos, informação do custo da viagem e tempo estimado de acesso ao transporte, bem como tempo estimado de chegada. Parte do utilizador aceitar as opções para iniciar a sua viagem. A transação financeira associada é também automática com recurso a saldos ou recurso de crédito.

Este processo integrado é um avanço significativo quer em termos de simplicidade, quer de conveniência relativo a processos anteriores, como por exemplo o de pedido de um táxi que normalmente envolvia uma ou mais chamadas telefónicas, ausência de informação em tempo real e transações financeiras físicas (troca de dinheiro).

No que diz respeito ao acesso ao veículo sem condutor, a aplicação móvel tem um papel fundamental nomeadamente no destranque do veículo, funcionando como chave para a abertura.

Os fatores diferenciadores que o presente trabalho apresenta relativamente a outras entidades, tais como a **Uber**, **Cabify** e **Lyft**, são o facto de não funcionar em tempo real, ou seja, são viagens pré-programadas, sendo que além disso o objetivo passa por maximizar o número de passageiros num determinado veículo. Para além disso a principal diferença é fazer com que pessoas que vivam em meios isolados possam ser transportadas para o meio citadino, enquanto que as entidades mencionadas anteriormente funcionam apenas dentro da cidade. O que o presente trabalho apresenta de comum com as entidades mencionadas anteriormente é o facto de serem *on-demand*.

2.1.1.1 Transport On demand

***Transport On Demand* (ToD)** refere-se ao transporte de passageiros ou mercadorias entre determinados pontos de origem e destinos que são requisitados por utilizadores.

Este conceito encontra-se muito presente em serviços telefónicos de transporte para pessoas idosas e portadores de deficiência (bombeiros, serviços especializados no transporte de doentes), serviços de correio urbano (CTT) e de emergência. Em todos estes sistemas, os utilizadores formulam pedidos de transporte desde um local de partida para um determinado local de destino. Posteriormente, essas solicitações são atendidas por um conjunto de veículos capacitados que geralmente fornecem um serviço compartilhado, ou seja, em que vários passageiros ou mercadorias podem estar num veículo ao mesmo tempo.

Os sistemas ToD podem ser estáticos ou dinâmicos. No primeiro caso, os pedidos são conhecidos de antemão, enquanto que no segundo caso, as solicitações são recebidas dinamicamente, ou seja, à medida que as necessidades dos utilizadores vão aparecendo, sendo que as rotas dos veículos devem ser ajustadas em tempo real para atender os pedidos que vão surgindo ao longo do tempo.

Atualmente, os utilizadores aderem mais ao conceito de ToD dinâmico. Semelhante quando os mesmos efetuam uma compra online, a maioria prefere que a encomenda seja entregue (entrega *just-in-time*) em casa (através do serviço porta a porta) levando a que haja uma redução de custos e uma melhor otimização do tempo para os mesmos, evitando assim que os utilizadores tenham que se deslocar para levantar a encomenda.

Desta forma, nos próximos subtópicos irão ser especificados vários conceitos que se encontram englobados no ***Transport On Demand***, sendo eles: ***Demand Responsive Transport, Dial-a-Ride problem*** e ***The Vehicle Routing Problem*** [18].

2.1.1.1.1 Demand Responsive Transport

***Demand Responsive Transport* (DRT)** é caracterizado por ter uma frota de veículos de pequena/média dimensão que se deslocam com itinerários e horários flexíveis operando num modo de partilha de viagem entre um local de origem e um local de destino, tendo sempre como base as

necessidades dos passageiros [19]. Os sistemas de DRT adequam-se especialmente a um serviço de transporte público em áreas rurais ou de pouca procura, como por exemplo, serviços noturnos.

Este conceito tem como objetivo reduzir as emissões de carbono e melhorar os fluxos de transporte nas cidades. Com o crescimento populacional, existe um maior aumento na utilização de carros particulares, sendo que as autoridades governamentais procuram apoiar um maior uso de soluções de mobilidade, de forma a, tal como foi referido anteriormente, reduzir as emissões de carbono, incluindo transporte público, ciclismo e caminhada [20]. O mesmo foi desenvolvido, sobretudo para ajudar pessoas de mobilidade reduzida, e é visto como um meio eficiente e ecológico, sendo que os meios eletrónicos facilitaram bastante o processo de pagamento de um pedido de transporte efetuado por um cliente, ajudando, desta forma, à questão da eficiência a simplicidade.

2.1.1.1.2 Dial-a-Ride Problem

O ***Dial-a-Ride Problem*** **(DARP)** consiste em calcular rotas e horários de veículos para vários utilizadores que efetuam pedidos de transporte de um local de origem para um local de destino. O objetivo passa por calcular o custo mínimo de rotas para satisfazer o maior número possível de utilizadores, com base num determinado conjunto de restrições [21]. O exemplo mais comum deste conceito surge como um serviço de transporte avançado de reserva para pessoas com mobilidade reduzida, ou seja, idosos ou pessoas portadoras de deficiência.

Desta forma, pode-se verificar que se devem considerar duas questões na resolução deste problema, sendo elas, a minimização de custos relativos à satisfação total dos pedidos e a maximização do número de pedidos tendo em conta o número de veículos existentes.

Existem algumas empresas que atualmente prestam vários serviços de *DARP*, tais como, a Flor da Ria [22] e os bombeiros para transporte de doentes. Tratando apenas especificamente do caso de transportes de deficientes, um ponto de recolha corresponde ao local onde uma determinada pessoa portadora de deficiência deve ser "apanhada" e o ponto de entrega da mesma diz respeito ao local onde o mesmo deve ser entregue. Estas empresas ao utilizar este tipo de serviço pretendem atingir vários objetivos, nomeadamente, reduzir os seus custos para satisfazer os pedidos de vários utilizadores, maximizar a satisfação de pedidos tendo em conta a disponibilidade de veículos existentes [23].

2.1.1.1.3 The Vehicle Routing Problem

Vehicle Routing Problem **(VRP)** é um problema de otimização que pretende fornecer o caminho com menor custo para entregar bens e serviços a clientes em diferentes locais, considerando um conjunto de veículos e restrições. Este também é conhecido como o problema do vendedor ambulante (TSP) [24] , sendo que o mesmo surgiu para representar o cenário onde um caixeiro

viajante pretende visitar **n** cidades sem passar mais do que uma vez por cada uma e retornar à cidade de partida.

Existem várias empresas que utilizam este conceito, designadamente as que efetuam serviços de transportes de encomendas, tais como a TNT e DHL [25]. O objetivo destas empresas passa por garantir que um determinado veículo de transporte entrega a maior quantidade de produtos para a maioria dos clientes, no menor trajeto possível e com a maior rapidez e eficiência possível.

Este conceito cada vez é mais importante, já que o planeamento de rotas atualmente leva a muitas incertezas, derivado a várias situações que acontecem, tais como acidentes, trânsito, ambiente político ou condições climatéricas inesperadas, podendo levar a mais custos na gestão da logística se estas não forem levadas em consideração.

A solução do VRP é uma forma de sobrevivência para as empresas, nomeadamente no setor de transportes ou de indústrias de alimentos, e pode trazer vários benefícios, tais como, **a poupança de tempo e de dinheiro, o aumento das margens de lucro e da satisfação do cliente** [26]**, entre outras.**

2.2 Agrupamento de pessoas

Uma das causas das grandes dificuldades na mobilidade, principalmente citadina é o volume de veículos privados a circular diariamente. Uma elevada percentagem dos veículos que circulam na cidade efetua viagens com apenas um passageiro, o que representa uma reduzida eficiência. Esta utilização ineficiente tem um grande impacto na fluidez das vias de trânsito, dificuldades relativamente a questões de estacionamento e um aumento da poluição. Agrupar pessoas em veículos que irão efetuar a mesma viagem ou percursos semelhantes, representa uma das formas de reduzir o impacto mencionado. Além disto há um impacto financeiro resultante da redução do custo da viagem por elemento. Existem incentivos financeiros e métodos de organização de vias para agrupar o máximo de pessoas num determinado veículo, tais como, ***High-occupancy vehicle lane, High-occupancy toll lane*** e ***Slugging lines*****.** O uso de um carro tem um custo relativamente alto, desde manutenção, consumos, desgaste a nível pneumático, entre outros. Se for considerada apenas uma pessoa por veículo o custo será muito maior, levando a que exista um maior congestionamento das vias, pois haverá um maior número de veículos nas mesmas e um maior impacto ao nível da poluição ambiental na sociedade.

O método ***High-occupancy vehicle lane (HoV)*** [27] incentiva os utilizadores a juntar o máximo de pessoas num veículo, ao fornecer-lhes vias de trânsito específicas para circular. Em estradas com elevado tráfego em veículos com um só ocupante, esta via especial facilita a circulação apresentando vantagem de poupança de tempo a quem partilha viagem. Em Portugal, existem corredores de circulação destinados a transporte público (autocarro e *táxi*), mas de uma forma genérica os **HoV** permitem o uso por veículos privados.

O método ***High-Occupancy toll lane*** [28] incentiva os utilizadores a juntar o máximo de pessoas num determinado veículo, reduzindo o preço da portagem. Um menor número de pessoas num determinado veículo, leva a um aumento do preço da portagem.

Outro método é o ***slugging lines*** [29] que é um sistema organizado em que as pessoas viajam para uma cidade para apanhar outros passageiros, mesmo que sejam completamente estranhos, ou seja, representa mais uma forma de incentivo para a junção de várias pessoas num veículo.

Enquanto que os métodos mencionados são suportados por estratégias de planeamento urbano e governamentais, é possível agrupar pessoas nos veículos por iniciativa pessoal. O *carpooling* é um método que junta várias pessoas no mesmo veículo, tipicamente com o mesmo destino ou destino próximo e que envolve a divisão dos custos das viagens entre os vários participantes. Independentemente de haver iniciativas como as já mencionadas, a divisão de custo é um grande impulsionador a esta escolha.

O presente trabalho utiliza o *clustering* como técnica para agrupar pessoas com proximidade espacial na sua localização de partida, impulsionando o método de *carpooling*.

2.2.1 Clustering

O ***Clustering*** ou **Análise de Agrupamento de dados** é o conjunto de técnicas de *data mining* que visa criar agrupamentos automáticos de dados segundo o seu grau de semelhança [30]. O critério de semelhança faz parte da definição do problema e é independente do algoritmo que vai ser usado. A cada conjunto de dados resultante do processo dá-se o nome de grupo, ou agrupamento (*cluster*) [30]. Esta técnica de análise de dados é cada vez mais usada para classificar elementos em grupos, de forma a entender quais os elementos de um conjunto de dados são semelhantes (encontram-se num *cluster*) e quais os elementos do conjunto são distintos entre si (encontram-se em *clusters* distintos) [31]. Os algoritmos de agrupamento de dados podem ser classificados em diferentes tipos: **particionados, hierárquicos, baseados em densidade, baseados em grelha** e ***spectral clustering***.

2.2.1.1 Métodos particionados

Estes métodos são considerados dos mais conhecidos e utilizados em *data mining*. Este método de agrupamento classifica as informações em vários grupos com base nas características e semelhanças dos dados. Este algoritmo obriga o utilizador a especificar o número de *clusters* que devem ser gerados. De uma forma muito sucinta, dado um conjunto de dados, *D*, contendo *n* objetos e *k*, como o número de *clusters* a formar, um algoritmo de particionamento organiza os objetos (*n*) em *k* partições, onde cada partição representa um *cluster* [32].

O algoritmo de *clustering* particionado tenta decompor um conjunto de *n* objetos em *k clusters* de modo a que as partições otimizem uma determinada função de critério. Cada *cluster* é representado pelo centro de gravidade (ou centroide) do *cluster*, como por exemplo, o K-means, ou pela instância mais próxima do centro de gravidade (medóide). Normalmente, *k* pontos são selecionados aleatoriamente e, em seguida, um esquema de realocação de forma iterativa redistribui pontos entre *clusters* para otimizar o critério de *clustering*. A minimização do critério de erro quadrado – soma das distâncias quadradas euclidianas de pontos do seu centróide do *cluster* mais próximo – é a mais utilizada. [33]

2.2.1.1.1 K-Means

O algoritmo K-means, também conhecido como *Hard C-means*, é um algoritmo iterativo que distribui os elementos de um conjunto de dados em *k clusters*, em que k é o número de *clusters* pretendido e definido no início da sua execução.

A execução do algoritmo envolve quatro etapas. O primeiro é a definição dos centros dos *clusters*, onde *k* pontos são escolhidos arbitrariamente e marcados como centro de cada *cluster*.

De seguida é calculada a distância de cada um dos restantes elementos do conjunto de dados aos centros de cada *cluster*. Cada elemento é atribuído ao *cluster* com o qual está mais próximo do seu centro.

Uma vez que o algoritmo tem por base uma escolha aleatória dos centros, não há garantia que a distribuição seja a melhor distribuição possível. Por esse motivo, o K-Means é um algoritmo iterativo. O aspeto iterativo do algoritmo existe no sentido de procurar o melhor resultado a cada iteração com alguns mecanismos para evitar terminar em ótimos locais. A qualidade da distribuição conseguida nos primeiros dois passos do algoritmo é definida por uma função de custo. O terceiro passo do algoritmo é o cálculo da função de custo da distribuição obtida, que consiste na soma das distâncias euclidianas de cada elemento ao centro do *cluster* a que pertence. Quanto menor for esta soma, mais próximos estarão os elementos aos centros dos *clusters*, representando uma boa distribuição por proximidade. Esta função de custo está representada na equação (1), onde *J* representa o valor de custo da distribuição, *c* o número de *clusters*, *Ji* o valor de custo de cada *cluster*, X_k cada elemento de um *cluster*, *k* o número de elementos do *cluster*, C_i o centro do *cluster* e G_i o conjunto de elementos do *cluster*.

$$J = \sum_{i=1}^{c} Ji = \sum_{i=1}^{c} \left(\sum_{k, x_k \epsilon G_i} \|x_k - c_i\|^2 \right) \quad (1)$$

Fonte: http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/Teses/Tese_Mest_Joao-Santos-Oliveira.pdf

Figura 1 - Cálculo do valor de uma função de custo

O resultado do cálculo é usado para comparar com o resultado de custo da iteração anterior. Caso o valor de custo seja menor que o da iteração anterior, temos uma melhoria da distribuição e permitimos que o algoritmo execute uma nova iteração, procurando melhorar mais. Caso o valor do custo aumente, temos uma distribuição pior. O algoritmo termina e assume a distribuição anterior como melhor execução e essa é retornada como distribuição final dos elementos por *clusters*.

De notar que quando é iniciada uma nova iteração do algoritmo há uma nova redefinição de centro de *cluster*. No entanto, esta definição não é aleatória como na primeira iteração. Para a definição de novos centros, calcula-se a média dos elementos de cada *cluster* que servirá de novo centro de cada *cluster*. Este cálculo é representado pela equação 2, onde c_i representa o novo centro do *cluster i*, G_i o conjunto de elementos do *cluster i* e x_k cada elemento do *cluster*.

$$c_i = \frac{1}{|G_i|} \sum_{k, x_k \in G_i} x_k \quad (2)$$

Fonte:http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/Teses/Tese_Mest_Joao-Santos-Oliveira.pdf

Figura 2 - Renovação dos centros de todos os *clusters*

Com os novos centros calculados, o algoritmo volta a efetuar a distribuição dos elementos pelos *clusters* e cálculo do custo da distribuição até não ser possível melhorar o custo.

Apesar de utilizar um método iterativo, o algoritmo continua a ter facilidade em ficar com ótimos locais na distribuição final. No sentido de contrariar esse aspeto, é possível executar múltiplas vezes o algoritmo, com diferentes distribuições aleatórias dos centros na primeira iteração. Depois de executar o algoritmo iterativo para cada combinação de centros aleatórios é escolhida a distribuição cuja combinação de centros resultou no menor custo [34]. Quanto mais execuções do algoritmo com centros de iniciais diferentes, maior é a probabilidade de obter melhores resultados e evitar ótimos locais.

2.2.1.1.2 Constrained K-Means

Um efeito do K-means por este usar a minimização das distâncias entre os pontos é que os *clusters* gerados poderão ter uma grande variabilidade no número de pontos contidos em cada agrupamento. No contexto do transporte há a necessidade de limitar o número de pontos de um *cluster* de modo a poder representar os potenciais passageiros de um veículo.

O *Constrained K-Means* é uma variação do K-means que altera o passo de atribuição de pontos a um *cluster*, transformando num problema de otimização de rede linear do tipo *Minimum Cost Flow* [35] [36] [37].

A vantagem de utilizar o Constrained K-means em relação ao K-means tradicional está inerente à possibilidade de indicar quer o número mínimo de pontos que um *cluster* pode ter, quer o número máximo de pontos, sendo que para além disso também garante a otimização da distância entre os pontos. Desta forma, esta variação do algoritmo pode ser usada para calcular *clusters* com número de pontos limitado aos passageiros de um só veículo (por exemplo, quando se pretende *clusters* com o máximo de 4 pontos). Não deixa de ser necessário indicar o número de *clusters* a criar a priori, mas evita grandes variações de densidade de pontos entre os *clusters*.

2.2.1.1.3 Fuzzy c-means clustering

O *Fuzzy Clustering* é um método de *clustering* difuso que generaliza os métodos de agrupamento de partição, permitindo que um indivíduo possa pertencer a mais do que um *cluster*. Ele é semelhante ao K-means pelo uso de uma função de minimização de distância. O conceito de *fuzzy* vem da possibilidade das fronteiras dos *clusters* sobreporem-se umas às outras. Esta sobreposição permite que um ponto junto de uma fronteira de um *cluster* possa estar dentro da área de outro *cluster* e daí fazer parte de ambos os *clusters*. O algoritmo sofre das mesmas questões que o K-means tradicional, nomeadamente a necessidade de definir o número de *clusters* a priori, sensibilidade de *outliers* e ser suscetível a otimizações locais [38].

2.2.1.2 Métodos Hierárquicos

Os métodos hierárquicos encontram-se divididos em dois tipos:

- **Agrupamento hierárquico aglomerativo:** Cada objeto inicialmente representa um *cluster* individual. Em seguida, os *clusters* são sucessivamente fundidos até que o *cluster* desejado seja obtido. É designado de abordagem de baixo para cima (*bottom up approach*).
    - Passos do algoritmo aglomerativo [38]:
        1. Fazer de cada ponto um *cluster* separado;
        2. Até que o agrupamento (*clustering*) seja satisfatório;
            a. Juntar os dois *clusters* com a menor distância entre si;
        3. Fim.
- **Agrupamento hierárquico divisivo:** Todos os objetos inicialmente pertencem a um *cluster*. Posteriormente, o *cluster* é dividido em *sub-clusters*, que são sucessivamente divididos em subgrupos próprios. Este processo continua até que a estrutura do *cluster* desejada é obtida. É designado de abordagem de cima para baixo (*top down approach*).

- Passos do algoritmo divisivo [38]:
    1. Construção de um único *cluster* contendo todos os pontos;
    2. Até que o agrupamento (*clustering*) seja satisfatório;
        a. Dividir o *cluster* que gera os dois componentes com a maior distância entre *cluster*;
    3. Fim.

Este tipo de algoritmo é considerado pouco robusto por ser sensível a ruído e *outliers*. Há uma grande tendência dos *clusters* formados terem formas esféricas derivado ao aspeto de divisão de agrupamento próprio da criação de partições. Ao contrário dos métodos particionados, o ciclo iterativo de aglomeração ou divisão de hierarquia não corrige potenciais erros na atribuição de pontos a *clusters* [38].

2.2.1.3 Métodos baseados em densidade

Os métodos baseados em densidade tentam encontrar *clusters* com base na densidade de pontos de dados numa região. A ideia principal de agrupamento baseado em densidade é que para cada instância de um *cluster* a vizinhança de um determinado raio (eps) tem que conter pelo menos um número de instâncias (MinPts) [39].

O algoritmo **DBSCAN** (*Density-Based Spatial Clustering of Application with Noise*) baseia-se na densidade em que os *clusters* crescem de acordo com um dado limiar [40] , em que o número de *clusters* é decidido dependendo dos dados que são fornecidos. Uma vez que se trata de um algoritmo de *clustering* baseado em densidade, alguns pontos podem não pertencer a nenhum *cluster*. Esta característica é diferente dos algoritmos particionados e hierárquicos, onde todos os elementos são considerados e obrigatoriamente pertencem a algum *cluster*. Isto pode ser explicado com a ajuda de dois parâmetros ***epsilon*** e ***min_points*** que são usados no algoritmo, sendo que o ***epsilon*** representa a distância máxima entre dois pontos para serem considerados vizinhos e o ***min_points*** corresponde ao número de vizinhos que um determinado ponto deve ter para ser classificado como um ponto principal. Os pontos podem ser divididos em três categorias [41]:

- ***Core Point*** se um ponto com pelo menos ***min_points*** pontos cuja distância em relação ao ponto estiver abaixo do limite definido por ***epsilon***;
- ***Border Point*** se um ponto não estiver próximo de pelo menos ***min_points*** pontos, mas estiver próximo o suficiente de um ou mais pontos centrais;

- ***Noise Point*** se os pontos não estiverem próximos o suficiente dos pontos principais para serem considerados ***border points*.** Os pontos de ruído são ignorados, ou seja, não pertencem a nenhum *cluster*.

O algoritmo DBSCAN é muito usado ao nível da densidade populacional e existem alguns aspetos que são relevantes e que valem a pena ser mencionados:

- É um algoritmo flexível, uma vez que é dinâmico em relação aos dados;
- Os parâmetros necessários para executar o algoritmo podem ser obtidos a partir dos próprios dados, usando o **DBSCAN adaptável** [42], tal como será abordado no subcapítulo 7.4**;**
- Proporciona um agrupamento mais intuitivo, pois é baseado na densidade não considerando pontos que não pertencem a nenhum lugar (*outliers*);
- Possui complexidade $O(n^2)$ [43], sendo *n* o número de pontos e é muito mais rápido do que comparado às técnicas tradicionais de *clustering* como o K-means cuja complexidade é O(IKn), sendo *n* o número de pontos , *k* o número de *clusters* e *I* o número de iterações de cálculo [44]. Já os métodos hierárquicos apresentam uma complexidade $O(n^3)$ [45].

2.2.1.4 Métodos baseados em Grelha

Os métodos baseados em grelha particionam o espaço em um número finito de células que formam uma estrutura de grade na qual todas as operações para agrupamento são realizadas. A principal vantagem deste método é o facto de apresentar um tempo de processamento rápido, a não necessidade de haver cálculos de distância e a facilidade de determinar quais *clusters* são vizinhos.

A grande vantagem dos métodos baseados em grelha deve-se à redução significativa da complexidade computacional nomeadamente para conjuntos de dados muito grandes [46].

Os passos básicos do algoritmo baseado em grelha são os seguintes:

1. Definir um conjunto de células em grelha;
2. Calcular a densidade das células;
3. Eliminar células com uma densidade menor que o limite;
4. Formar *clusters* de células contíguas [38].

2.2.1.5 Método Spectral Clustering

O método *Spectral Clustering* é um tipo de *clustering* baseado em grafos, que utiliza um grafo de semelhança como base. Aos *n* pontos que se pretende agrupar em *clusters* é calculado uma matriz

de semelhança *S* que descreve a semelhança entre os pontos. Cada valor da matriz s_{ij} é obtido por avaliação da semelhança entre o par de pontos x_i e x_j numa célula específica da mesma.

$$S= (s_{ij})\ i,j=1,\ldots,n.$$

$$s_{ij} = s(x_i,x_j)$$

Com base na avaliação de semelhança entre pontos, é construído um grafo de semelhança com uma matriz de adjacência com pesos, como por exemplo um grafo de *K-Nearest Neighbours*. De seguida são calculados a matriz Laplaciana *L* e os *eigenvectors* de *L*. Cada *eigenvector* passa a ser uma coluna de uma matriz *U*. Cada linha da matriz *U*, corresponde a um ponto do *dataset* e são calculados os *clusters* com base na informação de cada linha usando o K-means [38] [47].

As vantagens da utilização do *Spetral Clustering* é a sua facilidade de implementação e a oferta de bons resultados no *clustering* sendo razoavelmente rápido para conjuntos elevados de dados esparsos. Em termos de desvantagens, o *Spetral Clustering* utiliza o agrupamento K-Means na etapa final levando a que os *clusters* nem sempre sejam os mesmos, uma vez que estes podem variar de acordo com a escolha dos centroides iniciais. Para além disso é computacionalmente dispendioso para um conjunto de dados elevado, derivado ao facto de os *eigenvectos* e *eigenvalues* precisarem de serem calculados, e em seguida, recorrem ao *clustering* nesses vetores [48].

2.2.2 Comparação das técnicas de Clustering

Segundo João Simões [32], este efetuou uma comparação de várias técnicas de *clustering*, de onde foram aproveitadas algumas métricas que o mesmo analisou e que são relevantes para o projeto em questão:

- Capacidade de identificar *clusters* com formas aleatórias;
- Capacidade de identificar *clusters* em *datasets* com elevado volume de dados;
- Boa *performance* na obtenção de resultados, principalmente também em *datasets* com considerável volume de dados;
- Capacidade de lidar com ruído e conseguir que a sua presença não tenha impacto nos resultados obtidos;
- Parametrização inicial do algoritmo (a não obrigatoriedade de indicar o número de *clusters* finais que terão de ser gerados e estimação dos parâmetros iniciais).

Para o projeto em estudo, nomeadamente na fase de *clustering* de pedidos de transporte (tal como está apresentado no subcapítulo 1.3), não sabemos o número de *clusters* que devem ser gerados, o que torna a última característica relevante. A capacidade do algoritmo ser capaz de identificar *clusters* com várias formas aleatórias também é uma característica que se pretende, devido ao facto de não sabermos que tipo de forma terão os *clusters* e de se prever que as formas adotadas se correlacionam com o planeamento geográfico. Também será interessante analisar se o algoritmo a escolher consegue lidar com ruído e se este não afeta os resultados, sendo que o ruído neste domínio representa pontos muito distantes dos restantes e que condicionam a definição do *cluster* (exemplificando, podemos ter uma concentração de pontos próximos num raio de 1 quilómetro e um único ponto fora do grupo a 10 quilómetros desse grupo de pontos). Outra das métricas que também será importante analisar é uma boa *performance* computacional na identificação de *clusters*, uma vez que poderá utilizar *datasets* com elevado volume de dados, sendo crucial o cálculo dos *clusters* em tempo útil. Portanto qualquer categoria ou técnica que não cumpra alguma destas métricas é automaticamente excluído.

Verificou-se que as técnicas que pertencem aos métodos particionados não são capazes de identificar *clusters* em *datasets* com elevado volume de dados, não apresentando uma boa *performance* na obtenção de resultados sobretudo quando o volume de dados é elevado. Além disso outro aspeto relevante a mencionar é que os mesmos são sensíveis ao ruído, sendo que a sua presença influencia os resultados finais. Derivado a estes motivos e ao facto de ser necessário indicar o número de *clusters* previamente, decidiu-se não se utilizar nenhuma técnica referente a esta categoria numa primeira fase [32]. Numa segunda fase foi utilizada uma versão adaptada do K-means denominada de Constrained K-means para restringir o número mínimo e máximo de pontos por cada *cluster*, uma vez que um dos requisitos do projeto passava por cada oferta de boleia conter no mínimo 2 e no máximo 4 passageiros num determinado veículo. No que toca aos métodos de *Fuzzy Clustering* pertencentes à categoria dos métodos particionados e mais concretamente a técnica do *Fuzzy c-means* verificou-se que este admite que um ponto possa pertencer a mais que um *cluster* e para este problema em concreto os pontos são únicos e só podem pertencer a um só *cluster* [49]. Caso contrário, existe o risco desse ponto estar atribuído a mais do que uma viagem. Além disso esta técnica obriga à especificação do número de *clusters* no início do algoritmo e não consegue lidar com ruído e *outliers*, sendo também a mesma automaticamente rejeitada.

Também foram analisados os métodos hierárquicos, tendo-se verificado que os mesmos não possuem robustez e são sensíveis ao ruído e *outliers* pelo que podem influenciar negativamente os resultados [38], sendo logo também rejeitados.

Desta forma, verificou-se que os métodos baseados em densidade e os baseados em grelha eram aqueles que mais se ajustavam às necessidades do projeto. Posto isto, foram analisadas quais as técnicas mais recentes referentes a cada um dos métodos, tendo-se concluído serem o *Clique*, que

pertence à categoria dos métodos baseados em grelha, e o DBSCAN, que pertence aos métodos baseados em densidade [38]. Assim, tornou-se possível proceder à respetiva comparação e determinação de qual das duas se adequaria melhor ao projeto em questão.

Passando à categoria do *Spectral Clustering*, verificou-se que esta técnica apresentava dificuldade em trabalhar com ruído, podendo influenciar negativamente os resultados, sendo que para além disso necessita da especificação do número de *clusters* no início do algoritmo [47] e por isso também é rejeitada.

O método *Clique* têm uma complexidade temporal de O(n+kˆ2), inferior ao DBSCAN com O(n∗logn), e é mais escalável computacionalmente, mas menos adequado para um volume grande de dados. Para este projeto em concreto são apenas necessárias três dimensões, a latitude, a longitude e o tempo, sendo que uma delas (o tempo) é resolvida antes de ser aplicado o algoritmo de *clustering*. Agrupando previamente os dados por tempo, o conjunto de pontos a processar pelo algoritmo é limitado em termos de dimensionalidade, reduzindo o tempo necessário para obter os *clusters* a utilizar no pipeline de processamento descrito na secção 6.4. Ambos os algoritmos suportam múltiplas dimensões, mas o *Clique* suporta melhor um número elevado de dimensões. Porém, o DBSCAN é menos sensível a ruído que o *Clique*, o que se adequa bem ao domínio dos pedidos de transporte [38] .

Em conclusão, o DBSCAN é a técnica que mais se adequa como algoritmo de *clustering* para este projeto em concreto. A sua capacidade de considerar a densidade de pontos com formas aleatórias adequa-se bem à variabilidade que os pontos de *pickup* definidos pelos pedidos dos passageiros terão ao longo do tempo. A cada iteração do cálculo, não teremos previsão do número de pedidos a agrupar, o que significa que não há como definir o número de *clusters* a priori e é uma condição que o DBSCAN suporta bem. Finalmente devido à variação de posição e número de pontos de *pickup* a cada momento, alguns poderão ser considerados ruído ou *outliers* e o DBSCAN não sofre de sensibilidade com esse aspeto. Além disso, numa segunda fase foi efetuada uma pesquisa sobre algoritmos que pudessem ajudar a limitar o número mínimo e máximo de pontos por cada *cluster* e simultaneamente ter em conta a otimização entre pontos, e verificou-se que existe uma técnica de *clustering* adaptada do K-means denominada de Constrained K-means. Esta tem a capacidade de limitar o número mínimo e máximo de pontos por cada *cluster* e de ao mesmo tempo garantir a otimização da distância entre os pontos, tornando-se uma ajuda preciosa na concretização deste objetivo. Sendo assim, a cada *cluster* resultante do DBSCAN foi aplicado o Constrained K-means.

Estas conclusões estão de acordo com a restante literatura relacionada com a aplicação do *clustering* ao domínio de transporte o que reforça a sua adequação ao problema.

2.2.3 Clustering aplicado à Mobilidade

A importância do *clustering*, quando aplicado ao ecossistema da mobilidade, procura explorar padrões de viagem e direcionar passageiros de trânsito apenas utilizando dados dos cartões inteligentes [50]. Esses cartões inteligentes possuem transações individuais, ou seja, por utilizador, sendo que cada dia de trabalho foi usado para construir itinerários de viagem aplicando, como algoritmos de *clustering*, o DBSCAN e o K-means. Para isso, foi necessário recorrer ao DBSCAN, que foi utilizado para agrupar as últimas paragens de desembarque feitas pelos utilizadores. Para além disso, o mesmo algoritmo também agrupa as paragens de partida e o tempo em que um passageiro normalmente embarca num determinado veículo. Por fim, através do K-means, os utilizadores foram classificados e divididos em vários grupos, tais como "passageiros de trânsito", "passageiros regulares" e "passageiros irregulares".

Outro trabalho de investigação foi utilizado com recurso a ferramentas de *data-mining*, tendo sido realizado pelos autores *Catherine Morency*, *Martin Trépanier* e *Bruno Agard* [51] e apresentam diversas medidas relacionadas à variabilidade do comportamento de viagem pelos utilizadores de transportes públicos. Estas análises foram realizadas utilizando-se, mais uma vez, o cartão inteligente tendo sido recolhidas num período de dez meses. A análise realizada foi principalmente para entender a diferença em termos de embarque por dia e novas paragens, frequentes com os dias de viagem na rede de transportes públicos, e para isso os autores recorreram ao algoritmo K-means para construir *clusters* de dias que apresentam padrões de tempo semelhantes de embarque na rede de transporte público, a fim de entender se os passageiros têm comportamentos regulares de viagem e se os dias diferem significativamente. As experiências realizadas, mostraram que os comportamentos dos utilizadores habituais de transportes públicos evoluem ao longo do tempo, tanto em termos de paragens frequentes como em relação às horas de embarque, levando à conclusão de que a mudança de comportamento varia dependendo dos vários tipos de utilizadores existentes [51]. Outro estudo concentra-se no transporte de táxis em Taiwan [52], ajudando-os a encontrar locais de alta densidade. Isso deveu-se ao facto de muitas vezes, os motoristas não saberem onde os passageiros estão, levando a que eles gastem muito tempo a conduzir veículos desocupados, revelando assim uma grande perda para o negócio. Deste modo, foi essencial procurar uma solução de forma a que os taxistas conseguissem encontrar potenciais clientes e fossem identificadas as localizações de futuros clientes. Com base numa certa posição de referência, condição climatérica, tempo real e histórico de pedidos foram calculados *hotspots* que podem ser previstos e recomendados derivado à utilização de técnicas de *data-mining*. Neste caso, o DBSCAN serviu para agrupar as coordenadas das localizações dos clientes de acordo com a distância ao nível espacial, levando a que para cada *cluster* identificado, as estradas correspondentes fossem devidamente associadas. Com o resultado desta análise, os taxistas podem ajudar as suas estratégias e decidir para onde ir de forma a apanhar mais passageiros.

Outro trabalho de investigação publicado na GeoInfo [53], surgiu com o desenvolvimento de uma aplicação baseada no algoritmo de *clustering* DBSCAN com o objetivo de reduzir a perda diária de tempo na movimentação de um grande número de pessoas para um lugar comum. Os *clusters* são criados com base em vários atributos, tais como o tempo de partida de cada pessoa da sua residência, o destino final, e as suas localizações de partida e de chegada. As pessoas que compõem um dado *cluster* são transportadas num veículo alocado de acordo com o tamanho do mesmo. Foi então realizado um caso de estudo com um certo grupo de pessoas deslocando-se até ao campus II do Centro Federal de Educação Tecnológica de Minas Gerais utilizando um simulador de trânsito para medir o tempo individual e global que as pessoas precisam para se moverem.

Utilizando os conceitos de *clustering* sobre a latitude, longitude e tempo de movimentação dos indivíduos foram recolhidos os dados e processados de forma a agrupar pessoas com tempo de movimentação e localização de origem e destino semelhantes, levando a que fosse possível encontrar rotas que possam transportar esses grupos de pessoas de uma forma eficaz. De forma a ser encontrada uma solução que melhorasse o tempo diário de movimentação das pessoas, o problema foi subdividido em duas partes. Primeiro, as pessoas foram reunidas em grupos, de acordo com o algoritmo DBSCAN. Para cada *cluster*, um centroide foi calculado e definido como ponto de partida para as pessoas nesse *cluster* determinado. Assim sendo, na primeira parte da rota, cada pessoa caminha até esse ponto central, passando um certo tempo inicial da caminhada, tendo sido estabelecido que o tempo da mesma tinha de ser inferior a vinte minutos. A distância de cada indivíduo ao centro do *cluster* foi calculada utilizando-se a função *haversine* [1]e o tempo de movimentação estimado até ao centroide foi definido como o tempo médio da caminhada de 4,82 kilómetros/hora por cada pessoa. Posteriormente, na segunda parte do problema, depois de todas as pessoas caminharem para o centroide do seu respetivo *cluster*, a rota seria estabelecida a partir de um ou mais veículos. O tipo de veículo foi escolhido tendo em conta o número de pessoas alocadas em cada *cluster*, sendo que até 5 pessoas seriam deslocadas de carro, entre 6 a 15 pessoas seriam de carrinha, entre 16 a 40 seriam de autocarro ou sendo mais de 40, as pessoas iriam ser distribuídas entre vários veículos, dependendo do número total das mesmas. Após a escolha do veículo mais adequado, o resto da rota é efetuado sem qualquer tipo de paragem até ao destino final e o tempo de movimentação de cada veículo até ao destino final foi calculado com recurso aos dados do Google Maps, obtendo assim o tempo total que a mesma iria demorar [54].

Outro estudo propôs um sistema de transporte a pedido (*on demand*) com recurso ao *clustering*, tendo como principal objetivo ajudar os taxistas a recolher e transportar os passageiros da sua localização de partida para a sua localização de destino pretendida e simultaneamente andar pela estrada de forma a encontrar o próximo cliente [55]. Foram recolhidos dados de 1100 condutores em *Harbin*, uma cidade localizada na China. Estes dados contêm várias informações, tais como:

[1] *Haversine* – Fórmula usada para calcular a distância do grande círculo entre dois pontos, ou seja, a menor distância sobre a superfície da Terra (https://www.movable-type.co.uk/scripts/latlong.html)

- **tempo**, que indica a data e hora quando foram gravados os registos;
- **latitude** e **longitude** que especificam a localização do veículo do táxi;
- **velocidade,** que indica a velocidade média do veículo;
- **orientação,** que representa a direção em graus;
- **estado da viagem,** que é um valor booleano que indica se o táxi se encontra ocupado pelos passageiros.

Este estudo teve como principal objetivo ajudar os taxistas a recolher e transportar os passageiros, da sua localização de partida para a sua localização de destino pretendida e simultaneamente andar pela estrada de forma a encontrar o próximo cliente. Foi então relevante perceber qual é a área da cidade que pode atrair um maior número de pessoas e qual é a área espacial da mesma. Desta forma, decidiram utilizar como técnica de *clustering* o DBSCAN com o intuito de agrupar locais de embarque e desembarque por parte dos passageiros. A sua utilização trouxe vários benefícios, tais como o facto de conseguirem classificar locais num *cluster* com alta densidade, conseguirem encontrar locais específicos na rede rodoviária para cada *cluster* e conseguirem lidar bem com pontos de ruído.

2.3 Soluções comerciais / Concorrentes

Nesta secção irão ser abordadas algumas soluções comerciais/concorrentes de *mobility-as-a-service* que se encontram implementadas no mercado.

2.3.1 Whim

O ***Whim*** é uma aplicação que dá acesso à melhor opção de transporte a cada instante, dependendo da preferência do utilizador. Por exemplo se um utilizador preferir viajar de transporte público compra uma passagem e se eventualmente quiser ter acesso a um carro, pode o ter sem se preocupar com questões de seguro ou taxas de manutenção [56].

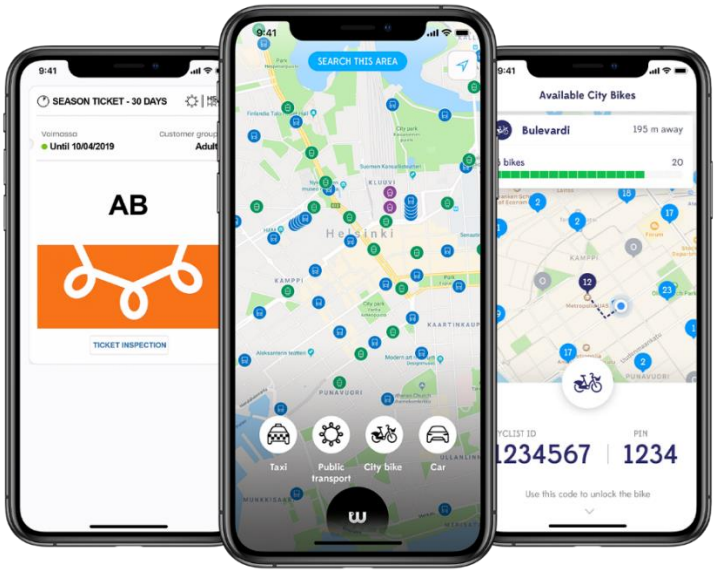
Esta aplicação foi aplicada na área de Helsínquia em 2016 e atualmente encontra-se presente em determinadas regiões de vários países, designadamente, Áustria (Viena), Finlândia (Região de Helsínquia), Reino Unido (Birmingham), Bélgica (Antuérpia). É baseada no conceito de *Mobility as a Service*[2] (MaaS), combinando uma vasta variedade de opções de transporte, tais como, metro, comboio, autocarro, aluguer de carros ou bicicletas, fornecendo um melhor suporte para viajar

[2] *Mobility as a Service* – É a integração de várias formas de serviços de transporte num único serviço de mobilidade acessível a pedido (https://maas-alliance.eu/homepage/what-is-maas/).

nesta área e a preços bastante acessíveis. Para além disso a aplicação dispõe de um pagamento eletrónico simples, evitando o uso de carteiras ou tempo na deslocação de um utilizador a um multibanco para efetuar o devido pagamento.

Além do que já foi mencionado anteriormente, a aplicação contém vários planos: ***Whim Urban 30, Whim Weekend, Whim Unlimited, Whim to Go***. O utilizador pode obter um plano com tudo incluído ou simplesmente pagar conforme o seu uso (sem assumir nenhum tipo de compromisso), sem ter de pagar uma taxa de inscrição quando adquire o mesmo [56].

Um exemplo da interface da solução comercial *Whim* pode ser visualizada na figura 3.



Fonte: https://computerwelt.at/printausgabe/mobilitaets-app-whim-in-wien-gestartet/

Figura 3 - Interface da aplicação ***Whim***

2.3.2 Moovel

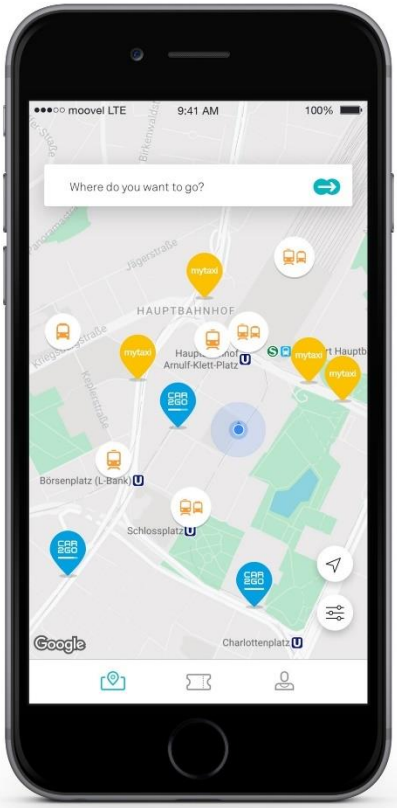
O ***Moovel*** é uma aplicação gratuita e encontra-se disponível na Alemanha e nos Estados Unidos da América, defendendo que não se deve gastar muito tempo em pequenas viagens, nem em pensar como fazê-las. Desta forma, o ***Moovel*** possui um sistema de seguimento de uma rota para vários tipos de transporte, tais como, metro, comboio, autocarro, avião, Uber, táxi e aluguer de carros e encontra-se disponível para Android e iOS [57].

Esta aplicação foi essencialmente construída para viagens pequenas. Além disso, tornou-se possível pagar viagens na aplicação através do *PayPal*, sendo para isso apenas necessário que o utilizador adicione a conta da mesma à aplicação como forma de pagamento. Ao efetuar a reserva, o utilizador pode optar se o pagamento deve ser confirmado com o seu PIN pessoal da aplicação ou através da sua própria impressão digital, facilitando o processo de compra de uma viagem [58].

O principal objetivo da mesma é rentabilizar e promover o uso de transporte público, de forma a torná-lo cada vez mais o meio de transporte usado, agrupando assim o máximo de pessoas num determinado veículo, dependendo da sua capacidade, levando a que haja uma redução do congestionamento das vias e uma maior fluidez nas mesmas.

O ***Moovel*** futuramente irá chamar-se ***ReachNow***, com a sua integração na BMW Group e Daimler AG. Além da aplicação móvel mencionada, irá fornecer um conjunto de ferramentas destinadas às cidades e operadores para a gestão das operações.

Um exemplo da interface da aplicação móvel *Moovel* pode ser visto na figura 4.



Fonte: https://pictures.attention-ngn.com/portal/35/375149/products/1498747529.6141_115_o.jpg

Figura 4 - Interface da aplicação ***Moovel***

2.3.3 Moovit MaaS Solution

As soluções de mobilidade como serviço do ***Moovit*** (*Mobility-as-a-Service*(MaaS)) ajudam empresas e governos a melhorar a mobilidade urbana. Com mais de 585 milhões de utilizadores em mais de 2900 cidades em mais de 90 países, o ***Moovit***, fabricante da aplicação de mobilidade

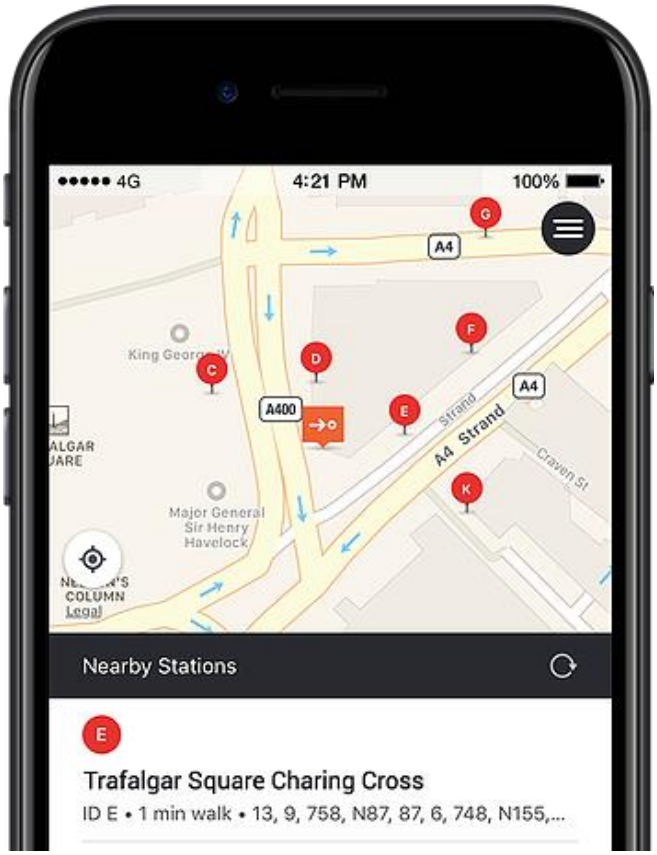
urbana mais popular do mundo, possui e opera o maior repositório de dados de trânsito do mundo, acumulando cinco bilhões de dados anónimos por dia.

A plataforma de mobilidade como serviço do ***Moovit*** fornece uma solução completa de software para governos e municípios que desejam oferecer MaaS aos seus cidadãos. A plataforma abrange todos os aspetos da viagem diária de um viajante, desde alertas de trânsito, planeamento de rotas multimodais (metro, comboio, bicicletas, *scooters*, Uber) e a inclusão de pagamento móvel.

Para os utilizadores finais dos transportes, a ***Moovit*** fornece uma aplicação móvel que facilita o acesso aos horários e itinerários de transporte público, incluindo o planeamento de rota, bem como a informação do estado dos transportes e opções alternativas. Existe também a *Web App* com uma vista mais detalhada e opções semelhantes. Em Portugal, é possível ver informação das cidades de **Bragança, Coimbra, Funchal, Leiria, Lisboa, Portimão, Porto, Braga e Vila Real.**

A ***Moovit*** inclui ainda ferramentas para operadoras e cidades nomeadamente o ***Moovit Urban Mobility Analytics*** **(MUMA)**, uma plataforma de análise de dados de mobilidade que permite a operadores compreender como as pessoas se movimentam pelas suas cidades e o ***Moovit TimePro***, que é um sistema de monitorização e análise de dados de mobilidade da cidade em tempo real [59].

Um exemplo da interface da aplicação *Moovit* pode ser observado na figura 5.



Fonte: https://mic.org.il/en/solutions/moovit/

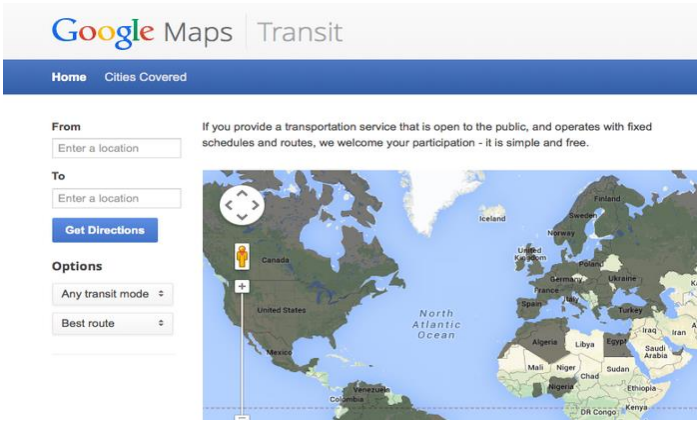
Figura 5 - Interface da aplicação ***Moovit MaaS Solution***

2.3.4 Google Transit

O **Google Transit** permite que os utilizadores visualizem opções de transporte público no Google Maps para ajudar os mesmos a planear melhor as suas rotas para chegarem ao seu destino. Ao combinar dados de agendamento e rota com o poder do Google Maps, as suas informações de trânsito público ficam facilmente acessíveis a milhões de utilizadores do Google em vários idiomas, tanto em computadores como em dispositivos móveis.

A utilização desta ferramenta traz vários benefícios, tais como, a simplificação do planeamento de viagens para os condutores com informações de paragens, rota, cronograma e tarifas integradas no Google Maps [60]. Além disso, o Google Transit solicita que os serviços de transporte aberto ao público com horários e rotas fixas participem, sendo simples e gratuito, levando a que haja uma maior visibilidade para os mesmos.

Como se pode visualizar na figura 6, um determinado utilizador seleciona um ponto de partida (ou até a partir da localização GPS, a aplicação reconhece de forma automática a mesma e coloca como ponto inicial, caso o utilizador assim o pretenda) e um ponto de destino. A aplicação dispõe, não só de várias opções de transporte, designadamente, **qualquer um dos modos, autocarros, metro, comboio, entre outros**, como também de filtros como o **melhor caminho, menos transferências ou menos caminhadas**.



Fonte: https://maps.google.com/landing/transit/index.html

Figura 6 - Interface da aplicação ***Google Transit***

Posteriormente, a aplicação é reencaminhada para o ***Google Maps***, onde aparece o devido planeamento de rota, de acordo com o ponto de partida e de destino introduzido pelo utilizador inicialmente, juntamente com o tempo que irá demorar essa viagem, conforme se pode verificar na figura 7.

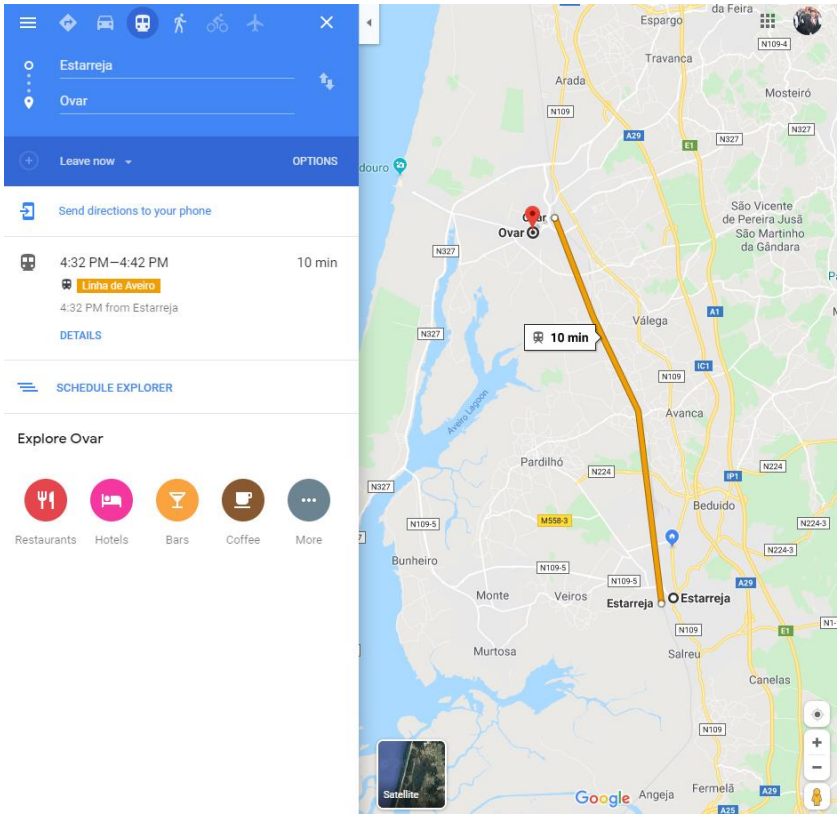


Figura 7 - Reencaminhamento para o ***Google Maps*** após ter sido efetuado um pedido por parte de um utilizador no ***Google Transit***

2.3.5 Transit

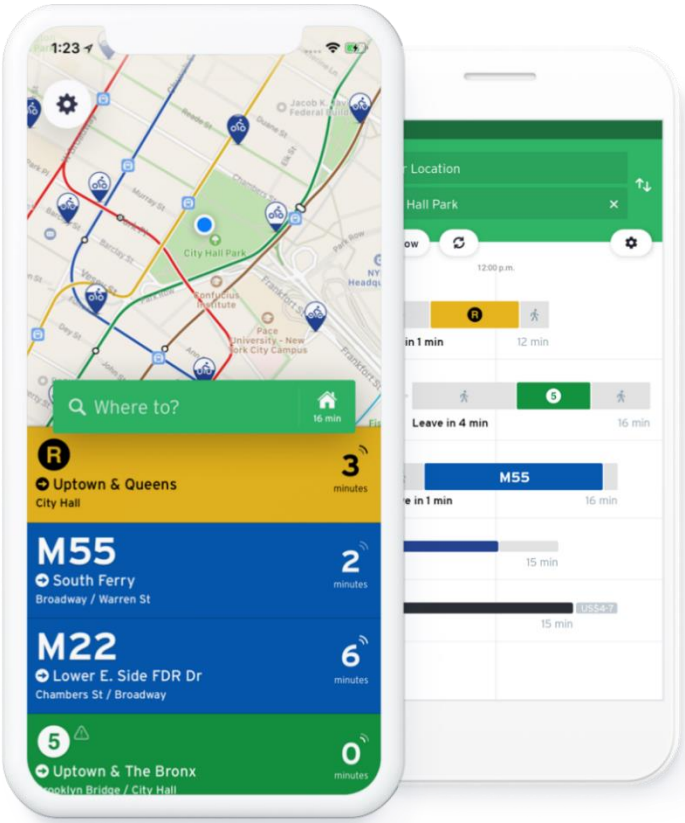
O ***Transit*** é uma aplicação móvel que fornece dados de transporte público em tempo real e foi desenvolvido em Montreal, região de Quebec no Canadá por Sam Vermette e Guillaume Campagna. A aplicação funciona em mais de 175 áreas metropolitanas e é compatível com Android e iOS.

A mesma oferece aos utilizadores horários e alertas para vários tipos de transporte quando disponíveis (a pé, metro, comboio, autocarro, bicicleta, *scooter*). Para além disso, a aplicação fornece aos utilizadores um sistema com código de cores que associa as mesmas a determinados modos de transporte. Em fevereiro de 2019, foi lançada uma atualização que permite aos utilizadores consultar horários de autocarros e comboios, assim como verificar se existem bicicletas disponíveis para aluguer. Outra das grandes vantagens ao utilizar esta aplicação é o facto do utilizador, mesmo offline, poder encontrar os locais de transporte público mais próximos da sua localização, otimizando o planeamento da sua rota. Além dos horários, a aplicação inclui ainda métodos de desbloqueio das bicicletas e trotinetes da cidade, requisição de viagens em serviço de

táxi, Uber e Lyft e também a localização de veículos disponíveis para *carshare*. O ***Transit*** tem neste modo de operação um claro foco no transporte multimodal.

O principal objetivo é minimizar a necessidade dos utilizadores possuírem veículos nas cidades, reduzindo assim os custos para cada utilizador, a poluição ambiental e, consequentemente o congestionamento do trânsito [61] [62].

Um exemplo da interface do *Transit* pode ser observado na figura 8.



Fonte: https://transitapp.com/

Figura 8 - Interface da aplicação ***Transit***

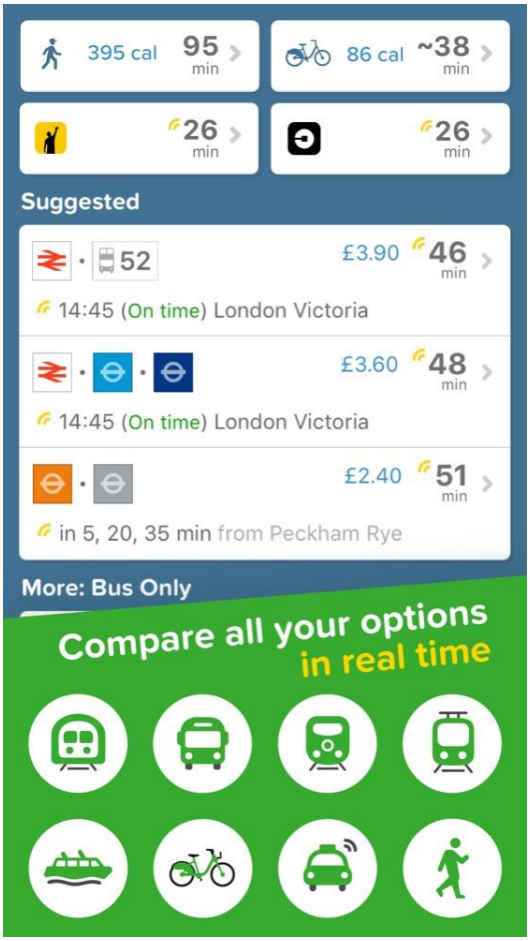
2.3.6 Citymapper

O ***Citymapper*** é uma aplicação de transporte público e serviço de mapeamento que integra vários modos de transporte à semelhança do ***Transit***. A aplicação é grátis e concorre com aplicações móveis gratuitas, tais como, o **Google Maps** e o **Apple Maps** [63]**.** Através desta aplicação, o utilizador não só consegue saber os melhores trajetos a percorrer num determinado horário sendo

os mesmos rastreados sempre em tempo real, como também usá-lo para planear viagens e saber informações sobre o clima e o horário/tempo do lugar.

Além disso, ainda consegue traçar as rotas com facilidade e mostrar o trânsito em tempo real, mostrar preços das viagens do transporte público, calcular calorias de trajetos feitos a pé ou de bicicleta oferecendo uma interface amigável e intuitiva para o utilizador [64].

Um exemplo da interface da aplicação do *Citymapper* pode ser visualizado na figura 9.



Fonte: https://citymapper.com/london

Figura 9 - Interface da aplicação ***Citymapper***

2.3.7 Blablacar

A **BlaBlaCar** [65] é considerada uma das principais aplicações de partilha de viagens em todo o mundo, sendo possível viajar para qualquer lugar. Esta possui várias vantagens tais como a acessibilidade de preços e facilidade de uso. Qualquer condutor pode publicar a viagem que pretende partilhar, indicando um local de origem e destino, pontos intermédios caso pretenda e o

horário da viagem. Como utilizador viajante, é possível no site pesquisar por origem e destino e um horário. O site lista as viagens disponíveis publicadas por outros condutores.

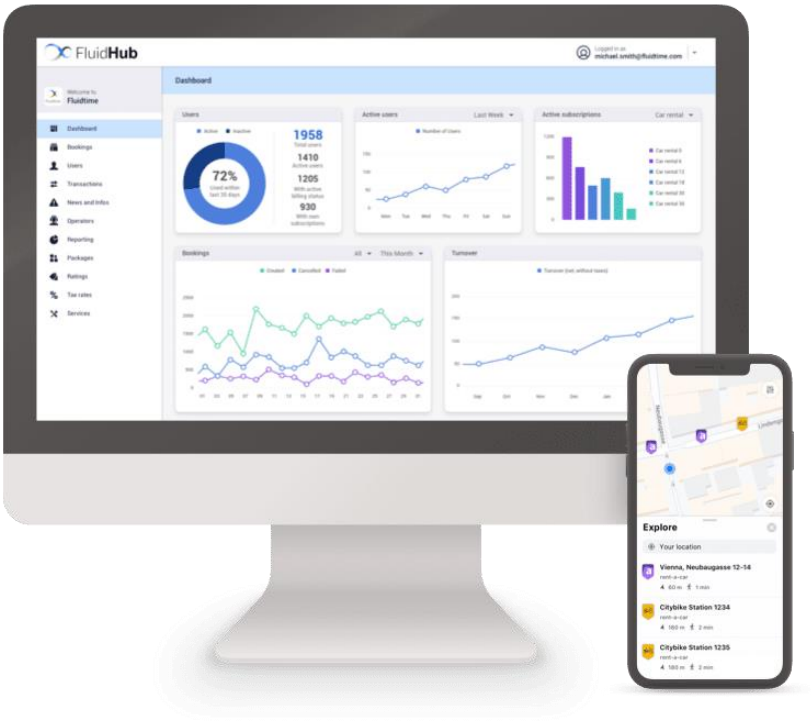
Um exemplo de uma viagem disponível na aplicação BlaBlaCar encontra-se disponível na figura 10.

Figura 10 - Interface da aplicação ***Blablacar***

2.3.8 FluidTime

Enquanto que as soluções anteriormente apresentadas, são aplicações com marca própria e com suporte para várias cidades, vendidas como serviço às cidades e aos seus utilizadores, a ***FluidTime*** é uma solução MaaS que pode ser customizada para cada cidade que a compra. A solução é composta por dois produtos. O primeiro, o ***FluidHub*** [66] é um *backend* com toda a funcionalidade de gestão do serviço incluindo *dashboards* e análise de dados, definição de preços dinâmicos, gestão de informações de horários e rotas de transportes e pontos de interesses, gestão de utilizadores e comunicação com os utilizadores. O segundo produto é o ***FluidGo***, uma aplicação móvel para os utilizadores que permite a pesquisa e pedido de transporte, bem como o pagamento do serviço seja por viagem individual ou subscrição.

Um exemplo da interface da solução *FluidTime* pode ser verificada na figura 11.



Fonte: https://www.fluidtime.com/wp-content/uploads/Fluidtime_Home/FluidHub_Fluidtime_Home_Header.jpg

Figura 11 - Interface da aplicação *FluidHub*

2.4 Conclusões sobre o Estado da Arte

Vimos que o contexto deste trabalho era o facto de pessoas que habitam em meios isolados não conseguirem transporte ou apresentarem dificuldades em se deslocar para o meio urbano. Além disso existem vários problemas, tais como, a existência de uma grande quantidade de veículos a circular nas vias de trânsito, o que causa grandes dificuldades na mobilidade, uma elevada percentagem de veículos a circular com apenas um passageiro, levando a que haja consequentemente um impacto negativo na fluidez das vias de trânsito, dificuldades no estacionamento, aumento da poluição e elevados custos. Através do estado da arte conseguiu-se perceber que já existem algumas técnicas que incentivam o agrupamento de pessoas num determinado veículo e que trazem várias vantagens às aplicações que recorrerem a estas técnicas.

Para ajudar a resolução deste problema e para se conseguir agrupar o máximo de pessoas num veículo podemos utilizar o *clustering*. Foi efetuada uma comparação entre várias categorias de *clustering* mais recentes, tendo em conta várias métricas, de forma a poder ser escolhida aquela

que melhor se adequa para o presente trabalho, sendo que a escolha da mesma foi complementada com a revisão da literatura. Concluiu-se que o DBSCAN, que pertence à categoria dos métodos baseados em densidade, seria a melhor opção para se efetuar o agrupamento de pedidos de transporte por localização de partida. Contudo esta solução não conseguia limitar o número máximo de pessoas num veículo ligeiro porque não tinha nada que conseguisse restringir o número máximo de pontos por cada *cluster*. Podiam aparecer *clusters* com mais de quatro pontos, excedendo assim o número de lugares de um veículo ligeiro. Foi efetuada uma pesquisa sobre algoritmos que pudessem então ajudar a restringir esse número máximo de pontos por cada *cluster*, tendo em conta a relação entre pontos e verificou-se que existia uma técnica de *clustering* adaptada do K-means denominada *constrained K-means*. Esta tem a capacidade de limitar o número mínimo e máximo de pontos por cada *cluster* e de ao mesmo tempo garantir a otimização da distância entre os pontos, tornando-se uma ajuda preciosa na concretização deste objetivo. Sendo assim, a cada *cluster* resultante do DBSCAN será aplicado o *constrained K-means*.

Foram abordados ainda vários conceitos importantes para o desenvolvimento deste projeto e algumas soluções já existentes no mercado sobre *Transport On-Demand*, de forma a ser possível analisar quais as principais diferenças que apresentavam relativamente às do presente trabalho. Das soluções que foram abordadas, as que apresentam um contexto mais semelhante com o presente trabalho são:

- *Moovit On Demand* que é um serviço do *Moovit MaaS Solution*, cujo principal objetivo passa por reduzir o congestionamento e resolver desafios de *first/last mile*, atuando em Portugal apenas em cidades como Bragança, Coimbra, Funchal, Leiria, Lisboa, Portimão, Porto, Braga e Vila Real;
- *Blablacar*, considerado o site nº 1 de *carpooling* da Europa, sendo possível viajar para qualquer lugar.

Estas duas soluções são semelhantes ao presente trabalho no sentido em que são dois serviços de mobilidade *on-demand* na qual pretendem levar passageiros desde o seu ponto de partida até ao seu ponto de destino.

A grande diferença do presente trabalho para a do *Moovit On Demand* é o facto de resolver problemas de *carpooling* e o facto de trazer pessoas de meios isolados para meios urbanos. Já a diferença para o *Blablacar*, prende-se com o facto de esta solução não garantir que haja uma otimização do número de passageiros, levando a que o condutor possa, por exemplo, num veículo com quatro lugares, levar apenas um passageiro, revelando uma grande ineficiência e acarretando mais custos para o mesmo.

3 Metodologia e planeamento

Neste capítulo pretende-se realizar uma descrição da metodologia utilizada ao longo do desenvolvimento do projeto. Além disso, pretende-se fazer uma breve comparação entre o planeamento inicial do estágio e o plano efetivamente cumprido.

3.1 Metodologia

Durante todo o estágio, foi utilizada a metodologia *Agile*, mais concretamente a framework *Scrum*. A metodologia Ágil surgiu devido à necessidade do mercado em atender às necessidades dos clientes. O desenvolvimento de software é baseado no desenvolvimento iterativo, em que os requisitos e as soluções evoluem com a colaboração de equipas multifuncionais auto-organizadas.

O desenvolvimento ágil utiliza uma abordagem de planeamento incremental e muito iterativa. Enquanto que nos métodos tradicionais, todas as etapas do projeto são documentadas detalhadamente, desde o início até ao fim do projeto, esse processo no método ágil é realizado em etapas curtas, chamadas iterações. Um elemento chave da metodologia Ágil é a "valorização de indivíduos e iterações sobre os processos e ferramentas".

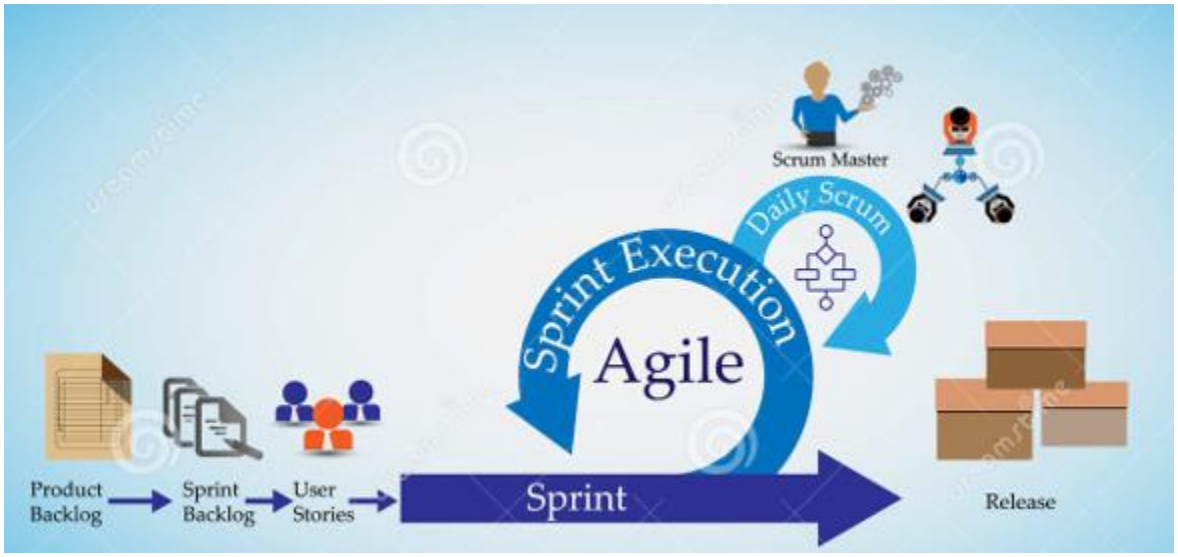
Este método tem como princípios [67]:

- A maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor;
- Aceitar alterações de requisitos, mesmo no fim do desenvolvimento. Processos ágeis adequam-se a mudanças, para que o cliente possa tirar vantagens competitivas;
- Permite ao cliente ter uma maior proximidade com o projeto e, caso seja necessário, fazer alguma alteração. Não se perde o trabalho realizado e permite ao cliente estar sempre a par do que se vai passando;
- Encurtamento do ciclo *"feedback"*, que permite obter a validação do trabalho efetuado mais cedo;
- A maneira mais eficiente e efetiva de transmitir informações é conversar pessoalmente.

Segundo o **Manifesto Ágil** (declaração de princípios que fundamentam o desenvolvimento ágil de software) [67], os valores relacionados com o desenvolvimento ágil de software, têm como base as seguintes premissas:

- Indivíduos e iterações são mais importantes do que processos e ferramentas;
- Software funcional importa mais do que documentação abrangente;
- A colaboração do cliente vale mais do que a negociação de contratos;
- Responder a mudanças é melhor do que seguir um plano.

Isso não significa, contudo, que processos, documentação, contratos e planos sejam afastados pelas metodologias ágeis. O que se verifica é que, por regra, têm uma importância secundária no processo de desenvolvimento de *software*. Como foi referido inicialmente ao longo do estágio foi adotado o SCRUM, uma framework pertencente à metodologia Ágil. Um exemplo deste tipo de metodologia pode ser visualizado através da figura 12.



Fonte: https://pt.dreamstime.com/ilustra%C3%A7%C3%A3o-stock-conceito-do-ciclo-de-vida-dodesenvolvimento-do-scrum-e-da-metodologia-%C3%A1gil-image90479166 [Acedido em junho de 2020]

Figura 12 - Metodologia Ágil

A framework SCRUM é caracterizada por ciclos, denominados como *sprint.* A equipa teve um ***Scrum* Master** (orientador da empresa), pessoa responsável por toda a gestão do projeto, designadamente por gerir o tempo de desenvolvimento das tarefas, organizar as reuniões e esclarecer dúvidas. Além disso também houve um ***ProductOwner*** (orientador da empresa) que forneceu todas as informações necessárias sobre o produto a desenvolver e quais os requisitos que eram prioritários, sendo que o estagiário foi responsável pelo desenvolvimento do projeto.

As funcionalidades a serem implementadas em cada *sprint* foram mantidas num repositório no GitLab[3], denominado ***Product Backlog*.** No que toca à especificação de requisitos, houve uma constante interação entre o orientador e estagiário, de forma a que cada um pudesse sugerir e transmitir a sua opinião, sendo este um aspeto extremamente relevante para a implementação e o crescimento do projeto.

Os requisitos mais relevantes (prioritários) foram colocados no ***Sprint Backlog*** para dar início a cada *sprint* com duração de duas semanas, que seguirá o processo de desenvolvimento da figura 13.

Diariamente realizou-se o *Daily Scrum* entre o estagiário e orientador de forma a poder verificar-se o estado de progresso do trabalho, identificar possíveis dificuldades durante o desenvolvimento levando assim a uma melhor comunicação entre a equipa. De salientar que a equipa neste projeto foi apenas composta pelo orientador da empresa e pelo estagiário.

Antes de se iniciar uma nova *sprint*, eram feitas reuniões entre a equipa de forma a poder ser revisto todo o trabalho realizado até ao momento e priorizar os requisitos subsequentes a serem implementados no próximo *sprint*.

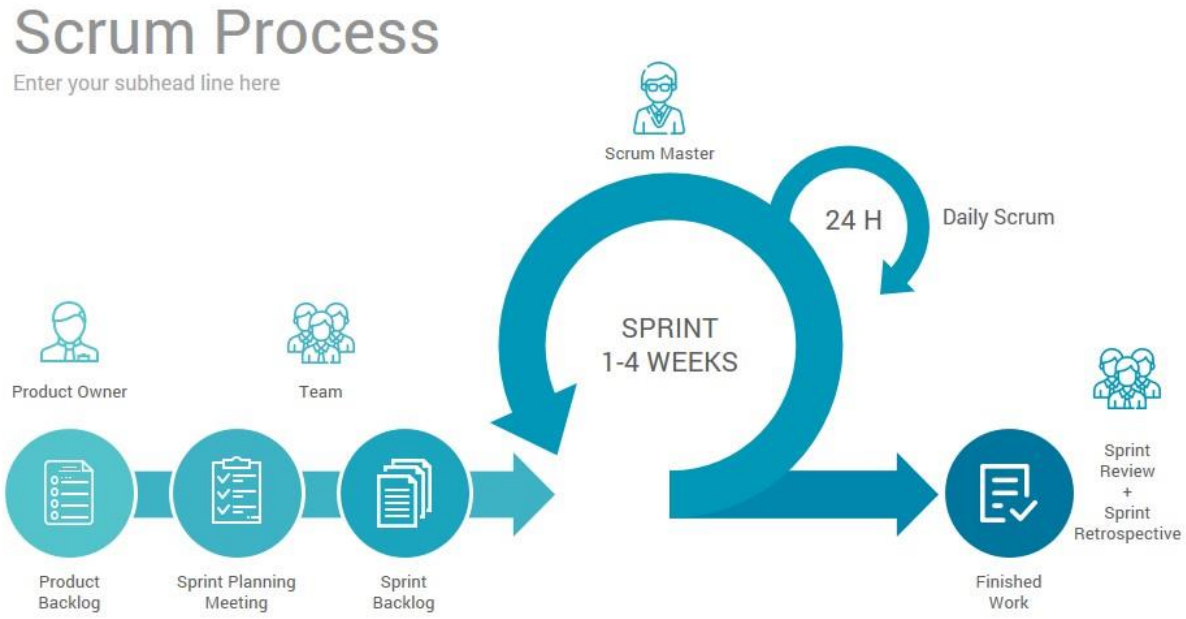


Figura 13 - Processo do *Scrum*

[3] GitLab – Esta ferramenta irá ser apresentada no próximo capítulo.

De notar que, apesar da pandemia o trabalho desenvolveu-se com normalidade. Todo o trabalho que habitualmente era feito presencialmente, passou a ser feito remotamente mantendo-se todas as etapas do *Scrum* mencionadas.

3.2 Planeamento

Derivado ao facto de o estágio encontrar-se dividido em dois semestres, o trabalho foi divido em duas subsecções, uma correspondente ao 1º Semestre e a outra correspondente ao 2º Semestre. No final de cada uma das subsecções é apresentado um diagrama de *Gantt* correspondente ao planeamento de cada um dos semestres.

3.2.1 1º Semestre

No que diz respeito ao 1º Semestre, conforme está demonstrado na figura 14, foi dividido da seguinte forma:

- **Estudo do problema e elaboração do estudo do Estado da Arte:** Neste período, foram analisados conceitos importantes para o desenvolvimento do projeto, assim como algumas soluções já existentes no mercado e que estão englobados no tema *Transport On Demand*. Além disso, ainda, foram analisadas várias tecnologias de forma a optar por aquelas que melhor se ajustavam ao projeto.
- **Levantamento e especificação de requisitos:** Nesta fase, foi efetuado o levantamento de requisitos da solução, tendo estes sido devidamente validados por parte do orientador da empresa confirmando se os mesmos se ajustavam às necessidades do projeto.
- **Estudo e definição da arquitetura do projeto:** Durante este período, foi definida a arquitetura lógica do sistema.

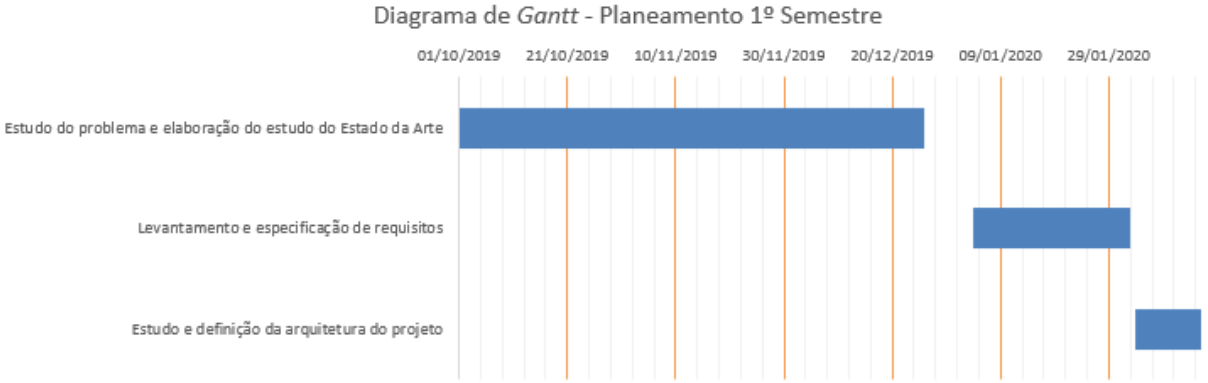


Figura 14 - Diagrama de *Gantt* - Planeamento 1º Semestre

3.2.2 2º Semestre

O 2º Semestre, conforme está demonstrado na figura 15, foi composto por cinco fases:

- **Desenvolvimento da solução:**
    - **Desenvolvimento da aplicação de *backoffice*:** Nesta fase foi desenvolvida não só a aplicação web para registar ofertas de boleia e pedidos de transporte, como também a respetiva API referente à mesma.
    - **Desenvolvimento do serviço de simulação de dados:** Nesta fase foi efetuado um serviço de simulação de dados com o objetivo de posteriormente, os mesmos poderem ser testados através do módulo de *clustering* e otimização da rota.
    - **Desenvolvimento do serviço de *clustering* e otimização de rota:** Nesta fase foi desenvolvido o módulo de *clustering* e otimização da rota.
- **Escrita do relatório e apresentação para a defesa intermédia:** Nesta fase foi escrito o relatório para a defesa intermédia e foi efetuada a apresentação para a mesma, após ter sido validada pelos orientadores tanto do DEIS, como da própria empresa.
- **Elaboração do relatório final do projeto:** Paralelamente ao desenvolvimento da solução foi escrito o relatório final de estágio.

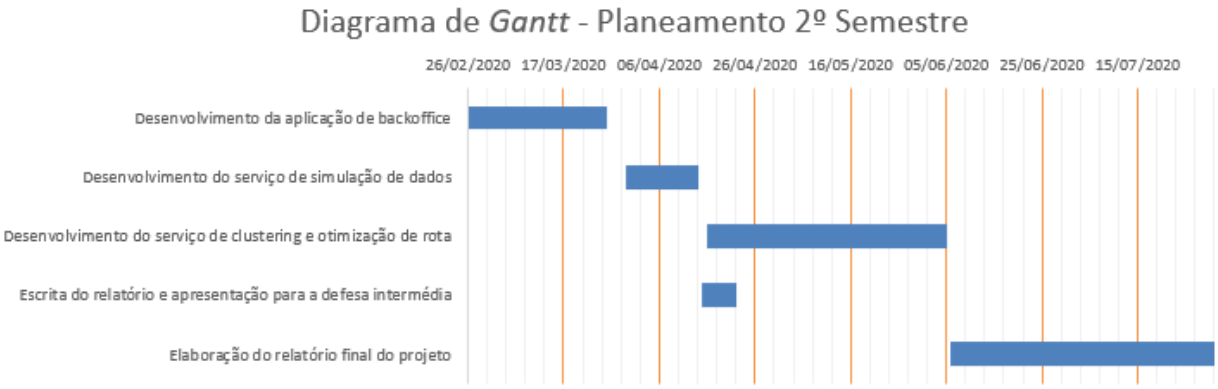


Figura 15 - Diagrama de *Gantt* - Planeamento 2º Semestre

Conforme a proposta de estágio que se encontra no anexo E, verificou-se que houve uma alteração face ao planeamento inicial do programa de trabalhos do estágio. Tanto no 1º semestre como no 2º semestre pode-se verificar que o plano foi praticamente seguido na íntegra, sendo que apenas não foram realizados testes, uma vez que não é um fator relevante para este projeto dado que houve um maior foco em *clustering* e o projeto é maioritariamente sobre análise de dados.

4 Tecnologias e Ferramentas Utilizadas

Neste capítulo, irão ser abordadas as tecnologias como também a arquitetura proposta durante o desenvolvimento do estágio. A empresa que acolheu o estágio desenvolve muita tecnologia em torno da linguagem *python* e as bibliotecas associadas, pelo que grande parte das escolhas tecnológicas são baseadas nesse contexto. Além disso, o *python* contém uma boa documentação, apresenta uma aprendizagem rápida e uma sintaxe simples e intuitiva, suporta diferentes plataformas, contém *web frameworks* que suportam o desenvolvimento da API e tal como já foi dito apresenta uma vasta quantidade de bibliotecas com o objetivo de ajudar o programador nas suas tarefas [68].

4.1 Tecnologias escolhidas para a aplicação de backoffice

4.1.1 Framework escolhida para backend

A escolha da framework para a parte de *backend* do desenvolvimento do módulo de *backoffice* é um fator importante para o desenvolvimento do projeto, uma vez que é necessário desenvolver esta aplicação de um modo rápido e eficiente [69]. Desta forma, optou-se por utilizar como *framework* de construção de *backend*, o **Django**.

O Django é muito utilizado nos dias de hoje devido à simplicidade que a sua estrutura apresenta, ao facto de ser *open-source*, rápido, seguro e escalável [70], apresentar uma excelente documentação e por apresentar uma curva de aprendizagem baixa, demonstrando assim fortes vantagens na sua utilização.

O Django foi uma das *frameworks* mais utilizadas como *backend* a nível *web* em 2019 [71] levando a um aumento da popularidade. Este aumento da popularidade leva a que a comunidade aumente e a que esteja constantemente a desenvolver ou melhorar funcionalidades, fazendo com que o Django não se torne obsoleto e entre em desuso, contribuindo para o sucesso do mesmo. O facto de também aplicações de elevada dimensão, nomeadamente o *Instagram*, *Pinterest*, *Disqus* utilizarem o Django mostra o poder desta framework.

O Django também possui integração com várias ferramentas, nomeadamente com uma extensão externa do PostgreSQL, o PostGIS que vai ser utilizado no desenvolvimento do projeto para fornecer suporte a dados geográficos.

Outras das vantagens de se escolher esta *framework* prende-se com o facto de se utilizar o mesmo na empresa, fazendo com que haja uma certa familiarização da ferramenta, além de que se convém

preservar a mesma *framework* para normalizar o esforço e, o mesmo poder ser transferido para a empresa para aspetos relevantes, tais como, continuação e manutenção. Também é importante mencionar que, uma vez que na empresa é maioritariamente utilizado o Django, pode ser mais fácil um dado projeto feito com recurso ao mesmo, integrar uma aplicação que use a mesma *framework*, reduzindo desta forma o tempo e esforço por parte do programador.

4.1.2 Base de dados

O projeto do presente estágio lidará com pedidos de transporte, onde um utilizador terá um formulário com vários campos, tais como: **local de partida, local de origem,** entre outros para requisitar um determinado pedido de transporte. Será fundamental que a base de dados escolhida possa armazenar facilmente as coordenadas GPS (base de dados espacial) e interligá-las com os dados já presentes noutras bases de dados relacionais do sistema.

O *DB-Engines Ranking* mostra uma classificação em que os dois Sistemas de Gestão de Bases de Dados (SGBDs) relacionais *open-source* mais populares são o **PostgreSQL** e o **MySQL** [72].

Segundo o DZone, os SGBDs mais utilizados em 2019 foram o **MySQL, PostgreSQL** e **MongoDB** [73]**.** Desta forma, foi efetuada uma comparação dos SGBDs mencionados anteriormente tendo em conta vários parâmetros, designadamente a **quantidade da comunidade/popularidade, tipo de licença, suporte para migrações, o tamanho máximo que a base de dados suporta** e **se suporta o Django nativamente.**

Tabela 1 - Comparação entre bases de dados

SGBD	**MySQL**	**MongoDB**	**PostgreSQL**
Comunidade (Quantidade de pessoas)?	1.266.283	413.183	491.071
Licenciamento	GNU	SSPL	Open-Source
Suporte Migrações?	Sim, mas caso a migração falhe, é preciso continuar manualmente	Sim, através do Djongo.	Sim.

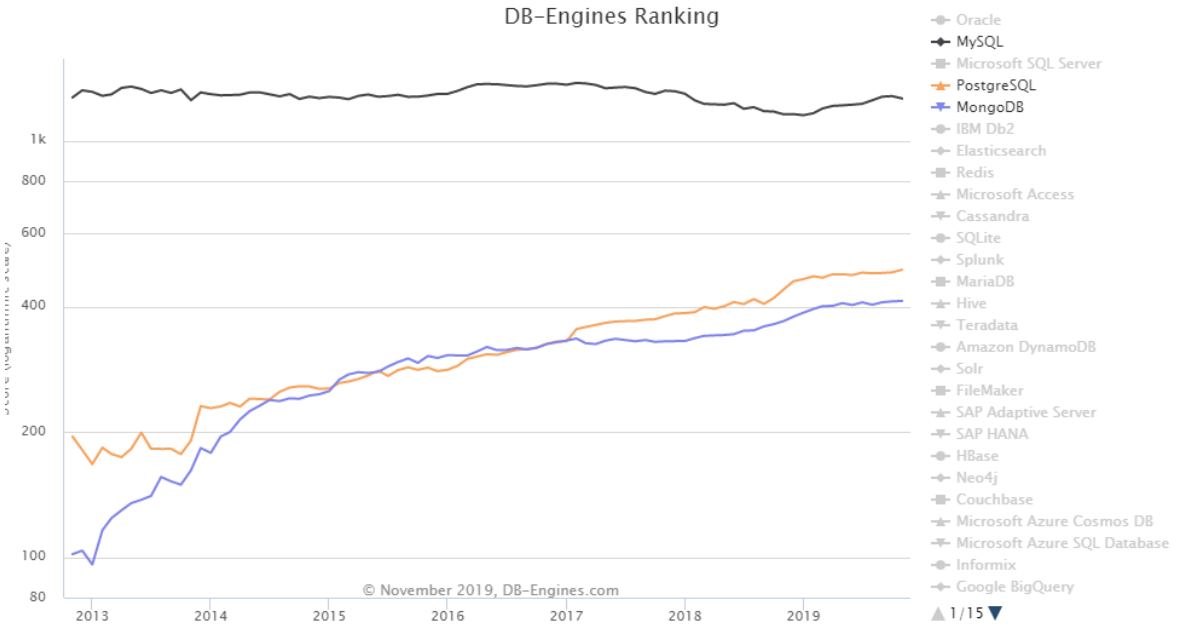
Máximo de tamanho suportado pela base de dados	Ilimitado	É escalável horizontalmente (adicionamos mais máquinas para aumentar a capacidade)	Ilimitado
Suporte ao Django nativo?	Nativo	Não nativo	Nativo
Suporte queries espaciais (nativo ou plugin)	Nativo	Nativo	Não nativo (com recurso à extensão PostGIS)
Apresenta bom desempenho em termos de execução de *queries* espaciais?	* (Comparação efetuada na tabela 2)	* (Comparação efetuada na tabela 2)	* (Comparação efetuada na tabela 2)

Relativamente à popularidade, é importante realçar que o método para calcular a mesma foi realizado através do método descrito pelo DB-Engines.com [74], com recurso aos seguintes parâmetros:

- **Número de vezes que o sistema é mencionado em websites** designadamente no **Google, Bing** e **Yandex;**
- **Interesse geral no sistema,** sendo que para esta medida é utilizada a frequência de pesquisas no **Google Trends;**
- **Frequência de discussões técnicas sobre o sistema,** onde é utilizado o número de perguntas relacionadas e o número de utilizadores interessados nos sites conhecidos de perguntas e respostas relacionadas à TI, como o **Stack Overflow** e **DBA Stack Exchange;**
- **Número de ofertas de trabalho em que o sistema é mencionado,** onde é utilizado o número de ofertas nos principais motores de busca de emprego, tais como o **Indeed** e **Simply Hired;**
- **Número de perfis em redes profissionais nas quais o sistema é mencionado,** onde são utilizadas as redes profissionais mais populares a nível internacional como o **Linkedin** e o **Upwork;**
- **Relevância nas redes sociais,** das quais são contados o número de *tweets* do **Twitter** em que um determinado sistema é mencionado.

Relativamente à popularidade, verificou-se que dos SGBDs mais utilizados em 2019, citados pelo DZone, o que possui uma maior comunidade é o **MySQL**, seguido do **PostgreSQL** e do

MongoDB. Segundo o DB-Engines Ranking, é possível verificar os dados que comprovam o que foi referido anteriormente, através do gráfico referente à figura 16 [75]:



Fonte: https://db-engines.com/en/ranking_trend

Figura 16 - Comparação da popularidade entre **MySQL, PostgreSQL e MongoDB**

Em termos de suporte a objetos geográficos, todos apresentam suporte, sendo que o PostgreSQL é o único que requer uma extensão externa para atribuir essa capacidade.

O Django suporta migrações para o **PostgreSQL** e o **MySQL**. O **MySQL** não é capaz de usar transações durante as migrações de *schema* [76], o que significa que se o processo de migração falhar, é necessário continuar o processo manualmente, levando consequentemente a perdas de tempo desnecessárias. Este critério é crítico para o trabalho em curso e permite excluir o **MySQL** da escolha. O **MongoDB** é *schemaless*, o que significa que nenhum *schema* é forçado pela base de dados. Desta forma, é possível integrar o **Django** com **MongoDB** através do **Djongo** [77]**,** do qual possui suporte para forçar e garantir um *schema*, e ajudar o mesmo ao longo do tempo, mas uma vez que é documental e é NoSQL (não é relacional), também é excluído automaticamente.

Quanto ao **PostgreSQL**, o PostGIS fornece suporte a dados geográficos, integra-se facilmente ao *Object-Relational Mapping* (ORM) do Django, sendo que o mesmo representa uma camada de abstração sobre as operações de leitura e escrita de dados numa base de dados relacional . Além disso, o PostGis permite consultas espaciais. Outro aspeto relevante a mencionar é que o **Django**, *framework* de Python que se vai utilizar no projeto, possui um maior suporte ao **PostgreSQL**, uma vez que a maioria dos campos específicos do **PostgreSQL** permite **integração com outras ferramentas**. Por este motivo, o **PostgreSQL** é compatível com uma

ampla variedade de linguagens e plataformas de programação, levando a que este seja um fator relevante, uma vez que poderá ser necessário migrar uma base de dados para outro sistema operacional ou integrá-lo a uma ferramenta específica.

De forma a comparar o desempenho em termos de execução de algumas *queries* espaciais, foi criada uma tabela com um campo geométrico do tipo *"Point"* que era composto por um par de coordenadas *x* e *y*, tendo posteriormente sido desenvolvido um *script* para efeitos de análise de comparação. De realçar que esta comparação foi efetuada numa máquina de desenvolvimento com um processador i7 com 32 GB de RAM e sobre um disco SSD com um sistema operativo Windows com múltiplos processos a decorrer. Para minimizar o efeito da flutuação de carga derivada dos múltiplos processos a decorrer, as operações foram executadas 10 vezes cada e feita a média dos tempos. Sendo assim, na tabela 2 podem ser visualizados os seguintes resultados.

Tabela 2 - Comparação de motores de base de dados em termos de média de execução de *queries*

Tipo de *query*/Motor de base de dados	MySQL	MongoDB	PostgreSQL
Inserir 1000 linhas	11,603732347488403	0,5776355266571045	0,3443329334259033
	11,11280083656311	0,5909864902496338	0,32024240493774414
	11,493143558502197	0,4455440044403076	0,32842516899108887
	10,71086049079895	0,5755953788757324	0,3155393600463867
	10,993676900863647	0,5286645889282227	0,2918984889984131
	12,187855005264282	0,5514774322509766	0,39344120025634766
	10,921396493911743	0,5826511383056641	0,32221508026123047
	10,925800800323486	0,5072979927062988	0,3830711841583252
	10,797261714935323	0,4938664436340332	0,3510093688964844
	10,861586332321167	0,5700814723968506	0,2948164939880371
Média dos 10 tempos de	11,16081 ≅ 11,1608	0,54238 ≅ 0,5424	0,334499 ≅ 0,3345

inserir 1000 linhas			
Inserir 1 linha	0,010321617126464844	0,01697063446044922	0,0020067691802978516
	0,010972738265991211	0,003983497619628906	0,0029904842376708984
	0,010929107666015625	0,02310800552368164	0,0018999576568603516
	0,009293079376220703	0,02420496940612793	0,0029463768005371094
	0,008976936340332031	0,012863874435424805	0,002953052520751953
	0,009004592895507812	0,006297588348388672	0,002985715866088867
	0,007702350616455078	0,014014720916748047	0,0019791126251220703
	0,010069608688354492	0,003988504409790039	0,0021402835845947266
	0,009325742721557617	0,013881444931030273	0,002030611038208008
	0,009011507034301758	0,02448439598083496	0,0029888153076171875
Média dos 10 tempos de inserir 1 linha	0,009561 ≅ 0,0096	0,01438 ≅ 0,0144	0,002492 ≅ 0,0025
Atualizar 1 linha	0,0	0,005620241165161133	0,005007505416870117
	0,0	0,006784200668334961	0,0039865970611572266
	0,0	0,016693115234375	0,00501251220703125
	0,0009872913360595703	0,005986690521240234	0,003552675247192383
	0,0	0,022989511489868164	0,003978729248046875
	0,00101470947265625	0,00742340087890625	0,006988048553466797
	0,0	0,008010387420654297	0,0030341148376464844

	0,0	0,025905847549438477	0,00598454475402832
	0,0	0,02427220344543457	0,004971742630004883
	0,0	0,02285623550415039	0,00392913818359375
Média dos 10 tempos de atualizar 1 linha	0,0002	0,014654 ≅ 0,0147	0,004645 ≅ 0,0046
Ler todas as linhas	0,0009970664978027344	0,03262066841125488	0,002008199691772461
	0,0009970664978027344	0,03706192970275879	0,0020232200622558594
	0,0020074844360351562	0,05188488960266113	0,0020062923431396484
	0,000990152359008789	0,058496952056884766	0,0031228065490722656
	0,0009930133819580078	0,04706144332885742	0,001993894577026367
	0,0001575946807861328	0,039505720138549805	0,0029964447021484375
	0,002034902572631836	0,041881561279296875	0,0025472640991210938
	0,001996755599975586	0,05403780937194824	0,0009965896606445312
	0,0009961128234863281	0,05686235427856445	0,0020339488983154297
	0,0009953975677490234	0,03376579284667969	0,0019931793212890625
Média dos 10 tempos de ler todas as linhas	0,001217 ≅ 0,0012	0,045318 ≅ 0,0453	0,002172 ≅ 0,0022
Eliminar todas as linhas	0,019915342330932617	0,019954442977905273	0,0020329952239990234
	0,019597768783569336	0,020736217498779297	0,002991914749145508
	0,021961212158203125	0,028394222259521484	0,002027750015258789

	0,02657461166381836	0,031995534896850586	0,0020046234130859375
	0,019939184188842773	0,02097153663635254	0,0019958019256591797
	0,020771265029907227	0,03164052963256836	0,002994060516357422
	0,019733190536499023	0,029917240142822266	0,001997709274291992
	0,02500748634338379	0,026018619537353516	0,0019948482513427734
	0,018921852111816406	0,03987693786621094	0,0038597583770751953
	0,022396087646484375	0,03791189193725586	0,00199127197265625
Média dos 10 tempos de eliminar todas as linhas	0,021482 ≅ 0,0215	0,028742 ≅ 0,0287	0,002433 ≅ 0,0024

Através da tabela 2, conseguimos visualizar que, para a *query* de inserção de 1000 linhas, o motor de base de dados que apresenta o melhor desempenho é o PostgreSQL, seguido do MongoDB e do MySQL.

Já para as *queries* de inserção de 1 linha e de eliminação de todas as linhas, o motor de base de dados que apresenta o melhor desempenho é o PostgreSQL, seguido do MySQL e MongoDB. Por último, podemos verificar que, para as *queries* de atualização de 1 linha e de leitura de todas as linhas, o MySQL é aquele que apresenta o melhor tempo médio de execução seguido do PostgreSQL e do MongoDB.

Em suma, a tecnologia que foi escolhida como sistema de gestão de base de dados para este projeto foi o **PostgreSQL** pelo facto de conter uma extensão espacial, o PostGIS que ajuda a fornecer suporte a dados geográficos, de se integrar facilmente ao ORM do Django e permitir consultas espaciais. Além disso, o PostgreSQL é compatível com uma grande quantidade de linguagens de programação, sendo um fator importante, uma vez que poderá ser necessário efetuar a migração de uma base de dados para outro sistema operacional ou até integrá-lo noutra ferramenta. Além dos motivos mencionados anteriormente, o PostgreSQL foi o sistema de gestão de base de dados que apresentou melhores resultados em termos de desempenho.

4.1.3 Nominatim

O Nominatim [78] é um serviço *open-source* que utiliza dados do OpenStreetMap e converte coordenadas geográficas para uma dada morada e também permite converter uma morada em coordenadas. Este encontra-se alojado na Ubiwhere e foi utilizado pelo estagiário para extrair as coordenadas das localizações de partida e chegada, tanto das ofertas de boleia introduzidas pelos condutores, como dos pedidos de transporte introduzidos pelos passageiros na aplicação *web*.

4.1.4 Django REST Framework

O Django REST Framework é um conjunto de ferramentas poderosas e flexíveis utilizado para a construção de Web API's. A sua enorme utilização deve-se nomeadamente ao facto de conter políticas de autenticação, possuir serialização, suportar ligação de base de dados com ORM e sem ORM, uma extensiva documentação, um grande suporte por parte da comunidade e ser bastante utilizado por empresas de renome internacional, tais como, a *Mozilla*, o *Red Hat*, *Heroku* e *Eventbrite* [79]. Foi utilizado pelo estagiário para construir a API para a aplicação web desenvolvida.

4.2 Tecnologias usadas para o serviço de simulação de dados

4.2.1.1 Python

O Python é uma linguagem de programação que surgiu em 1991, no centro de Matemática e Computação em Amesterdão, na Holanda. Este é bastante usado em várias áreas, tais como, matemática, estatística e ciência de dados [80]. Foi optado por utilizar esta linguagem derivado à sua simplicidade, baixa curva de aprendizagem e, enorme popularidade que apresenta. Além disso possui uma vasta quantidade de bibliotecas que facilitam a vida do programador.

4.2.1.2 OpenCage Geocoder

O OpenCage Geocoder [81] é uma API que converte coordenadas geográficas para uma dada morada e também permite converter uma morada em coordenadas. De forma a ser possível utilizar este serviço é necessário uma API *key* válida que deve ser passada como parâmetro em cada chamada para a API que é efetuada. Foi optado por utilizar o OpenCage Geocoder como alternativa ao Nominatim (sugerido pela empresa), uma vez que na altura do desenvolvimento do projeto foi encontrada uma limitação relativamente a esta última. Esta limitação devia-se ao facto do Nominatim bloquear várias vezes sempre que se efetuava um pedido à sua API. Desta forma, foi procurada uma alternativa que permitisse através de uma API *key* ter acesso a 2500 pedidos por dia e assim, resolver o problema.

Foi utilizado pelo estagiário para extrair as coordenadas das localizações de partida e chegada, tanto das ofertas de boleia dos condutores como dos pedidos de transportes dos passageiros, de forma automática no serviço de simulação de dados realizado.

4.3 Tecnologias usadas para o módulo de clustering e otimização da rota

4.3.1.1 Python

Além de uma linguagem de programação geral como mencionada anteriormente, possui uma vasta quantidade que bibliotecas de análises de dados com algoritmos pré implementados, tais como, o *scikit-learn* para *Machine Learning*, o *Numpy* e o *Pandas* para análise de dados, sendo que as mesmas vão ser utilizadas para o desenvolvimento do módulo de *Clustering* e otimização de rota.

4.3.1.2 PostGIS

O PostGIS é uma extensão espacial que possibilita armazenar e manipular dados espaciais em base de dados PostgreSQL. Este é grátis, *open-source* e possui uma vasta quantidade de funcionalidades. Além disso é bastante utilizado e possibilita a importação e exportação de dados. Foi utilizado pelo estagiário para construir *queries* espaciais nomeadamente para fazer a respetiva correspondência entre as ofertas de boleia de condutores e pedidos de transporte de passageiros agrupados pelo clustering [82].

4.3.1.3 Pandas

O Pandas foi construído com recurso à linguagem de programação Python e é uma ferramenta eficiente, rápida, flexível e de fácil utilização [83]. Esta é uma biblioteca grátis e *open-source* usada para análise, manipulação e visualização de dados [84].

4.3.1.4 Scikit-learn

O Scikit-learn é uma biblioteca grátis muito utilizada para *machine learning* em Python. Este possui vários algoritmos de classificação, regressão e agrupamento incluindo por exemplo o K-Means e o DBSCAN e também possui integração com várias bibliotecas de Python, tais como o *Numpy* e o *SciPy* [85] [86]. Foi usado pelo estagiário para utilizar os algoritmos de *clustering*.

4.3.1.5 RabbitMQ

O RabbitMQ é um servidor de mensagens *open-source* desenvolvido em Erlang, que foi implementado com o objetivo de suportar mensagens através de um protocolo denominado *Advanced Message Queueing Protocol* (AMPQ). Este tem a vantagem de lidar com várias mensagens de uma forma rápida, ser compatível com várias linguagens de programação e

apresentar uma interface administrativa na qual se pode ver as mensagens que são publicas e consumidas através das *queues*. Em suma, este garante assincronismo entre as aplicações, diminui o acoplamento entre as mesmas, distribui alertas e controla filas de trabalhos em background [87]. Foi utilizado pelo estagiário para ler e publicar mensagens entre várias *queues*.

4.3.1.6 Celery

O Celery é uma ferramenta usada para executar várias tarefas em segundo plano. Existem dois conceitos importantes no Celery que são o *daemon* e o *Celerybeat*. O *daemon* permite a execução de tarefas, sendo que o *CeleryBeat* permite o agendamento das mesmas [88]. O Celery é usado como um *trigger* da tarefa de *match* entre pedidos de boleia e ofertas de transporte agrupados pelo *clustering*. Neste caso foi utilizado pelo estagiário para de 1 em 1 minuto repetir as várias tarefas do módulo de *clustering* e otimização de rota, tal como será discutido na secção 6.4.1. O arranque do processo é efetuado através da publicação de uma mensagem de uma *queue* do RabbitMQ.

4.3.1.7 Psycopg

O Psycopg é um adaptador de base de dados PostgreSQL. A sua utilização permite através da linguagem Python reconhecer a base de dados PostgreSQL, permitindo assim o acesso às suas funcionalidades [89]. Foi utilizado pelo estagiário nomeadamente para fazer a correspondência entre as várias ofertas de boleia disponíveis e os pedidos de transporte agrupados no *clustering* através de *queries* espaciais, tal como será discutido na secção 6.4.3.

4.3.1.8 Open Source Routing Machine (OSRM)

O OSRM é um algoritmo eficiente de roteamento que serve para determinar trajetos em redes rodoviárias. Além disso, é *open-source* e qualquer utilizador pode hospedá-lo na sua máquina sem ter de pagar uma licença [90].

Este serviço disponibiliza um conjunto de API's totalmente grátis para o utilizador. As API's disponibilizadas são as seguintes [91]:

- *Nearest Service*: Através da indicação de uma coordenada, retorna as *n* partidas mais próximas;
- *Route Service*: Encontra a rota mais rápida entre as coordenadas especificadas;
- *Table Service*: Calcula a duração da rota mais rápida entre todos os pares de coordenadas fornecidos;
- *Match Service*: Dado um conjunto de dados GPS, o serviço de correspondência tentará igualar esses pontos à rede rodoviária de maneira mais árdua;

- *Trip Service*: Utiliza um algoritmo de força bruta para menos de 10 pontos e que por defeito se baseia no problema do caixeiro viajante, fazendo com que garanta uma aproximação da melhor rota;
- *Tile Service*: Este serviço gera *Mapbox Vector Tiles* que podem ser visualizados com um visualizador de mapas deslizantes com capacidade para vetores.

Foi optado por utilizar o *endpoint Trip Service*, sendo que as razões desta escolha se encontram devidamente justificadas no subcapítulo 7.3.

4.4 Ferramentas

4.4.1 GitLab

O GitLab [92] é uma plataforma de hospedagem de código-fonte com controlo de versões usando o *Git*. Esta plataforma, permite que os programadores e utilizadores que estejam registados contribuam em projetos privados e/ou *Open Source* de qualquer lugar do mundo.

Esta ferramenta é um serviço de hospedagem de projetos, possibilitando a criação de repositório e criação de uma *branch* que funciona, de certa forma, como uma nova pasta de trabalho (*online*) e que permite ao programador armazenar código ou fazer *download* do mesmo para o seu computador local. Permite também visualizar todos os *commits* realizados e comparar as diferenças entre o mesmo ficheiro ao longo do tempo. Além disso o GitLab também contém ferramentas de integração e entrega contínua (CI/CD), *boards* para gerir a execução do projeto, levando a que haja por isso uma maior monitorização do mesmo.

Foi utilizado pelo estagiário não só para guardar o código tanto da aplicação *web* desenvolvida, como também do simulador de dados e do serviço de *clustering* e otimização de rota, ajudando também simultaneamente a gerir todo o projeto.

4.4.2 Jupyter Notebook

O Jupyter Notebook [93] é uma aplicação web que permite o desenvolvimento, visualização e resultados dos dados. Além disso, permite a partilha e a publicação *online* de um ou mais ficheiros num repositório *git*. Foi utilizado na determinação dos parâmetros para a técnica de *clustering* DBSCAN.

4.4.3 PyCharm

O PyCharm [94] é um ambiente de desenvolvimento direcionado para a linguagem Python e foi desenvolvido pela empresa JetBrains. Foi utilizado pelo estagiário para desenvolver a aplicação

de *backoffice*, o serviço de simulação de dados dos pedidos de transporte e ofertas de boleia e o módulo de *clustering* e otimização de rota.

4.4.4 Docker

O Docker é uma plataforma *open-source* que foi desenvolvida na linguagem GO e cujo objetivo passa por criar ambientes isolados para aplicações e serviços [95]. Este foi utilizado pelo estagiário nomeadamente para executar vários serviços, como por exemplo o RabbitMQ.

5 Requisitos do sistema

Neste capítulo irão ser descritos os requisitos tanto da aplicação de *backoffice* como do serviço de módulo de *clustering* e otimização de rota e do serviço de simulação de pedidos de transporte e ofertas de boleias.

5.1 Requisitos da aplicação de backoffice

5.1.1 Requisitos Funcionais

No que toca aos requisitos funcionais, estes foram levantados ao estilo de *user stories*. Estas ajudam a decompor um grande problema em partes mais pequenas, agindo como um guia no desenvolvimento de um projeto. Além disso, é uma forma de ajudar o programador a entregar exatamente aquilo que um dado cliente especificou [96].

Sendo assim, foram definidas as seguintes *user stories*:

- Como passageiro posso registar um pedido de transporte. Para isso, terei de fornecer a seguinte informação:
    - Localização de partida;
    - Localização de chegada;
    - Data e hora do pedido.
- Como passageiro, posso visualizar a lista dos pedidos de transporte que efetuei;
- Como passageiro, posso tornar-me um condutor se assim pretender e ter assim acesso às funcionalidades que dizem também respeito aos condutores. Para tal, terei de fornecer a seguinte informação:
    - Número de identificação fiscal;
    - Carta de condução.
- Como condutor, posso registar veículos. Para tal, terei de fornecer a seguinte informação:
    - Marca do veículo;
    - Modelo do veículo;

  - Matrícula do veículo;
  - Capacidade;
  - Possui carga ou não.
- Como condutor, posso registar ofertas de boleia. Para tal, terei de fornecer a seguinte informação:
  - Localização de partida;
  - Localização de chegada;
  - Data e hora de partida da oferta de boleia;
  - Data e hora de retorno da oferta de boleia.
- Como condutor, posso visualizar a minha lista de veículos;
- Como condutor, posso visualizar a minha lista de ofertas de boleia;
- Como condutor, devo conseguir visualizar o estado de viagem das minhas ofertas de boleia;
- Como passageiro, devo conseguir visualizar o estado de viagem dos meus pedidos de transporte;
- Como passageiro/condutor, devo conseguir sair da aplicação voltando ao ecrã inicial da mesma;
- Como passageiro, devo ser notificado nas seguintes ocasiões:
  - Quando o registo de pedido de transporte é submetido com sucesso;
  - Quando o registo de pedido de transporte dá algum erro;
  - Caso pretenda ser um condutor, mas já tenha aderido é fornecido uma mensagem de erro.
- Como condutor, devo ser notificado nas seguintes ocasiões:
  - Quando o registo de oferta de boleia é submetido com sucesso;
  - Quando o registo de oferta de boleia dá algum erro.
- Como passageiro/condutor, posso efetuar login desde que me encontre registado na aplicação. Para tal, tenho de fornecer a seguinte informação:
  - *Username*
  - *Password*

- Como passageiro/condutor, posso recuperar a *password* caso me tenha esquecido da mesma. Para tal, tenho de fornecer a seguinte informação: *Email*;

  Posteriormente, o passageiro/condutor recebe um link via *email*. Quando clicar no mesmo é redirecionado para uma página *web*, sendo que terá de indicar a nova *password* duas vezes.

- Como passageiro/condutor, devo ser capaz de visualizar o meu perfil;

- Como passageiro/condutor, devo ser capaz de editar o meu perfil. Para isso tenho de fornecer a seguinte informação: *Email* ou primeiro nome ou último nome ou até todos.

5.1.2 Requisitos não funcionais

Para além dos requisitos funcionais, também foram identificados vários requisitos não funcionais sendo que os mesmos estão relacionados ao modo como as funcionalidades da aplicação são entregues ao utilizador. Estes, encontram-se apresentados na tabela 3.

Tabela 3 - Requisitos não funcionais da aplicação de *backoffice*

Referência	**Requisito**
RNF-1	Usabilidade
RNF-2	Acessibilidade
RNF-3	Segurança
RNF-4	Interoperabilidade
RNF-5	Desempenho

RNF-1: Usabilidade

A aplicação deve apresentar uma *interface* amigável, intuitiva e de fácil utilização de forma a garantir uma boa comunicação entre o utilizador e o sistema mesmo para utilizadores com pouca experiência computacional.

RNF-2: Acessibilidade

Os pontos de acesso aos conteúdos da aplicação devem ser encontrados de forma simples e intuitiva pelo utilizador, de modo a facilitar a sua acessibilidade.

RNF-3: Segurança

O sistema deve garantir que os dados do utilizador são salvaguardados, garantindo privacidade e confidencialidade.

RNF-4: Interoperabilidade

A aplicação deve interagir com a API do Nominatim, via web API.

RNF-5: Desempenho

O sistema deve apresentar um bom desempenho ao utilizador na utilização da aplicação, uma vez que uma má *performance* leva a uma redução de produtividade dos utilizadores, podendo levar a que a aplicação fique obsoleta. Sendo assim, o sistema deverá responder a qualquer pedido em menos de 1 segundo.

5.1.3 Casos de uso

Os casos de uso têm como função descrever a interação entre o utilizador e a aplicação.

A identificação dos requisitos da aplicação permitiu elaborar um diagrama de casos de uso que se encontra na figura 17. Este diagrama de casos de uso ilustra os diferentes cenários de utilização, que se encontram descritos no **anexo A**, resultantes da interação do utilizador com a aplicação.

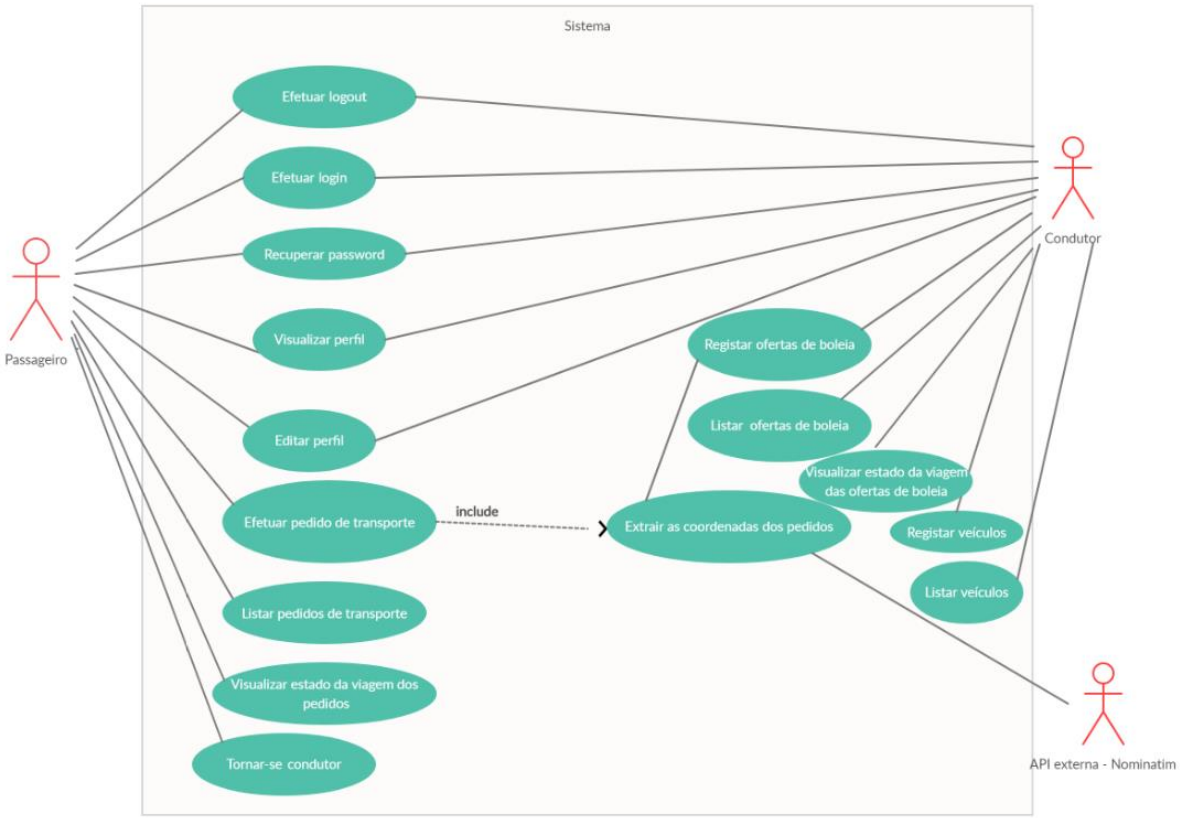


Figura 17 - Diagrama de casos de uso da aplicação de *backoffice*

5.2 Requisitos do módulo de clustering e otimização da rota

5.2.1 Requisitos gerais

O processo de associação de pedidos de boleia dos passageiros a ofertas de boleia por parte dos condutores não é realizado no instante em que é registado o pedido ou oferta de boleia. O processo de atribuição é assíncrono e executado periodicamente. Esta opção tem em consideração que todos os pedidos de boleia não são em tempo real e quando o pedido de boleia é efetuado pode não haver ofertas que façam o *match*. O inverso também se aplica – quando uma oferta de boleia é registada pode ainda haver ausência de pedidos de boleia que sejam associados.

Neste sentido, um passageiro quando pede boleia não deve esperar que haja logo imediatamente a associação a uma oferta de boleia. Deve apenas ver que o seu pedido foi registado com sucesso. O mesmo aplica-se ao condutor que oferece boleia, que vê a sua oferta ser registada sem associação imediata de passageiros. Quando é feita uma associação, os intervenientes podem ser notificados.

Porque a associação não é feita de imediato, a mesma pode ter regras otimizadas para melhoria do volume de passageiros, bem como otimização das rotas. Para tal, o processo de associação deve ser executado periodicamente sobre o conjunto de pedidos ainda sem viagem marcada. O processo é executado em *background* sem visualização de UI. Os resultados são transmitidos aos condutores e passageiros apenas quando é possível obter um *match*.

O processo de associação deve ter em conta o período temporal da viagem, sendo a oferta da boleia associada de modo a cumprir as necessidades de horários do passageiro. O mesmo se aplica em termos de localização geográfica, sendo que a boleia deve ser a mais próxima possível dos pedidos que foram agrupados pelo *clustering*.

Nesta iteração do sistema deve assumir que todos os veículos associados são de 5 lugares, sendo que um deles é ocupado pelo condutor, permitindo associar no máximo 4 passageiros na viagem. A definição de viagem deve ser no momento em que o *match* é detetado, podendo ser criadas as viagens com no mínimo dois passageiros. Uma vez criada uma viagem, a oferta de boleia deve ser excluída das próximas tentativas de *match*.

As rotas que os condutores assumem para as viagens devem ser otimizadas para minimizar o custo do percurso para o condutor.

5.2.2 Regras de associação de boleias a condutores

Na associação de viagens:

- Os pedidos devem ser agrupados por data e hora do dia;

- Cada grupo da mesma data e hora deve ser agrupado geograficamente e dividido em parcelas de no mínimo dois pedidos e no máximo quatro pedidos;
- Para cada grupo resultante do *clustering* são procuradas as ofertas de boleia que estão dentro da mesma *bounding box*;
- É escolhida a oferta de boleia mais próxima do centro do *cluster* de cada grupo, sendo feita a associação entre os pedidos e a oferta;
- Posteriormente, com a respetiva oferta de boleia e pedidos agrupados é feita a otimização da rota, conforme irá ser abordado na secção 6.4.4.

A iteração seguinte do algoritmo incluirá apenas os pedidos e ofertas sem viagem atribuída, bem como qualquer novo pedido ou oferta que seja registada no sistema no intervalo entre cálculos, sendo iniciado uma nova iteração do algoritmo de *clustering* e otimização de rota a cada minuto.

5.3 Requisitos do serviço de simulação de pedidos de transporte e ofertas de boleia

No que toca ao serviço de simulação de pedidos de transporte e ofertas de boleia foi necessário gerar dados artificiais de forma a serem testados no módulo de *clustering* e otimização da rota. Esta necessidade deveu-se ao facto de não haver grande disponibilidade de *datasets* abertos para o tema de *carpooling*. De notar que todos os pedidos de transporte e ofertas de boleia foram restritos para a área geográfica da região de Coimbra.

6 Arquitetura

De modo a cumprir com os requisitos do sistema descritos no capítulo anterior e considerando que se pretende uma aplicação *web*, foi desenhada uma arquitetura modular, cuja representação se encontra apresentada na figura 18.

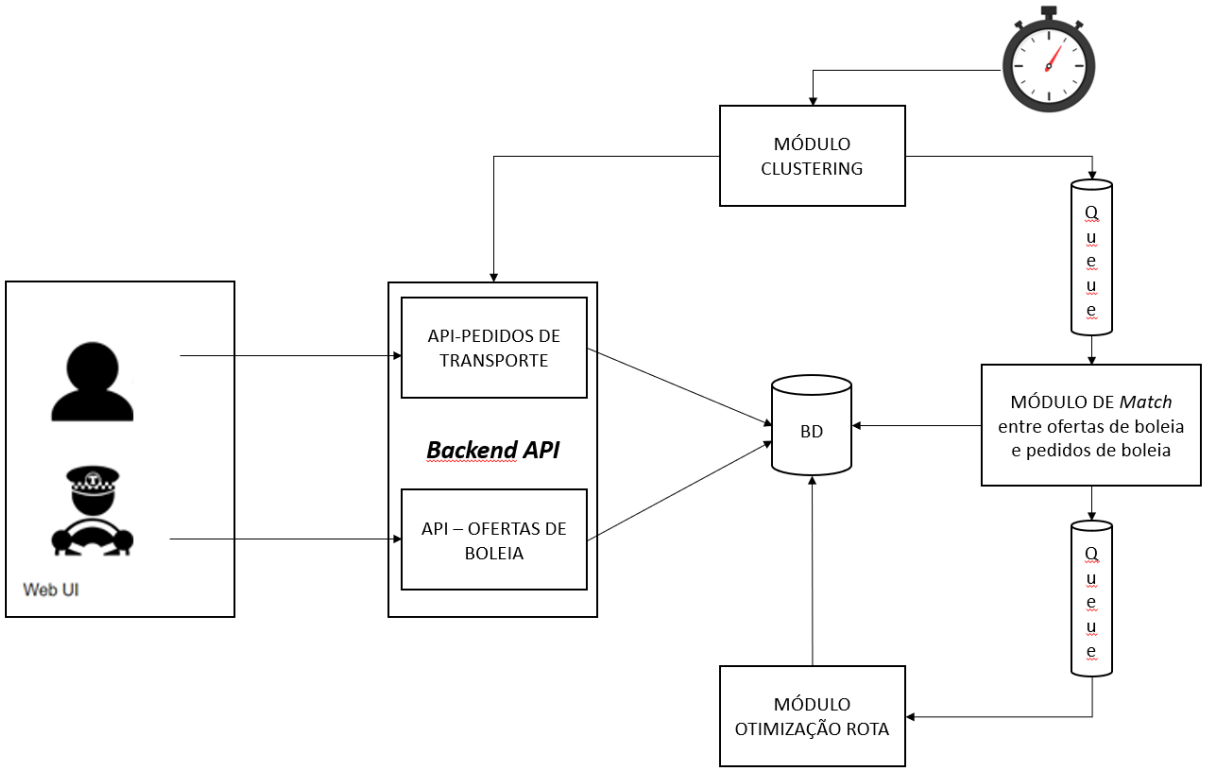


Figura 18 - Arquitetura do sistema do módulo de *clustering e otimização da rota*

A aplicação tem uma interface *web* (Web UI à esquerda na Figura 18) para os utilizadores finais (condutor, passageiro e administrador). A Web UI comunica por HTTP com a API de *backend* através dos *endpoints* dedicados às funções de registo de pedidos de transporte por parte dos passageiros e de registo de ofertas de boleia por parte dos condutores. A API interage com uma base de dados central para o armazenamento de dados. O sistema é também composto por um conjunto de módulos responsáveis por efetuar o *clustering* dos pedidos, o *match* entre os pedidos e ofertas e a otimização de rota das viagens geradas, tal como será discutido no capítulo 6.4. Estes módulos compõem a *pipeline* de processamento de pedidos para geração de viagens e funcionam como processos independentes de execução assíncrona e interligados por *queues*. A *pipeline* de processamento é iniciada periodicamente com recurso a um componente de calendarização.

A decomposição do sistema em módulos traz várias vantagens, nomeadamente, a possibilidade de desenvolvimento independente de cada bloco, bem como a escolha de tecnologias distintas para cada bloco. No entanto, a independência dos módulos requer atenção aos métodos de comunicação entre eles de modo a permitir a integração adequada. Como protocolos de comunicação, foram escolhidos os pedidos por HTTP para todas as comunicações com a API e foi utilizado o protocolo *Advanced Message Queuing Protocol* (AMQP) para a publicação de mensagens em *queues* entre os módulos do *pipeline* de processamento. Seguindo o padrão de microsserviços, um sistema modular é também mais resiliente a falhas, especialmente se existirem processos independentes, sendo que um módulo pode falhar e os restantes do sistema mantêm-se em funcionamento.

6.1 Módulo web de UI

O módulo *web* de UI é responsável por apresentar uma interface de interação com os utilizadores do sistema, aqui definidos como sendo o passageiro que efetua pedidos de transporte e os condutores que publicam ofertas de boleia. Trata-se de uma interface *web* acedida pelo *browser* no computador ou em dispositivos móveis.

De modo a suportar este modo de interação, a aplicação é construída em Django, executando num processo alojado num servidor central disponível na *web* pública.

As interações dos utilizadores são transformadas em pedidos HTTP e enviados para as APIs de modo a obter dados para preencher as páginas que compõem a interface ou registar pedidos e ofertas de boleia.

Este módulo integra um mecanismo de autenticação providenciado pelo próprio Django, que recorre a *tokens* do tipo *Json Web Token* (JWT) para validar as comunicações entre o cliente e o servidor. Os dados dos utilizadores são armazenados na base de dados central. Outra integração incluída no módulo de UI é a integração com a API privada do Nominatim alojada pela Ubiwhere e que é utilizada para *geocoding*.

Apesar do módulo *web* poder ser desenvolvido e executado num processo independente, a escolha feita foi de integrar os componentes de UI na mesma base de código e processo de execução que a API de *backend* do sistema. Ambos os componentes têm por base o Django e, como tal, a integração torna-se mais simples e económica do que manter dois processos independentes ou tecnologias distintas.

6.2 Módulo de API de backend

A API de *backend* expõe um conjunto de *endpoints* REST, acessíveis por HTTP com as ações possíveis, relativa a pedidos de boleia e ofertas de boleia. Cada *endpoint* exige autenticação do utilizador e quando invocado executa um processo de negócio que tipicamente envolve obtenção

e gravação de dados na base de dados. As ações de obtenção dos dados são através de pedidos com o verbo *get* enquanto as invocações que envolvem a alteração de estado do sistema recorrem a pedidos *post*. Os dados transmitidos entre cliente e servidor são transportados no corpo do pedido ou resposta serializados no formato JSON.

Como mencionado na subsecção anterior, o processo da API corre em conjunto e partilha o código do módulo *web*.

6.3 Base de dados

O sistema recorre a uma base de dados relacional PostgreSQL, com auxílio à extensão espacial do PostGIS que permite armazenamento e pesquisas de dados espaciais, conforme se pode visualizar na figura 19. Todos os dados armazenados estão organizados em tabelas associadas a um único *schema* público. De realçar que foram criadas várias classes que representam uma determinada entidade, que refletem as propriedades da mesma. Cada propriedade de uma determinada entidade pode ser mapeada para uma coluna da tabela que a descreve. Para a construção da base de dados, não há interação direta com as tabelas. A construção efetiva da base de dados é efetuada por um processo denominado de "Migração". O processo de "Migração", envolve a execução de dois passos numa aplicação Django.

- *"python manage.py makemigrations"*
- *"python manage.py migrate"*

O primeiro comando é responsável por criar novas migrações, sempre que houver alterações nos modelos de dados, sendo que o segundo comando é encarregue de aplicar essas alterações à base de dados, ou seja, executa todos os ficheiros de migração que ainda não foram executados na mesma.

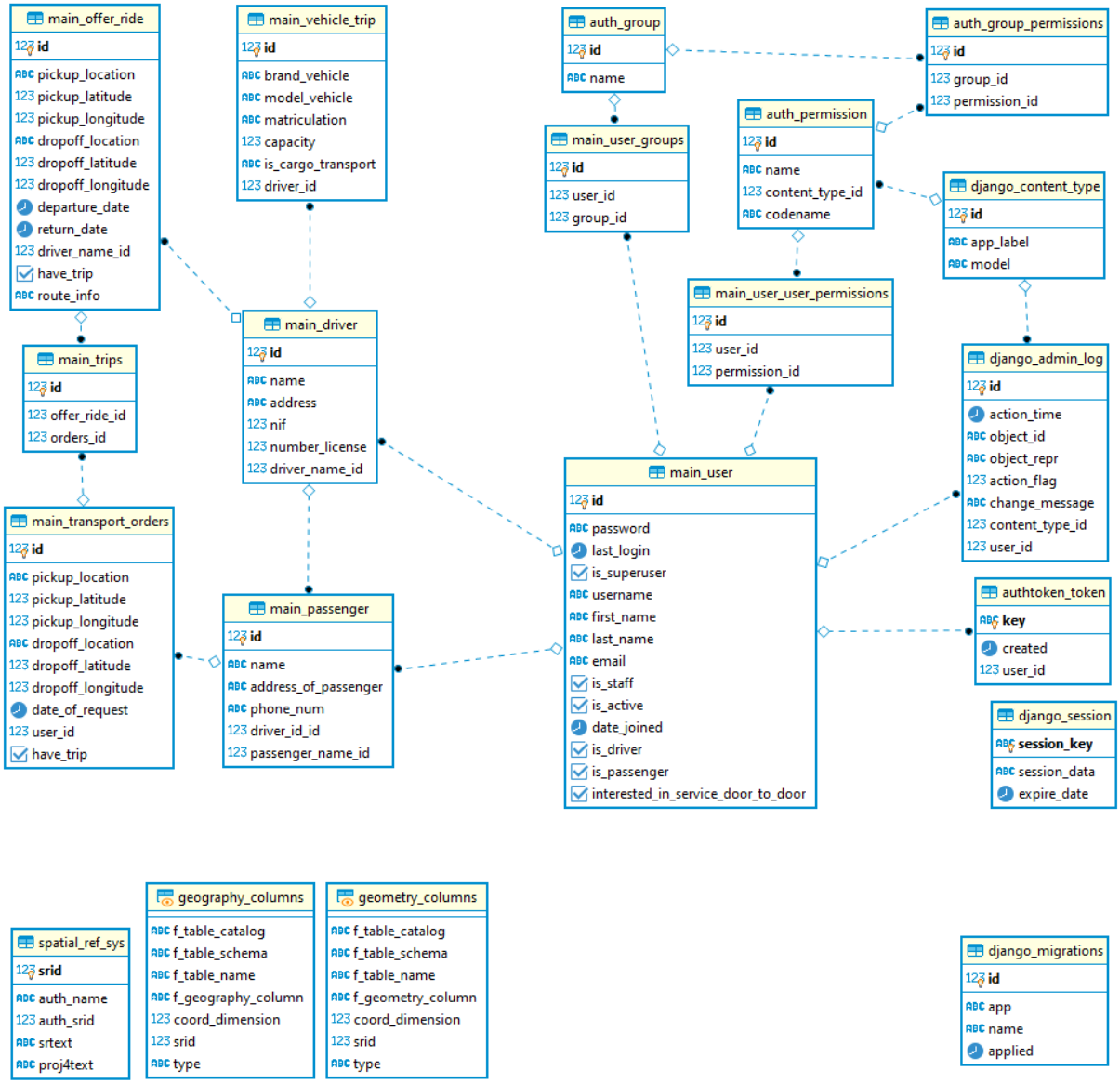


Figura 19 - Diagrama de base de dados

Com o diagrama de base de dados, verifica-se que existem quatro grupos de tabelas que servem o armazenamento de estado da aplicação. A principal (canto superior esquerdo) serve o domínio da gestão das viagens, é auxiliada por um grupo de tabelas de gestão de utilizadores (canto superior direito), um grupo de tabelas associadas à informação geográfica do PostGIS (canto inferior esquerdo) e uma tabela da gestão do estado da estrutura da base de dados (canto inferior direito).

A estrutura de tabelas que descreve o domínio de pedidos de transporte está focada em duas entidades: *main_transport_orders* e *main_offer_rides* que representam pedidos de boleia e ofertas de transporte respetivamente.

Cada registo em *main_transport_orders* refere um pedido de transporte que pertence a um único passageiro identificado pela coluna *user_id*. A coluna *user_id* é uma chave estrangeira para a tabela *main_passenger* que descreve *users* que escolheram ser passageiros no sistema. A tabela *main_passenger* inclui ainda colunas para identificar dados pessoais dos passageiros nomeadamente o nome, morada, número de telemóvel, bem como uma chave estrangeira *passenger_name_id* que liga a tabela de utilizadores *main_user*.

A tabela *main_transport_orders* inclui informação que descreve o pedido, nomeadamente a morada e coordenadas de partida (*pickup_location*, *pickup_latitude*, *pickup_longitude*), informação de destino (*dropoff_location*, *dropoff_latitude*, *dropoff_longitude*) e a data e hora em que o utilizador pretende realizar a viagem (*date_of_request*). Os campos mencionados são preenchidos no momento de inserção da linha na tabela, havendo apenas a coluna *have_trip*, que é atualizada posteriormente, indicando que o pedido tem ou não uma viagem atribuída, auxiliando as pesquisas.

A tabela *main_offer_rides* representa uma oferta de boleia e encontra-se interligada com os dados do condutor presentes na tabela *main_driver* através da chave estrangeira *driver_name_id*. Uma oferta de boleia pertence a um único condutor. Semelhante ao *main_transport_orders*, o *main_offer_ride* inclui informação que descreve a oferta nomeadamente a morada e coordenadas de partida (*pickup_location*, *pickup_latitude*, *pickup_longitude*), informação de destino (*dropoff_location*, *dropoff_latitude*, *dropoff_longitude*) e a data e hora em que o utilizador pretende começar a viagem (*departure_date*). Uma vez que se pretendia que o condutor pudesse oferecer boleia, também na viagem de regresso foi incluída a coluna *return_date*, mas este aspeto não foi implementado na aplicação. Semelhante ao *main_transport_orders*, quando a oferta de boleia é associada a uma viagem a coluna *have_trip* é usada para controlar o estado da mesma. A *main_offer_ride* inclui também a coluna *route_info* para registar a informação do percurso otimizado depois de calculado.

A definição de uma viagem acontece com a introdução de um registo em *main_trips*, que interliga os pedidos de transporte (*main_transport_orders*) com as ofertas de boleia (*main_offer_ride*). Trata-se de uma simples tabela associativa, onde cada pedido de transporte pode apenas ter uma entrada, mas cada oferta de boleia pode ter múltiplas entradas onde o número de entradas é o número de passageiros na viagem.

Numa primeira análise e considerando a existência da tabela *main_trips*, a coluna *have_trip* das tabelas *main_transport_orders* e *main_offer_ride* são redundantes. Bastaria pesquisar na *main_trips* para verificar se o pedido de transporte ou oferta de boleia tinha viagem atribuída. No entanto, a inclusão da coluna *have_trip* simplifica a interpretação da *query* usada para encontrar os pedidos sem viagem e as ofertas sem viagem, evitando a operação de *join* na linguagem SQL. Esta desnormalização da estrutura é intencional. Neste caso, o momento da escrita na criação de

uma viagem é ligeiramente mais complexo exigindo o uso de uma transação para garantir que as três tabelas são atualizadas de forma atómica.

A tabela *main_driver* serve o registo de informação a condutores. Esta informação inclui nome, morada, NIF, número da carta de condução. O condutor pode registar os veículos que vai utilizar no sistema que ficam armazenados na tabela *main_vehicle_trip* que inclui informação de marca, modelo, matrícula, capacidade e a possibilidade de transporte de carga. A interligação com a tabela *main_driver* é efetuada através da chave estrangeira *driver_id*. Nesta iteração do sistema, não foi incluída a ligação que regista qual o veículo a usar na viagem caso o condutor tenha mais que um veículo.

As tabelas *main_driver* e *main_passenger* não são necessárias se quiséssemos ter uma estrutura de base de dados rigorosamente normalizada. A sua inclusão desnormaliza a estrutura e cria alguma redundância em alguns campos, como por exemplo, o do nome que existe tanto na tabela de passageiros (*main_passenger*) ou do condutor (*main_driver*) como também do utilizador (*main_user*). Podia ter a ligação direta entre *main_transport_orders* com *main_user* e a *main_offer_ride* com *main_user*. No entanto, condutores e passageiros requerem alguma informação distinta – um condutor tem informação de NIF e carta de condução que o passageiro não necessita. Estes dados auxiliares que descrevem estas entidades poderiam ser incluídos na tabela *main_user*, mas iria gerar alguns registos com dados incompletos. A obrigatoriedade dos dados, conforme o contexto do condutor ou passageiro, não seria garantida na base de dados, porque caso fossem incluídos na tabela *main_user* teriam de permitir campos nulos em alguma parte do seu tempo de vida. De realçar que uma vez que estamos num ambiente de partilha de viagens, nem o condutor nem os passageiros sabem com quem vão e se conhecem alguém numa determinada viagem.

A gestão de utilizadores é efetuada numa estrutura de tabelas oferecida pelo Django, sendo este um dos benefícios de utilizar a *framework* para o desenvolvimento da aplicação. O Django inclui os mecanismos necessários para efetuar registo, autenticação e autorização de utilizadores numa aplicação *web*. A tabela central desta estrutura é o *main_user* que, além de dados pessoais do utilizador, serve para interligação com os dados do domínio. A tabela *main_user*, inclui as colunas *is_driver*, *is_passenger*, *interested_in_service_door_to_door* para auxiliar pesquisas.

A tabela *django_migration* é outra facilidade oferecida pelo Django que serve para controlar as migrações que são efetuadas ao modelo de base de dados da aplicação *web*, tal como já foi referido anteriormente.

As restantes tabelas que surgem isoladamente como *spatial_ref_sys*, *geography*_columns, geometry_*columns* pertencem à integração do PostGIS com o motor de base de dados PostgreSQL.

6.4 Pipeline de processamento

A *pipeline* de processamento é constituída por processos independentes, implementados em *python* que se encontram interligados por *queues* através do modo *publish* e *subscribe* no RabbitMQ. Nesta pipeline foram utilizados microsserviços interligados por *exchanges* do RabbitMQ.

A imagem da *pipeline* de processamento encontra-se no Anexo C.

6.4.1 Módulo de Trigger do pipeline de processamento

O módulo de *Trigger* é responsável por iniciar toda a pipeline de processamento periodicamente (minuto a minuto). O *Trigger* é efetuado por publicação de uma mensagem numa *queue* específica à qual o primeiro processo de módulo de *Clustering* está à escuta. A gestão de publicação desta mensagem é conseguida através do componente *beat* do *Celery*. A mensagem publicada pelo *beat* é uma mensagem vazia não havendo necessidade de parametrização da *pipeline*.

6.4.2 Módulo de Clustering

O módulo de *Clustering* é composto por um conjunto de processos *python* com a responsabilidade de obter os pedidos sem viagem, agrupá-los por pedido e hora e executar os processos de *Clustering* para definir o grupo de passageiros para uma viagem. A utilização de processos *python*, que estão à escuta de mensagens em *queues* específicas, faz com que estejam sempre disponíveis para processar e apenas executam quando têm de reagir a uma mensagem disponível na sua *queue* de entrada.

A utilização de múltiplos processos, permite distribuir o trabalho por múltiplas instâncias conforme a carga do sistema. No entanto, para o volume de informação disponível, considera-se necessário apenas uma instância de cada processo. Para um volume maior de informação, seria possível escalar horizontalmente o processo de *Clustering*. Neste caso, os vários processos estariam à escuta de mensagens na mesma *queue* e cada um consumia num esquema de *Round-Robin*, publicando resultados apenas se produzissem *clusters* válidos. De notar que, os primeiros dois processos do módulo (*read_messages.py* e *filter_results.py*) não permitem escalonamento horizontal de momento, porque só pode existir um processo de cada devido à forma como trabalham a mensagem da lista de pedidos.

O processamento dos pedidos no módulo de *clustering* é feito assincronamente sem interação com o utilizador, sendo que, apenas o primeiro processo de obtenção de pedidos sem viagem, interage com a API. Toda a transferência entre processos é feita com recurso a *queues* no RabbitMQ e as mensagens contêm os dados formatados em JSON.

6.4.3 Módulo de Match entre ofertas de boleia e pedidos de boleia

O módulo de *Match* é composto por um processo *python* com a responsabilidade de consumir uma mensagem com um grupo de pedidos gerados pelo módulo de *Clustering* e, com base na informação geográfica do grupo e as ofertas de boleia disponíveis na base de dados, encontrar a oferta mais adequada a realizar a viagem pedida pelos passageiros. Cada ciclo de processamento, que neste caso está associado a uma mensagem lida da *queue*, refere-se apenas a um único grupo dos *clusters* gerados com o máximo de quatro passageiros.

Este módulo, conforme está implementado, não pode ser escalado horizontalmente devido à possibilidade de haver colisão de atribuição de viagens à mesma oferta de boleia se os grupos processados em paralelo tivessem as condições de gerar o *match*.

Este módulo integra com a base de dados no sentido de atualizar os pedidos e as ofertas de boleia para marcá-las como tendo viagem e evitar o reprocessamento na iteração seguinte da pipeline.

6.4.4 Módulo de Otimização da rota

O módulo de otimização é composto por um processo *python* com a responsabilidade de consumir uma mensagem gerada pelo módulo de *Match* anterior que é composto pela informação dos pedidos e da boleia associada. Com base nessa informação, o módulo de otimização calcula a melhor rota a seguir pelo condutor com recurso a uma API externa privada alojada pela Ubiwhere, denominada OSRM. De realçar que os pedidos, apesar de próximos, são dispersos geograficamente, sendo que a rota pretende visitá-los da forma mais otimizada possível.

Este módulo integra com a base de dados no sentido de atualizar a viagem com a informação da melhor rota a seguir pelo condutor.

7 Implementação

Dados os requisitos apresentados nos capítulos anteriores, neste capítulo pretende-se descrever todo o trabalho realizado em termos de implementação do projeto, nomeadamente a aplicação web de backoffice, o serviço de simulação de dados, o módulo de clustering e otimização da rota e a determinação dos parâmetros para a técnica de clustering DBSCAN.

7.1 Implementação da aplicação web de backoffice

No início do segundo semestre deu-se início ao desenvolvimento do projeto e começou-se, primeiramente, por desenvolver uma aplicação que pudesse registar pedidos de transporte por parte de passageiros e ofertas de boleia por parte de condutores. A aplicação foi desenvolvida com recurso à *framework* de *python*, **Django** com auxílio do motor de base de dados **PostgreSQL**, tal como se encontra mencionado no capítulo 4. Esta possui dois tipos de utilizadores: **passageiros** e **condutores.**

Conforme está descrito nos diagramas de caso de uso constantes da Figura 17 da secção 5.1.3, referente aos requisitos funcionais, tanto o utilizador como o passageiro podem autenticar-se na aplicação, desde que se encontrem devidamente registados na mesma, conforme figuras 34 a 37 do anexo D.

Os passageiros podem registar os pedidos de transporte que pretendem. Para isso no formulário, basta preencherem como *inputs* os seguintes campos: **localização de partida, localização de destino, data e hora do pedido.** Além disso, podem visualizar os pedidos de transporte que efetuaram e se quiserem também podem tornar-se um condutor e registar ofertas de boleia e veículos. Já os condutores, podem não só registar os seus veículos, como também registar ofertas de boleia, bastando preencher os seguintes campos: **localização de partida, localização de chegada, data e hora de partida** e **data e hora de retorno.** Além disso, podem visualizar os seus veículos e as ofertas de boleia que registaram na aplicação.

Para além da aplicação *web* desenvolvida, foi também disponibilizada uma API REST que permite a obtenção de dados e a realização de várias operações de *Create*, *Read*, *Update* e *Delete* (CRUD) no sistema. Para ter acesso a cada um dos *endpoints* da API, é necessário estar devidamente autorizado na plataforma desenvolvida. Para assegurar a autorização, foram utilizados os mecanismos de armazenamento de utilizadores e geração de *tokens* de autorização oferecidos pelo Django.

7.1.1 Definição da API

7.1.1.1 Registo de pedidos de boleia

O método da API que contém o registo de pedidos de boleia por parte dos passageiros contém:

- **URL usado:** http://127.0.0.1:8000/api/list_transport_orders_test/
- **Estrutura da mensagem do pedido POST:**

  {
  pickup_location:,
  pickup_latitude:,
  pickup_longitude:,
  dropoff_location:,
  dropoff_latitude:,
  dropoff_longitude:,
  date_of_request:,
  id_passenger: }

- **Definição de cada campo:**
  - *pickup_location:* representa a localização de partida de um pedido de boleia;
  - *pickup_latitude:* representa a latitude da localização de partida de um pedido de boleia;
  - *pickup_longitude:* representa a longitude da localização de partida de um pedido de boleia;
  - *dropoff_location:* representa a localização de chegada de um pedido de boleia;
  - *dropoff_latitude:* representa a latitude da localização de chegada de um pedido de boleia;
  - *dropoff_longitude:* representa a longitude da localização de chegada de um pedido de boleia;
  - *date_of_request:* representa a data e hora de um pedido de boleia;
  - *id_passenger:* representa o id do passageiro que requisitou o pedido de boleia.

- **Tipo de resposta enviada:**

  {
  pickup_location: ,
  pickup_latitude: ,
  pickup_longitude: ,
  dropoff_location: ,
  dropoff_latitude: ,
  dropoff_longitude: ,
  date_of_request: ,
  id_passenger} seguido do estado do POST.

- **Dados registados na base de dados:**
  {
  id (gerado de forma automática e sequencial),
  pickup_location,
  pickup_latitude,
  pickup_longitude,
  dropoff_location,
  dropoff_latitude,
  dropoff_longitude,
  date_of_request,
  id_passenger,
  have_trip(campo booleano que por defeito vem a falso, significando que o pedido de transporte ainda não tem viagem atribuída)}

7.1.1.2 Registo de oferta de boleia

- **URL usado:** http://127.0.0.1:8000/api/list_offer_ride/

- **Estrutura da mensagem do pedido POST:**

  {
  pickup_location:,
  pickup_latitude:,
  pickup_longitude:,
  dropoff_location:,
  dropoff_latitude:,
  dropoff_longitude:,
  departure_date:,
  return_date:,
  id_driver}

- **Definição de cada campo:**
    - *pickup_location:* representa a localização de partida de uma oferta de boleia;
    - *pickup_latitude:* representa a latitude da localização de partida de uma oferta de boleia;
    - *pickup_longitude:* representa a longitude da localização de partida de uma oferta de boleia;
    - *dropoff_location:* representa a localização de chegada de uma oferta de boleia;
    - *dropoff_latitude:* representa a latitude da localização de chegada de uma oferta de boleia;
    - *dropoff_longitude:* representa a longitude da localização de chegada de uma oferta de boleia;
    - *departure_date:* representa a data e hora de partida de uma oferta de boleia;
    - *return_date:* representa a data e hora de chegada de uma oferta de boleia;
    - *id_driver:* representa o id do condutor que irá realizar a oferta de boleia.

- **Tipo de resposta enviada:**

{
pickup_location:,
pickup_latitude:,
pickup_longitude:,
dropoff_location:,
dropoff_latitude.,
dropoff_longitude:,
departure_date:,
return_date:,
id_driver}, seguido do estado do POST.

- **Dados registados na base de dados:**

{
id(gerado de forma automática e sequencial),
pickup_location,
pickup_latitude,
pickup_longitude,
dropoff_location,
dropoff_latitude,
dropoff_longitude,
departure_date,
return_date,
id_driver,
have_trip(campo booleano que por defeito vem a falso, significando que a oferta de boleia ainda não tem viagem atribuída)}

7.1.1.3 Registo de veículos

- **URL usado:** http://127.0.0.1:8000/api/list_vehicles/

- **Estrutura da mensagem do pedido POST:**

{
brand_vehicle:,
model_vehicle:,
matriculation:,
capacity:,
is_cargo_transport:,
id_driver:}

- **Definição de cada campo:**
  - *brand_vehicle:* representa a marca do veículo;
  - *model_vehicle:* representa o modelo do veículo;
  - *matriculation:* representa a matrícula do veículo;
  - *capacity:* representa a capacidade do veículo;
  - *is_cargo_transport:* representa se o transporte é carga ou não;
  - *id_driver:* representa o id do condutor que é responsável por aquele veículo.

- **Tipo de resposta enviada:**

  {
  brand_vehicle:,
  model_vehicle:,
  matriculation:,
  capacity:,
  is_cargo_transport:,
  id_driver:}, seguido do estado do POST.

- **Dados registados na base de dados:**

{
id_vehicle(gerado de forma automática e sequencial),
brand_vehicle,
model_vehicle,
matriculation,
capacity,
is_cargo_transport,
id_driver}

7.2 Implementação do serviço de simulador de dados

Posteriormente a ter sido desenvolvida a aplicação *web*, foi realizado num projeto à parte, um *script* para inserir vários dados na aplicação desenvolvida, de forma a ser possível aplicar o módulo de *clustering* e otimização de rota sobre os mesmos. Este *script* surge sobretudo para registar pedidos de transporte por parte dos passageiros e ofertas de boleia por parte de condutores, sendo que para isso foi necessário ter em consideração os URLs desses métodos desenvolvidos anteriormente para a API. O processo de inserção de pedidos de transporte e ofertas de boleia foi feita via API REST com recurso ao método HTTP POST (utilizado quando se pretende inserir novos dados). Como *headers* dos métodos da API, basta introduzir o *token* de um dado utilizador (passageiro ou condutor) existente na aplicação. O processo de escolha dos condutores associados à criação de ofertas de boleia e o processo de escolha dos passageiros associados à criação de pedidos de transporte foi gerado de forma aleatória, sendo que as respetivas datas dos pedidos de transporte, ofertas de boleia e os ID dos utilizadores também foi gerado de forma aleatória. Esta

geração de pedidos abrangeu o período de 1 ano. A cada um destes ID, gerados aleatoriamente, foi associado o respetivo *token*.

Para as localizações das ofertas de boleia e pedidos de transporte serem geradas aleatoriamente foi efetuado um pequeno *script* através dos dados disponibilizados pelo Instituto Geográfico Português. Neste *dataset* existem 2 *shapefiles*, um que diz respeito às estradas e outra que diz respeito às localidades de Portugal. Desta forma, foi importado através do PostGIS, os *shapefiles* para uma tabela e seguidamente foram intersetadas as estradas que cruzam a freguesia de Santo António dos Olivais e que cruzam o concelho de Coimbra, sendo que depois estas estradas foram escolhidas aleatoriamente. A opção pela região de Coimbra deveu-se ao facto de ser um território familiar, tendo sido escolhida para efeito de testes.

Ao todo o *dataset* inclui 2891 pedidos que se encontram geograficamente representados na figura 20. Estes estão distribuídos por 355 dias distintos no período entre 15/04/2019 e 31/12/2020, sendo que aproximadamente 80% dos dias têm 10 ou menos pedidos. A figura 21 mostra a distribuição dos pedidos ao longo do tempo. Analisando por períodos de hora usados pelo módulo do *clustering*, os pedidos de boleia estão distribuídos por 1561 blocos horários. 62 % destes blocos horários apenas têm um pedido de boleia registado e como consequência não terão viagens geradas porque não cumprem o requisito mínimo de geração de viagem. Dos restantes blocos horários que têm possibilidade de gerar viagens, 86% têm entre 2 e 4 pedidos. O período com maior número de pedidos inclui 41 pedidos de boleia (conforme se demonstra na tabela 4).

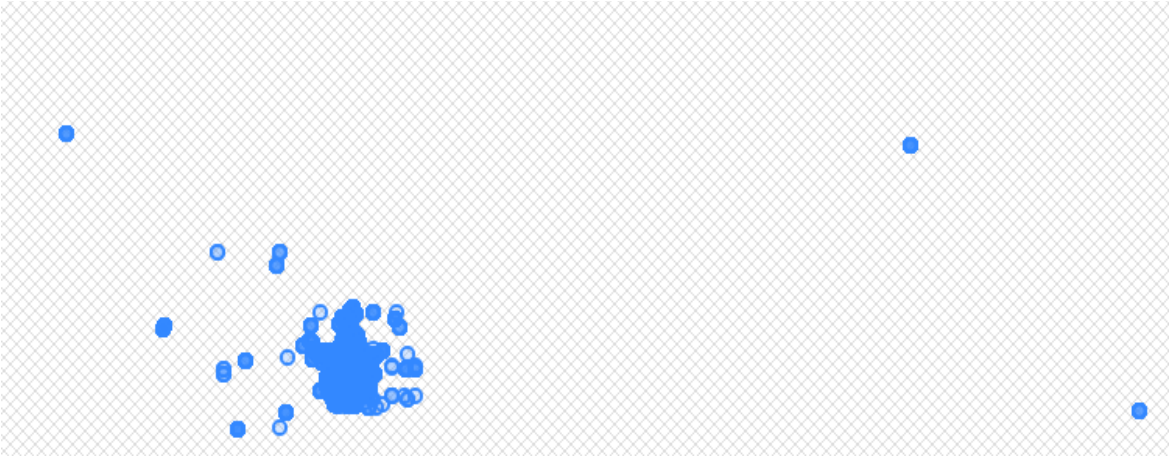


Figura 20 - Distribuição geográfica do conjunto de pontos de partida do *dataset*

Figura 21- Distribuição dos pedidos do *dataset* por dias

Tabela 4- Distribuição de pedidos de boleia por períodos de data/hora no *dataset*

Número de pedidos no período data/hora	**Número de períodos**
1	966
2	324
3	121
4	64
5	32
6	13
7	12
8	8
9	4
10	5
11	1
12	1

13	1
14	2
15	3
16	1
19	1
41	1

7.3 Implementação do módulo de clustering e otimização de rota

Este módulo foi dividido em vários serviços *python* de forma a ser possível isolar as tarefas. Estas encontram-se interligadas através de *queues* do RabbitMQ. Cada serviço publica uma ou várias mensagens conforme a estrutura e quem lê essas mensagens utiliza essa informação para executar a sua tarefa. O arranque deste trabalho é gerido pelo módulo Celery que periodicamente publica uma mensagem de início de execução. O diagrama de *workflow* deste módulo encontra-se no anexo B.

O ficheiro "*start_process.py*" é responsável por configurar e iniciar o Celery. A configuração define a periodicidade do trabalho (minuto a minuto), a *queue* para onde deve ser escrita a mensagem e a tarefa a executar. Uma vez iniciado, o Celery publica periodicamente uma mensagem para a *queue* denominada *"retrieve_orders"* de modo a lançar a cadeia de processamento, conforme apresentado na figura 22.

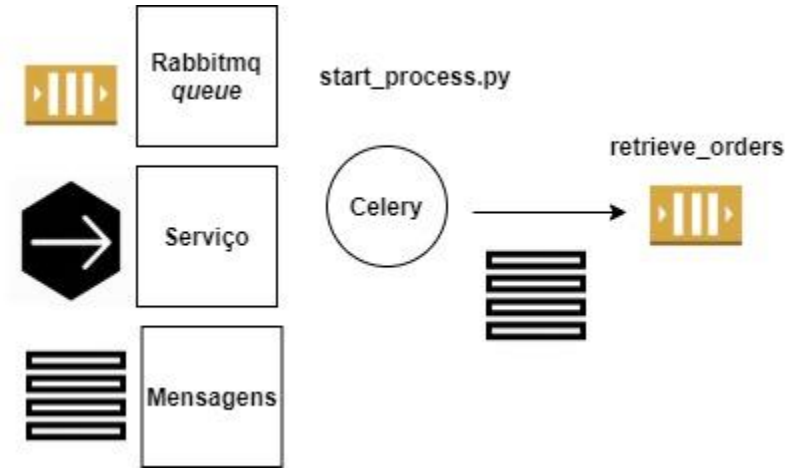


Figura 22 - Diagrama do processamento "*start_process*"

O segundo processo é o "*read_messages.py*" cujo objetivo, sempre que chega uma mensagem na *queue* "*retrieve_orders*", passa por ler a lista de pedidos de transporte dos passageiros proveniente da API que ainda não têm viagem atribuída, sendo posteriormente publicada uma mensagem com a lista na *queue* "*pending_orders*", tal como se pode visualizar na figura 23.

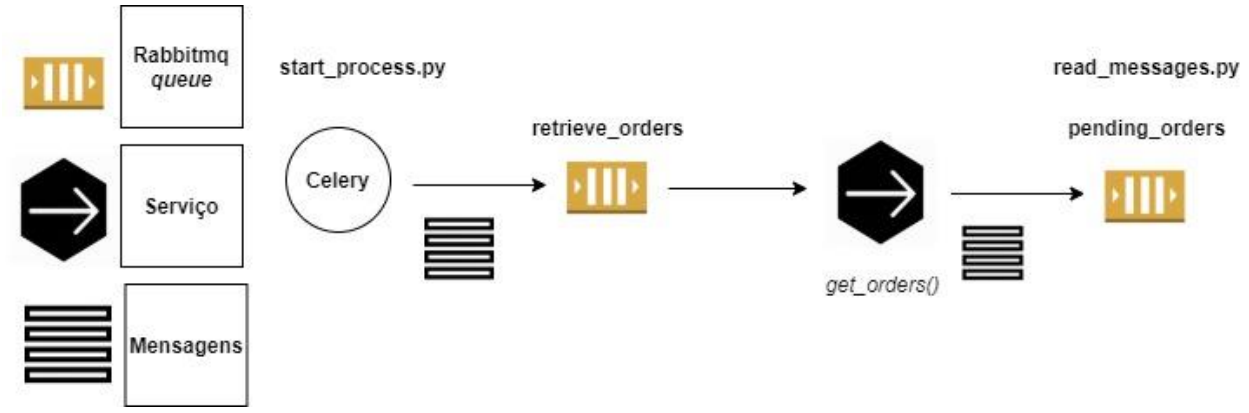


Figura 23 - Diagrama da transição do processamento entre "*start_process.py*" e "*read_messages.py*"

As entradas e saídas do processo "*Read_Messages.py"* podem ser visualizadas na tabela 5.

Tabela 5 - Entradas e saídas do processo "*Read_Messages.py*"

Processo	Read_Messages.py
Trigger	Mensagem na *queue* "Retrieve Orders"
Dados de entrada	Mensagem vazia
Dados de saída	Mensagem com a lista de pedidos de transporte sem viagem
***Queue* de destino**	*"Pending Orders"*

Já o processo denominado "*filter_results.py*" lê as mensagens que se encontram na *queue* "*pending_orders*", ou seja, recebe a lista de pedidos de transporte dos passageiros que ainda não têm viagem atribuída e divide as mesmas em grupos de pedidos por dia e hora, sendo cada grupo composto por dois ou mais pedidos publicados para a *queue* "*filter_orders*", conforme se pode visualizar pela figura 24. O objetivo de considerar um número mínimo de pedidos num grupo é de eliminar a possibilidade de gerar viagem com apenas um passageiro.

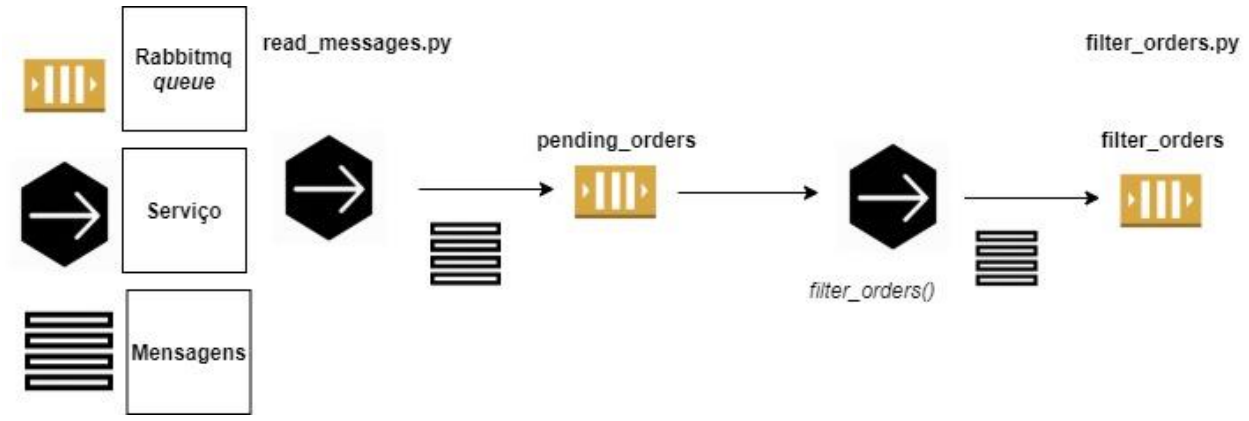


Figura 24- Diagrama da transição do processamento entre "*read_messages.py*" e "*filter_orders.py*"

As entradas e saídas do processo "*Filter_Results.py"* podem ser visualizadas na tabela 6.

Tabela 6 - Entradas e saídas do processo "*Filter_Results.py*"

Processo	Filter_Results.py
Trigger	Mensagem na *queue* "*Pending Orders*"
Dados de entrada	Mensagem com lista de pedidos de boleia sem viagem associada
Dados de saída	Mensagem com lista de pedidos de boleia com o mesmo dia e hora
Número de mensagem de saída	Um por cada grupo de cada dia e hora encontrado
Limitação de publicação	Apenas se um grupo tiver dois ou mais pedidos
***Queue* de destino**	*"Filter Orders"*

O processo "*cluster_results.py*" lê as mensagens vindas da *queue* "*filter_orders*", ou seja, recebe, de cada vez, uma mensagem com os pedidos do mesmo dia e hora, sendo posteriormente calculado os *clusters* a aplicar ao conjunto, tal como se pode visualizar pela figura 27. Em termos de *clustering*, foi usado o DBSCAN, como resultado da comparação feita na secção 2.2.2, para agrupar os pedidos de transporte do mesmo dia e hora em termos de densidade.

Posteriormente, com os resultados do DBSCAN era necessário ter em consideração outro dos grandes objetivos para o estágio, que era restringir cada grupo no máximo com 4 pontos por cada *cluster*, ficando assim, otimizado o número de lugares de um veículo ligeiro com capacidade para 5 lugares. O algoritmo DBSCAN, nativamente, não tem nenhum parâmetro de limitação máxima de número de elementos por *cluster*.

Numa primeira abordagem foi criado um algoritmo de divisão do *cluster* em grupos de 4 elementos simples, mas sem qualquer parametrização relativamente à distância entre pontos e otimização de rota. O algoritmo dividia o *array* em grupos de 4, sendo cada grupo um *sub-cluster* do principal. A falta de consideração da relação entre pontos levou a resultados pouco ótimos, tal como se pode visualizar pela figura 25.

Figura 25 - Representação da falta de consideração da otimização entre os pontos

Foi efetuada uma pesquisa sobre algoritmos que pudessem ajudar a restringir esse número máximo de pontos por cada *cluster*, mas tendo em conta a proximidade entre pontos, e verificou-se que existe uma técnica de *clustering* adaptada do *K-Means* denominada de *Constrained K-means*. Esta tem a capacidade de limitar o número mínimo e máximo de pontos por cada *cluster* e de ao mesmo tempo garantir a otimização da distância entre os pontos, tornando-se uma ajuda preciosa na concretização deste objetivo. Sendo assim, a cada *cluster* resultante do DBSCAN foi aplicado o *Constrained K-means*. Desta forma, como qualquer K-Means exige a especificação do número de *clusters* no início do algoritmo, foi então optado por usar o DBSCAN com o valor do épsilon de 0,0005 e número mínimo de pontos o valor 2 num primeiro nível. Posteriormente, numa segunda fase, foi então aplicado o *Constrained K-means*, que é uma versão adaptada do K-means onde foi necessário especificar o número de *clusters* pretendidos e o número mínimo e máximo de pontos em cada *cluster*. O número de *clusters* para o Constrained K-Means, é o número de pontos do cluster do DBSCAN a dividir por quatro (máximo de pontos em cada cluster). Esse valor é incrementado em um se houver resto da divisão por quatro para não haver nenhum ponto sem *cluster*. Como valores dos parâmetros do número mínimo de pontos em cada *cluster* foi colocado 2 e como número máximo de pontos em cada *cluster* foi colocado 4. Estas opções deveram-se ao facto de como uma oferta de boleia possui um condutor e este possui um veículo, um dos requisitos do projeto era conter no mesmo, o mínimo de dois e no máximo quatro passageiros.

Com base nos resultados do *clustering* foram calculados os centroides das localizações de partida de cada *sub-cluster*, os *bounding-boxes* definidos pelos pontos de partida de cada *sub-cluster* e os *bounding-boxes* definidos pelos pontos de destino de cada *sub-cluster*. Estes três parâmetros auxiliam o desenho dos mapas e o processamento dos *matches* com as ofertas de boleia dos

condutores. Particularmente, os *bounding-boxes* servirão como parâmetro de pesquisa para encontrar as ofertas de boleia adequadas ao *sub-cluster*. Pretende-se que o ponto de partida da oferta de boleia esteja inserido no espaço geográfico de *bounding-box* dos pontos de partida do *sub-cluster* e que o ponto de destino da oferta de boleia esteja inserido no espaço geográfico de *bounding-box* dos pontos de chegada do *sub-cluster*.

Finalizado o processo de *clustering*, é publicada uma mensagem por *sub-cluster* na *queue* "*clustered_orders*" com os pontos do *sub-cluster*, os centróides e os limites.

Para auxiliar o processo de desenvolvimento e visualização dos resultados do processo de *clustering*, o *"Cluster_Results.py"* inclui o desenho e gravação de um ficheiro com os pontos de *pickup* de cada *sub-cluster* desenhados sobre o mapa com distinção de cor entre *sub-clusters*. A figura 26 apresenta um exemplo desses mapas, onde são visíveis pela cor e proximidade dos pontos os vários *sub-clusters* resultantes do *Constrained K-Means*.

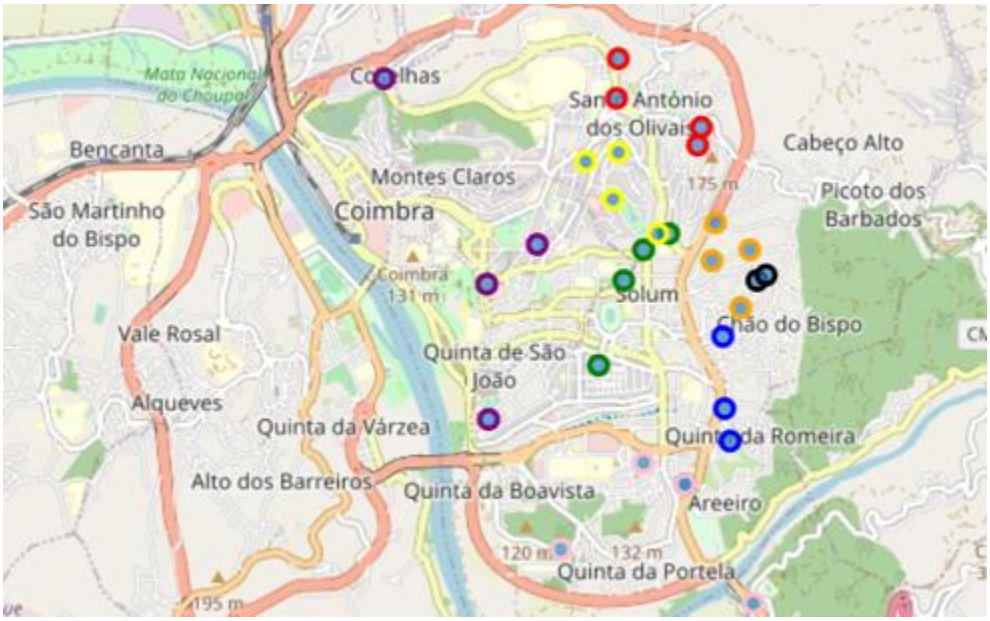


Figura 26 - Exemplo de mapa com os pontos dos vários *sub-clusters* gerados para a mesmo dia e hora

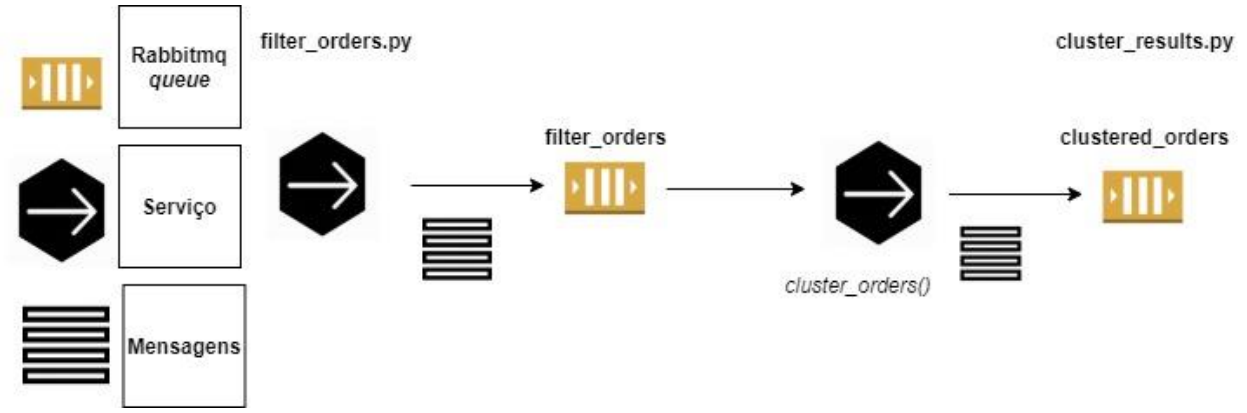


Figura 27 - Diagrama da transição do processamento entre "*filter_orders.py*" e "*cluster_results.py*"

As entradas e saídas do processo "*Cluster_Results.py"* podem ser visualizadas na tabela 7.

Tabela 7 - Entradas e saídas do processo "*Cluster_Results.py*"

Processo	Cluster_Results.py
Trigger	Mensagem na *queue* "*filter_orders*"
Dados de entrada	Mensagem com lista de pedidos de boleia com o mesmo dia e hora
Dados de saída	Mensagem com a lista de pontos de um *sub-cluster* com o máximo de 4 pontos e inclui os centroides e limites geográficos do *cluster*
Número de mensagem de saída	Um por cada *sub-cluster* encontrado
Limitação de publicação	Apenas se tiver dois ou mais pontos no *sub-cluster*
***Queue* de destino**	*"Clustered_Orders"*

O processo "*match_offer_rides_and_orders.py*" está encarregue de ler as mensagens vindo da *queue* "*clustered_orders*", ou seja, de receber por cada mensagem na *queue* um grupo de pedidos que pertence a um só *cluster*, tal como se encontra apresentado na figura 27. Posteriormente, é feito o *match* das ofertas de boleia com os *clusters*. Este *match* possui várias condições, sendo elas:

- A data e hora de partida das ofertas de boleia dos condutores têm de ser iguais à data e hora dos pedidos agrupados pelo *clustering*;
- A zona de localização de partida das ofertas de boleia tem de pertencer aos limites de localização de partida do *cluster*;
- A zona de localização de destino das ofertas de boleia têm de pertencer aos limites de localização de destino do *cluster*.

De seguida para cada grupo de *clusters* e já com a informação do match de várias ofertas de boleia, foi calculado através do *haversine* a distância entre as ofertas de boleia ao centro de partida de cada *cluster*, de forma a ser possível calcular qual das ofertas de boleia é a que possui menor distância ao centro de partida do *cluster*. Com a informação da oferta de boleia, que tem menor distância do centro de partida de cada *cluster*, é fornecida a informação do veículo e do condutor que é responsável por levar os passageiros aos seus destinos, sendo depois, no final deste processo, especificamente marcado como viagem atribuída a essa mesma oferta de boleia e esses mesmos pedidos de transporte. No final é publicada uma mensagem na *queue* "*match_offer_ride_and_orders*" com a informação relativa à oferta de boleia escolhida, à informação do veículo, do condutor, do centro de partida do *cluster* e dos pontos agrupados pelo mesmo.

A partir deste passo de *workflow*, os pedidos de transporte e ofertas de boleias para a qual haja *match* deixam de fazer parte dos próximos processamentos deste passo, bem como da nova iteração de todo o *workflow*.

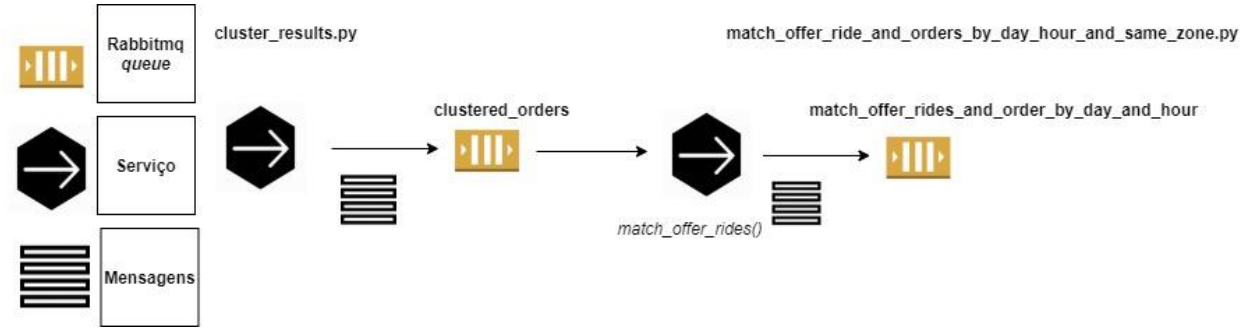


Figura 28 - Diagrama da transição do processamento entre "*cluster_results.py*" e "*match_offer_rides_and_orders.py*"

As entradas e saídas do processo "*match_offer_rides_and_orders_by_day_hour_and_same_zone.py"* podem ser visualizadas na tabela 8.

Tabela 8 - Entradas e saídas do processo " *match_offer_rides_and_orders_by_day_hour_and_ same_zone. py*"

Processo	"*match_offer_rides_and_orders_by_day_hour_and_same_zone.py*"
Trigger	Mensagem na *queue* "*clustered_orders*"
Dados de entrada	Mensagem com a lista de pontos de um *sub-cluster* com o máximo de 4 pontos e inclui os centroides e limites geográficos do *cluster*

Dados de saída	Mensagem com informação de oferta de boleia escolhida, do veículo, do condutor, centro de partida do *cluster* e lista de pontos a visitar
Número de mensagem de saída	Um por cada *match* encontrado
Limitação de publicação	Apenas se tiver *match* entre pedidos do *cluster* e ofertas de boleia
***Queue* de destino**	*"match_offer_rides_and_orders_by_day_and_hour"*

Por fim, o processo final "*otimization_route*", tal como o nome indica, é responsável por ler a *queue* "*match_offer_ride_and_orders*" e calcular através dessa informação a rota ótima, conforme se pode visualizar pela figura 29. Para esse processo foi utilizado o OSRM da empresa, sendo este um serviço que fornece uma API com vários métodos para determinar rotas em redes rodoviárias. Foi optado por utilizar o *endpoint Trip service* que utiliza um algoritmo de força bruta para menos de 10 pontos e que por defeito se baseia no problema do caixeiro viajante, fazendo com que garanta uma aproximação da melhor rota. Além disso, uma das grandes razões pelo qual foi escolhido este método foi o facto de se poder especificar o ponto de localização do início e de fim da viagem, uma vez que como ponto de partida da viagem irá ser a localização de partida da oferta de boleia do condutor e como ponto de destino da viagem irá ser a localização de destino da oferta de boleia do condutor. Tendo em conta o que foi mencionado anteriormente e os pedidos agrupados pelo *clustering* é calculada a rota ótima com as várias paragens que o condutor terá de fazer, sendo posteriormente essa informação publicada na *queue* "*otimization_route*" e sendo atualizado na tabela de base de dados das ofertas de boleia a informação com essa mesma rota daquela oferta específica.

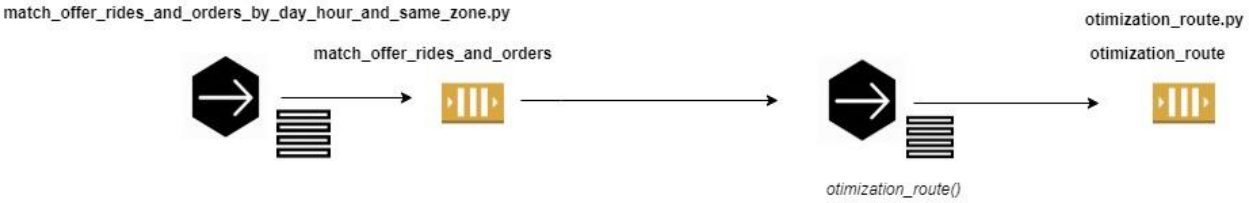


Figura 29 - Diagrama da transição do processamento entre "match_offer_rides_and_orders_by_day_hour_and_same_zone.py" e "otimization_route.py"

As entradas e saídas do processo "*otimization_route.py"* podem ser visualizadas na tabela 9.

Tabela 9 - Entradas e saídas do processo otimization_route.*py*"

Processo	"*otimization_route.py*"
Trigger	Mensagem na *queue* "*match_offer_ride_and_orders*"

Dados de entrada	Mensagem com informação de oferta de boleia escolhida, do veículo, do condutor, centro de partida do *cluster* e lista de pontos a visitar
Dados de saída	Mensagem com informação da rota que o condutor terá de fazer
Número de mensagem de saída	Um por cada mensagem com informação de oferta de boleia escolhida, do veículo, do condutor, centro de partida do *cluster* e lista de pontos a visitar
Limitação de publicação	Apenas se tiver *match* entre pedidos do *cluster* e ofertas de boleia
***Queue* de destino**	*"otimization_route"*

7.4 Determinação dos Parâmetros para a Técnica de Clustering DBSCAN

A execução da técnica de *clustering* DBSCAN requer *à priori* a definição dos valores dos parâmetros *épsilon* e *min_points*. Uma das grandes dificuldades referentes à técnica de DBSCAN passa pela especificação dos valores dos parâmetros na configuração inicial do algoritmo. De acordo com a literatura, verificou-se a existência de um método dinâmico que tem a capacidade de encontrar o valor adequado do parâmetro *épsilon* para cada nível de densidade de um dado conjunto de dados, cujo nome é DMDBSCAN [97] [98]. O pseudocódigo deste algoritmo encontra-se retratado na figura 30.

Algorithm 1. The pseudo code of the proposed technique DMDBSCAN to find suitable Epsi for each level of density in data set.

Purpose:	1. *To find suitable values of Eps*
Input:	2. *Data set of size n*
Output:	3. *Eps for each varied density*
Procedure:	4. $For\ i = 1\ to\ n$ 5. $For\ j = 1\ to\ n$ 6. $d(i,j) \leftarrow find\ distance(x_i, x_j)$ 7. $find\ minimum\ values\ of\ distances\ to\ nearest\ 3$ 9. $end\ for$ 10. $end\ for$ 11. $sort\ distances\ ascending\ and\ plot\ to\ find\ each\ value$ 12. $Eps\ corresponds\ to\ critical\ change\ in\ curves$

Figura 30 - Pseudocódigo do algoritmo DMDBSCAN

Passando a uma explicação sucinta da figura 30, o algoritmo calcula a distância em cada par de funções de latitude e longitude até ao seu vizinho mais próximo, utilizando o algoritmo *Nearest Neighbors*. O ponto em si é incluído em *n_neighbors*. Posteriormente foi utilizado o método *kneighbors* da biblioteca *scikit-learn* que contém como parâmetro a lista de coordenadas das localizações de partidas dos pedidos de boleia dos passageiros e que retorna duas matrizes, uma que contém a distância para os pontos *n_neighbors* mais próximos e a outra que contém o índice para cada um desses pontos. Em seguida, foram ordenados de forma crescente pelos valores das distâncias e foram traçados os resultados, tendo sido obtido por fim um gráfico (Figuras 31 e 32) que tem no eixo dos x o índice dos pontos ordenados por distância crescente e no eixo dos y as respetivas distâncias. O gráfico indica através do ponto de curvatura máxima o valor ideal a ser utilizado no parâmetro do *épsilon* do *DBSCAN*.

Desta forma foi efetuado um caso de estudo que se encontra armazenado no GitHub de forma pública, podendo ser acedido através do seguinte url:

https://github.com/pedroafonsoo/Determination_of_optimal_value_of_epsilon_for_dbscan/blob/master/Clustering_test_dataset_internship.ipynb

Relativamente ao caso de estudo, o valor ideal do *épsilon* pode ser determinado aplicando o algoritmo de *Nearest Neighbors* sobre o *dataset* da API referente aos pedidos de transporte introduzidos pelos passageiros, mais concretamente sobre os pontos de *pickup* dos mesmos. O valor de *épsilon* é escolhido a partir de uma representação gráfica do resultado e é escolhido o valor da coordenada *y* do ponto que apresenta uma maior inflexão.

Essa mudança acentuada no valor do gráfico corresponde a um valor adequando de *épsilon* para cada nível de densidade do conjunto de dados.

Com base nos dados de localização de partida, foi executado o algoritmo de *Nearest Neighbors* para obter um valor de épsilon a usar no algoritmo DBSCAN e de seguida executou-se o algoritmo de DBSCAN e a remoção de pontos com ruído. Posto isto, para o conjunto de dados relativo aos pedidos de boleia dos passageiros foram obtidos como valor ótimo de *épsilon* 0.0005 (valor em radianos, aproximadamente 55 metros) e relativamente ao parâmetro *min_points*, foi escolhido o valor 2 derivado ao facto de não haver viagens para menos de dois passageiros. Esta restrição é um requisito do sistema que foi indicado pelo *product owner*. Para um novo conjunto de dados tem que ser determinado novamente um novo valor ótimo de *épsilon*.

O DBSCAN gerou 289 *clusters* (encontram-se representados na figura 33) com um coeficiente de silhueta de 0.957. Este coeficiente é uma métrica de desempenho do algoritmo de DBSCAN que utiliza a distância média entre pontos do mesmo *cluster* e a distância média entre *clusters* mais próximos. O valor da silhueta varia entre -1 e 1 sendo -1 a pior pontuação possível e 1 a melhor. O valor de 1 representa *clusters* de alta densidade (pontos internos muito próximos uns dos outros) e os *clusters* distantes entre si. Um valor de 0 sugere sobreposição de *clusters* [99] e um valor

negativo representa que uma amostra foi atribuída ao *cluster* errado. O valor obtido como coeficiente de silhueta indica um valor de silhueta forte traduzindo-se numa boa qualidade no resultado e significando que os pontos dos *clusters* se encontram muito próximos uns dos outros.

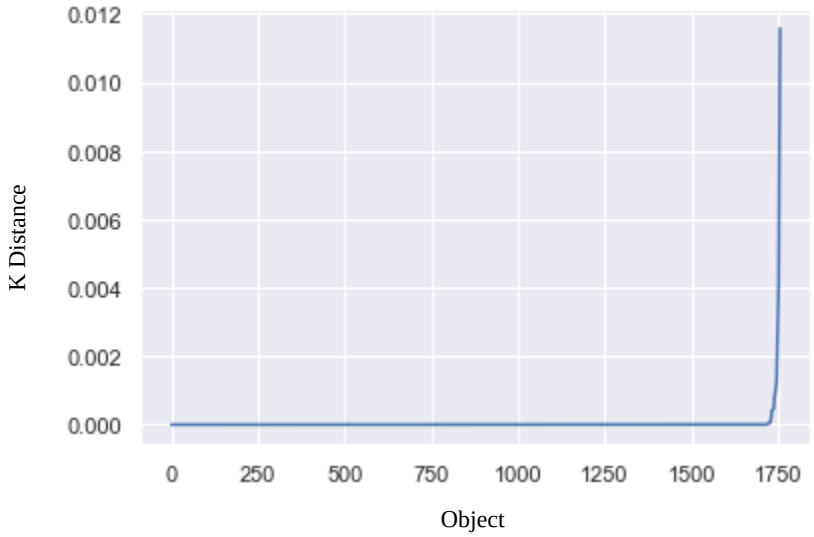


Figura 31 - Gráfico de determinação do valor ótimo do *épsilon*

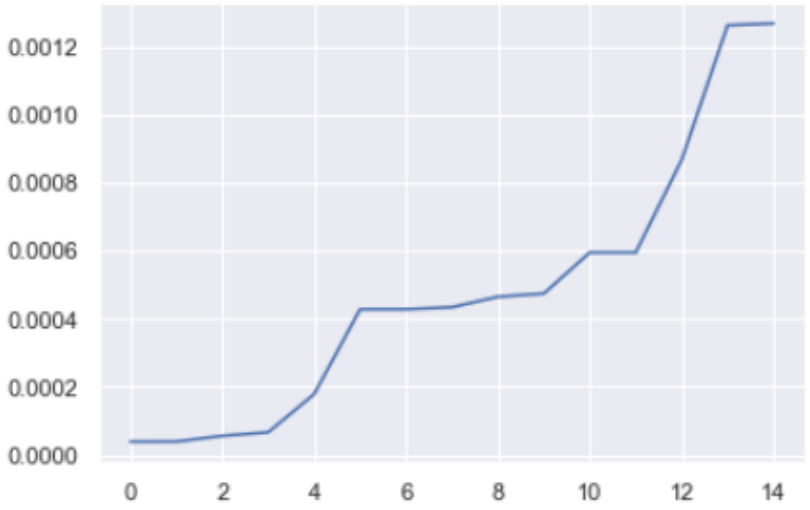


Figura 32 - Determinação do valor ótimo do *épsilon* de modo a focar a visualização na zona de inflexão

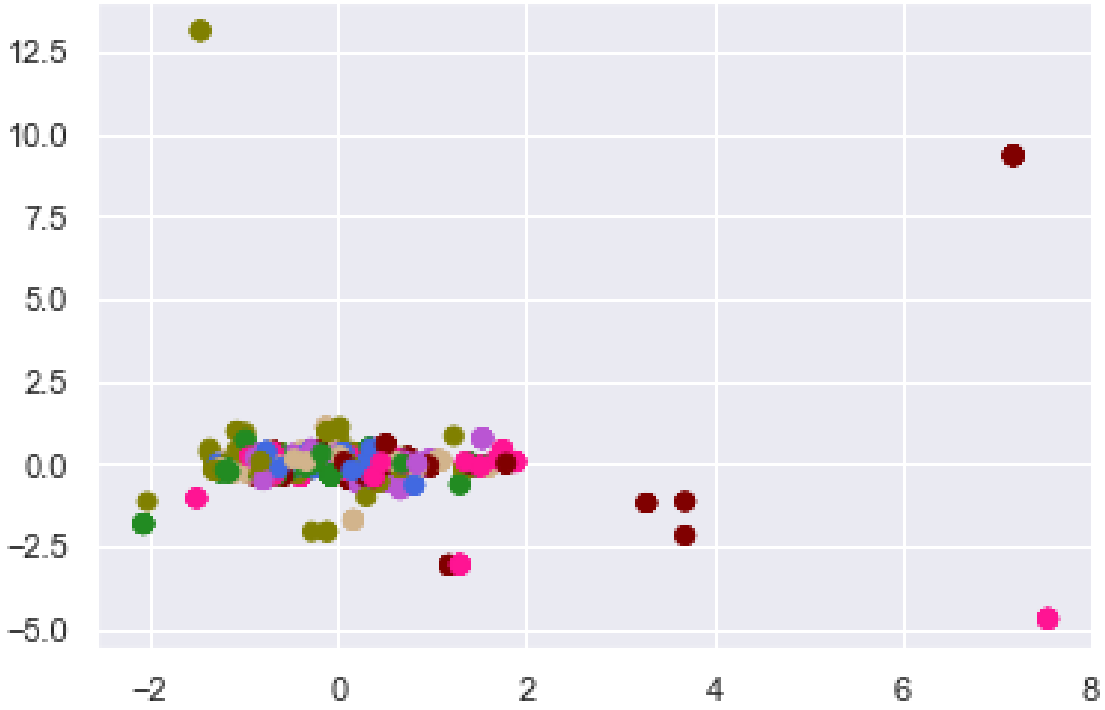


Figura 33 - Representação espacial dos pontos de *pickups* e dos *clusters* a que pertencem

8 Conclusões e Trabalho Futuro

Este capítulo apresenta uma reflexão sobre o trabalho desenvolvido, nomeadamente em termos de objetivos atingidos e competências adquiridas, assim como apresenta propostas de desenvolvimento futuro.

8.1 Conclusões

O objetivo principal do presente trabalho passou por agregar pedidos de transporte num sistema *on-demand*, agrupar as pessoas e garantir a otimização da rota e respetivos pontos de recolha. Com o trabalho desenvolvido, pode concluir-se que os seguintes objetivos foram em regra geral, concluídos com sucesso:

- Produção de um documento de estágio que inclui o estado da arte e decisões tecnológicas para o desenvolvimento da solução;
- Desenvolvimento de uma aplicação *web* para registar pedidos de transporte por parte dos passageiros e ofertas de boleia por parte de condutores e respetiva API REST;
- Desenvolvimento de um simulador de dados nomeadamente para o registo de pedidos de transporte e registo de ofertas de boleia com recurso à API que foi desenvolvida;
- Desenvolvimento de um módulo de *clustering* de agrupamento de pedidos de transporte;
- Atribuição de ofertas de boleia, pessoas e disponibilidade de veículos a *clusters* de pedidos de transporte.

No decorrer deste trabalho, pude ter contacto com novas tecnologias e alargar os conhecimentos, designadamente os seguintes:

- Compreender a importância da mobilidade no transporte;
- Aprendizagem sobre algoritmo de *clustering*;
- Adquirir novos conhecimentos ao nível de linguagens de programação;
- Utilizar novas ferramentas e tecnologias em contexto profissional;

- Trabalhar em equipa num ambiente profissional.

Inicialmente, foi trabalhado e testado o algoritmo DBSCAN sobre um conjunto de dados de táxis de Nova Iorque com o objetivo de apresentar o caso de estudo num artigo submetido e aceite para a conferência INTELLI 2020, conforme irá ser referido na secção 8.3.1.

Ao longo de todo o estágio, mais concretamente no desenvolvimento do módulo de *clustering* e otimização de rota, surgiram algumas dificuldades. Estas dificuldades, sobretudo na compreensão de novas técnicas de *clustering*, foram colmatadas com várias pesquisas tanto na internet como em vários artigos. As funcionalidades foram implementadas com sucesso e o estagiário cumpriu os requisitos que lhe foram apresentados. O ritmo da realização das tarefas propostas pela empresa foi aumentando, com o decorrer do estágio derivado à experiência e conhecimento que ia adquirindo na aplicação das linguagens de programação. Este estágio foi uma oportunidade de contacto com o mercado de trabalho e o ambiente empresarial, de forma a complementar as competências socioprofissionais, através da ligação entre o sistema educativo e mundo laboral.

No cômputo geral, o estagiário considera que o estágio foi desenvolvido com sucesso, tendo sido cumpridas todas as etapas propostas pela Ubiwhere.

8.2 Trabalho Futuro

Como trabalho futuro, são abordadas aqui, várias sugestões com o objetivo de melhorar o trabalho que foi desenvolvido:

- Na parte dos *matches* entre a oferta de boleia dos condutores e os pedidos de transporte dos passageiros agrupados pelo *clustering* poderia ser:
    - Criado um tópico específico para notificar os passageiros;
    - Criado um tópico específico para notificar os condutores com a inclusão da informação da rota que o mesmo irá fazer depois de ter sido publicado na base de dados.

    Desta forma, no final da otimização da rota depois de ter sido registado a viagem com a rota ótima na base de dados poderia ser publicada uma mensagem nas *queues* mencionadas anteriormente para executar o processo de notificação.
- Na tentativa do cálculo de *match*, poderiam ser excluídos os pedidos que já passaram no tempo (dia e hora) e notificar assim o condutor e o passageiro que nada vai ser associado.
- Poder-se-ia otimizar boleias já existentes no sentido de preencher os veículos que ainda não têm todos os lugares preenchidos quando surgem novos pedidos de boleia

e ainda se poderia considerar métodos com veículos com mais capacidade, ou até menos capacidade (com 2 lugares) e incluir informação de necessidade de transporte com carga ou sem carga.

- Inclusão de um módulo que seria responsável por gerir os custos das viagens dos condutores.
- Tornar a determinação dos parâmetros do clustering DBSCAN dinâmicos calculados a cada iteração, com base nos pontos de cada data/hora, em vez de utilizar um valor fixo a configurar para o sistema.

8.3 Contribuições

Este subcapítulo apresenta as principais contribuições em termos de investigação teórica e prática que efetuei.

8.3.1 Contribuições académicas

Paralelamente ao estágio, o artigo que tinha sido trabalhado no âmbito da unidade curricular de Seminários Industriais com o tema "*Comparison of Clustering techniques for On Demand Transport*", foi melhorado com o objetivo de ser submetido à nona conferência internacional sobre sistemas inteligentes e aplicações [100], INTELLI 2020, que vai ser realizado na cidade do Porto, entre os dias 18 e 22 de Outubro de 2020, tendo já sido confirmada a sua aceitação.

8.3.2 Contribuições práticas

Relativamente às contribuições práticas, o estagiário desenvolveu todo o projeto de raiz, tendo desenvolvido uma aplicação de *backend* que regista ofertas de boleia por parte de condutores e pedidos de boleia por parte de passageiros, um serviço de simulação de dados que garante a inserção dos mesmos de uma forma automática e ainda o módulo de *clustering* e otimização de rota.

Referências

[1] S. M. I. M. e. S. S. Shannon Bouton, "Urban mobility at a tipping point," 1 Setembro 2015. [Online]. Available: https://www.mckinsey.com/business-functions/sustainability/our-insights/urban-mobility-at-a-tipping-point. [Accessed 27 Julho 2020].

[2] "Ubiwhere," [Online]. Available: https://www.itjobs.pt/empresa/ubiwhere. [Accessed 4 Junho 2020].

[3] "Como as medidas de desestímulo ao uso do automóvel melhoram a mobilidade urbana," [Online]. Available: https://saopaulosao.com.br/%E2%80%98%E2%80%99/%E2%80%98%E2%80%99/nossos-caminhos/2705-como-as-medidas-de-desest%C3%ADmulo-ao-uso-do-autom%C3%B3vel-melhoram-a-mobilidade-urbana.html. [Accessed 20 Julho 2020].

[4] "Lipor-Mobilidade sustentável," [Online]. Available: https://www.lipor.pt/pt/residuos-conceitos-fundamentais/mobilidade-sustentavel-conceito/. [Accessed 30 Outubro 2019].

[5] "O que é a Mobilidade Urbana?," [Online]. Available: https://www.vivadecora.com.br/pro/arquitetura/o-que-e-mobilidade-urbana/. [Accessed 20 Julho 2020].

[6] "Plano de ação de mobilidade urbana sustentável," [Online]. Available: https://www.cm-pombal.pt/ordenamento-territorial/mobilidade-e-acessibilidade/plano-de-acao-de-mobilidade-urbana-sustentavel/. [Accessed 30 Outubro 2019].

[7] "Mobility on Demand," [Online]. Available: https://www.modalliance.org/. [Accessed 31 Outubro 2020].

[8] "Uber," [Online]. Available: https://www.uber.com/pt/pt-pt/. [Accessed 31 Outubro 2019].

[9] "Cabify," [Online]. Available: https://cabify.com/pt. [Accessed 31 Outubro 2019].

[10] "Lyft," [Online]. Available: https://www.lyft.com/. [Accessed 31 Outubro 2019].

[11] "DriveNow," [Online]. Available: https://www.drive-now.com/pt/pt/lisbon. [Accessed 31 Outubro 2019].

[12] "GetAround," [Online]. Available: https://www.getaround.com/. [Accessed 31 Outubro 2019].

[13] "Car2Go," [Online]. Available: https://www.car2go.com/US/en/. [Accessed 31 Outubro 2019].

[14] "Lime," [Online]. Available: https://www.li.me/pt/. [Accessed 31 Outubro 2019].

[15] "Circ," [Online]. Available: https://play.google.com/store/apps/details?id=com.goflash.consumer&hl=pt_PT. [Accessed 31 Outubro 2019].

[16] "eCoolTra," [Online]. Available: https://www.ecooltra.com/pt/. [Accessed 31 Outubro 2019].

[17] Stigo, "The Last Mile — the term, the problem and the odd solutions," 4 Outubro 2017. [Online]. Available: https://medium.com/the-stigo-blog/the-last-mile-the-term-the-problem-and-the-odd-solutions-28b6969d5af8. [Accessed 21 Julho 2020].

[18] Cordeau, J. F., Laporte, G., Potvin, J. Y., & Savelsbergh, M. W. (2007). Transportation on demand. *Handbooks in operations research and management science*, *14*, 429-466.

[19] "An Introduction to Demand Responsive Transport," [Online]. Available: https://mobility.here.com/learn/smart-mobility/introduction-demand-responsive-transport. [Accessed 23 Outubro 2019].

[20] I. Europe, "Demand-responsive transport," Junho 2018. [Online]. Available: https://www.interregeurope.eu/fileadmin/user_upload/plp_uploads/policy_briefs/2018-06-27_Policy_Brief_Demand_Responsive_Transport.pdf. [Accessed 23 Outubro 2019].

[21] F. A. C. Proença, "Bus On Demand," [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/395142123798/dissertacao.pdf. [Accessed 23 Outubro 2019].

[22] "Flor da ria," [Online]. Available: https://transportesflordar.wixsite.com/flordaria-embreve. [Accessed 23 Outubro 2019].

[23] G. R. Mauri and L. A. N. Lorena, "Uma nova abordagem para o problema dial-a-ride," Abril 2009. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-65132009000100004. [Accessed 23 Outubro 2019].

[24] "Vehicle routing problem," [Online]. Available: https://developers.google.com/optimization/routing/vrp. [Accessed 23 Outubro 2019].

[25] Weise, T., Podlich, A., & Gorldt, C. (2009). Solving real-world vehicle routing problems with evolutionary algorithms. In Natural intelligence for scheduling, planning and packing problems (pp. 29-53). Springer, Berlin, Heidelberg.

[26] "How to effectively solve the vehicle routing problem?," 9 Agosto 2017. [Online]. Available: https://www.abivin.com/single-post/2017/08/07/How-To-Effectively-Solve-The-Vehicle-Routing-Problem. [Accessed 23 Outubro 2019].

[27] "High Occupancy Vehicle (HOV) Lanes," [Online]. Available: http://www.its.leeds.ac.uk/projects/konsult/private/level2/instruments/instrument029/l2_029summ.htm. [Accessed 17 Abril 2020].

[28] "High-Occupancy Toll Lanes (Partial Facility Princing)," [Online]. Available: https://ops.fhwa.dot.gov/congestionpricing/strategies/involving_tolls/hot_lanes.htm. [Accessed 17 Abril 2020].

[29] "About Slugging," [Online]. Available: http://slug-lines.com/Slugging/About_slugging.asp. [Accessed 17 Abril 2020].

[30] "Clustering in machine learning," [Online]. Available: https://www.geeksforgeeks.org/clustering-in-machine-learning/. [Accessed 17 Abril 2020].

[31] "O que é a análise de cluster?," [Online]. Available: https://operdata.com.br/blog/analise-de-cluster/. [Accessed 17 Abril 2020].

[32] J. P. F. Simões, "Repositório comum - Análise de dados e Machine Learning na Mobilidade Urbana," Abril 2019. [Online]. Available: https://comum.rcaap.pt/bitstream/10400.26/29858/1/Joao-Pedro-Fernandes-simoes.pdf. [Accessed 30 Dezembro 2019].

[33] Sisodia, D., Singh, L., Sisodia, S., & Saxena, K. (2012). Clustering techniques: a brief survey of different clustering algorithms. *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, *1*(3), 82-87.

[34] J. P. d. S. Oliveira, "Classificação de Literatura Biomédica," Dezembro 2014. [Online]. Available: http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/Teses/Tese_Mest_Joao-Santos-Oliveira.pdf. [Accessed 30 Dezembro 2019].

[35] "Minimum Cost Flows," 1 Outubro 2018. [Online]. Available: https://developers.google.com/optimization/flow/mincostflow. [Accessed 18 Julho 2020].

[36] J. Levy-Kramer, "k-means-constrained," [Online]. Available: https://pypi.org/project/k-means-constrained/. [Accessed 18 Julho 2020].

[37] K. B. A. P.S.Bradley, "Constrained K-Means Clustering," Maio 2000. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2000-65.pdf. [Accessed 18 Julho 2020].

[38] Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., ... & Lin, C. T. (2017). A review of clustering techniques and developments. *Neurocomputing*, *267*, 664-681.

[39] R. Chauhan, "Clustering Techniques: A Comprehensive Study of Various Clustering Techniques," *International Journal of Advanced Research in Computer Science*, vol. 5, no. 5, 2014.

[40] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, No. 34, pp. 226-231).

[41] C. Maklin, "DBSCAN Python Example: The Optimal Value For Epsilon (EPS)," 30 Junho 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc. [Accessed 30 Dezembro 2019].

[42] Khan, M. M. R., Siddique, M. A. B., Arif, R. B., & Oishe, M. R. (2018, September). ADBSCAN: Adaptive density-based spatial clustering of applications with noise for identifying clusters with varying densities. In *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)* (pp. 107-111). IEEE.

[43] S. Agrawal, "Machine Learning - DBSCAN," 29 Maio 2013. [Online]. Available: https://algorithmicthoughts.wordpress.com/2013/05/29/machine-learning-dbscan/. [Accessed 30 Dezembro 2019].

[44] "Clustering Algorithms: K-means," [Online]. Available: https://www.cs.princeton.edu/courses/archive/spring08/cos435/Class_notes/clustering2_toPost.pdf. [Accessed 30 Dezembro 2019].

[45] D. Dey, "ML | Hierarchical clustering (Agglomerative and Divisive clustering)," 9 Maio 2019. [Online]. Available: https://www.geeksforgeeks.org/ml-hierarchical-clustering-agglomerative-and-divisive-clustering/. [Accessed 7 Setembro 2020].

[46] C. M. W. Guojun Gan, "12. Grid-Based Clustering Algorithms," in *Data clustering: theory, algorithms, and applications.*, Society for Industrial and Applied Mathematics, 2007.

[47] Von Luxburg, U. (2007). A tutorial on spectral clustering. Statistics and computing, 17(4), 395-416.

[48] N. Doshi, "Spectral clustering," 4 Fevereiro 2019. [Online]. Available: https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7. [Accessed 7 Setembro 2020].

[49] Klawonn, F., & Höppner, F. (2003, August). What is fuzzy about fuzzy clustering? Understanding and improving the concept of the fuzzifier. In *International symposium on intelligent data analysis* (pp. 254-264). Springer, Berlin, Heidelberg.

[50] A. &. C. E. Bhaskar, "Passenger Segmentation Using Smart Card Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1537-1548, Junho 2014.

[51] M. T. B. A. Catherine Morency, "Analysing the variability of transit users behaviour with smart card data," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 44-49, 2006.

[52] Y.-c. T. H.-w. C. J. Y.-j. H. Han-wen Chang, "iTaxi: Context-Aware Taxi Demand Hotspots Prediction Using Ontology and Data Mining Approaches," *International Journal of Business Intelligence and Data Mining*, 2009.

[53] "GEOINFO," [Online]. Available: http://www.geoinfo.info/geoinfo2020/. [Accessed 16 Julho 2020].

[54] Andrade, T. C., de Arruda Pereira, M., & Wanner, E. F. (2014). Development of an Application Using a Clustering Algorithm for Definition of Collective Transportation Routes and Times. In *GeoInfo* (pp. 13-24).

[55] Tang, J. (2019). Urban Travel Mobility Exploring With Large-Scale Trajectory Data. In *Data-Driven Solutions to Transportation Problems* (pp. 137-174). Elsevier.

[56] "Whim," [Online]. Available: https://whimapp.com. [Accessed 30 Novembro 2019].

[57] "Moovel," [Online]. Available: https://www.moovel.com/en. [Accessed 30 Novembro 2019].

[58] "By app from A to B. moovel simplifies urban mobility," [Online]. Available: https://www.daimler.com/products/services/mobility-services/moovel/. [Accessed 30 Novembro 2019].

[59] "Moovit MaaS Solution," [Online]. Available: https://company.moovit.com/maas-solutions/. [Accessed 30 Novembro 2019].

[60] "Google Transit," [Online]. Available: https://support.google.com/transitpartners/answer/1111471?hl=en. [Accessed 30 Novembro 2019].

[61] "Transit," [Online]. Available: https://transitapp.com/. [Accessed 30 Novembro 2019].

[62] "Transit (app)," [Online]. Available: https://en.wikipedia.org/wiki/Transit_(app). [Accessed 30 Novembro 2019].

[63] "Citymapper," [Online]. Available: https://en.wikipedia.org/wiki/Citymapper. [Accessed 30 Novembro 2019].

[64] "Citymapper," [Online]. Available: https://www.techtudo.com.br/tudo-sobre/citymapper.html. [Accessed 30 Novembro 2019].

[65] "Blablacar," [Online]. Available: https://www.blablacar.pt/. [Accessed 30 Novembro 2019].

[66] "FluidHub," [Online]. Available: https://www.fluidtime.com/en/. [Accessed 30 Novembro 2019].

[67] "Metodologia Ágil," [Online]. Available: http://metodologiaagil.com/. [Accessed 5 Junho 2020].

[68] "8 motivos para aprender a programar em Python," [Online]. Available: https://www.somosicev.com/blogs/8-motivos-para-aprender-programar-em-python/. [Accessed 27 Julho 2020].

[69] Keyul, "10 Most Popular Web Frameworks in 2020," 27 Novembro 2019. [Online]. Available: https://medium.com/front-end-weekly/10-most-popular-web-frameworks-in-2020-167b9103e08a. [Accessed 29 Abril 2020].

[70] S. Hansen, "Advanges and Disadvantages of Django," 23 Maio 2017. [Online]. Available: https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5. [Accessed 14 Janeiro 2020].

[71] "Top 7 Backend Web Frameworks to use in 2019," 16 Outubro 2018. [Online]. Available: https://teqnation.com/top-7-backend-web-frameworks-to-use-in-2019/?cn-reloaded=1. [Accessed 14 Janeiro 2020].

[72] "DB-Engines Ranking," [Online]. Available: https://db-engines.com/en/ranking. [Accessed 19 Novembro 2019].

[73] "2019 Open Source Database," [Online]. Available: https://dzone.com/articles/2019-open-source-database-report-top-databases-pub. [Accessed 19 Novembro 2019].

[74] "DB-Engines," 21 Novembro 2019. [Online]. Available: https://db-engines.com/en/ranking_definition. [Accessed 25 Novembro 2019].

[75] "DB-Engines Ranking - Trend Popularity," [Online]. Available: https://db-engines.com/en/ranking_trend. [Accessed 21 Novembro 2019].

[76] "Django - Migrations," [Online]. Available: https://docs.djangoproject.com/en/2.2/topics/migrations/. [Accessed 23 Novembro 2019].

[77] "Integrating Django with MongoDB," [Online]. Available: https://nesdis.github.io/djongo/integrating-django-with-mongodb/. [Accessed 23 Novembro 2019].

[78] "Nominatim," [Online]. Available: https://nominatim.org/. [Accessed 29 Junho 2020].

[79] "Django REST Framework," [Online]. Available: https://www.django-rest-framework.org/. [Accessed 11 Junho 2020].

[80] P. Victória, "Qual a melhor linguagem para ciência de dados?," 6 Agosto 2019. [Online]. Available: https://blog.geekhunter.com.br/guia-qual-a-melhor-linguagem-para-ciencia-de-dados/. [Accessed 29 Abril 2020].

[81] "OpenCage Geocoder," [Online]. Available: https://opencagedata.com/. [Accessed 9 Junho 2020].

[82] A. Medeiros, "7 Motivos para Usar o PostGIS em seus Projetos," 6 Fevereiro 2013. [Online]. Available: http://www.clickgeo.com.br/motivos-para-usar-postgis/. [Accessed 9 Junho 2020].

[83] "Pandas," [Online]. Available: https://pandas.pydata.org/. [Accessed 29 Abril 2020].

[84] "What's the future of the pandas library?," 12 Dezembro 2018. [Online]. Available: https://www.dataschool.io/future-of-pandas/. [Accessed 29 Abril 2020].

[85] S. Pal, "Scikit-learn Tutorial: Machine Learning in Python," [Online]. Available: https://www.dataquest.io/blog/sci-kit-learn-tutorial/. [Accessed 29 Abril 2020].

[86] "Scikit-learn," [Online]. Available: https://scikit-learn.org/. [Accessed 29 Abril 2020].

[87] N. d. S. Martins, "RabbitMQ: o que é e como utilizar," 29 Março 2018. [Online]. Available: https://blog.cedrotech.com/rabbitmq-o-que-e-e-como-utilizar/. [Accessed 9 Junho 2020].

[88] "Celery," [Online]. Available: https://www.fullstackpython.com/celery.html. [Accessed 9 Junho 2020].

[89] "psycopg2 - Python-PostgreSQL Database Adapter," [Online]. Available: https://github.com/psycopg/psycopg2/. [Accessed 9 Junho 2020].

[90] J. Chathuranga, "Leaving Google Maps? Use OSRM for Routing," 23 Setembro 2018. [Online]. Available: https://medium.com/@janakachathuranga/leaving-google-maps-use-osrm-for-routing-aa1afc912df3. [Accessed 16 Junho 2020].

[91] "OSRM API Documentation," [Online]. Available: http://project-osrm.org/docs/v5.22.0/api/#general-options. [Accessed 16 Junho 2020].

[92] "GitLab," [Online]. Available: https://about.gitlab.com/. [Accessed 14 Janeiro 2020].

[93] "Jupyter," [Online]. Available: https://jupyter.org/. [Accessed 10 Setembro 2020].

[94] "PyCharm," [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed 10 Setembro 2020].

[95] A. Kiffer, "Iniciando com o docker: dicas práticas para começar a usar agora mesmo," 19 Abril 2016. [Online]. Available: https://tableless.com.br/iniciando-com-o-docker-dicas-praticas-para-comecar-usar-agora-mesmo/. [Accessed 29 Abril 2020].

[96] W. Meller, "Scrum: O que são User Stories," 16 Novembro 2016. [Online]. Available: https://sitecampus.com.br/scrum-o-que-sao-user-stories/. [Accessed 8 Junho 2020].

[97] C. Maklin, "DBSCAN Python Example: The Optimal Value For Epsilon (EPS)," 30 Junho 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-clustering-

dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc. [Accessed 29 Junho 2020].

[98] Rahmah, N., & Sitanggang, I. S. (2016, January). Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP Conference Series: Earth and Environmental Science* (Vol. 31, No. 1, p. 012012). IOP Publishing.

[99] E. Lutins, "DBSCAN: What is it? When to Use it? How to use it.," 6 Setembro 2017. [Online]. Available: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818. [Accessed 29 Junho 2020].

[100] "The Ninth International Conference on Intelligent Systems and Applications," [Online]. Available: https://www.iaria.org/conferences2020/SubmitINTELLI20.html. [Accessed 17 Junho 2020].

Anexos

Anexo A – Especificação do levantamento de requisitos

ID do caso de uso:	1		
Nome do caso de uso:	**Efetuar login**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor, Passageiro		
Descrição:	Neste caso de uso, o condutor e o passageiro têm de se autenticar para ter acesso às diversas funcionalidades em conformidade com as permissões que lhe foram concedidas, sendo que para isso terão que inserir um *username* e *password*.		
Pré-condições:	1. O utilizador tem de ter acesso à internet; 2. Estar registado na aplicação		

Pós-condições:	1. Ficar autenticado na aplicação com as permissões inerentes ao tipo de utilizador registado na aplicação.
Fluxo principal:	**1.0 Efetuar login** 1. O Utilizador acede ao sistema sem autenticação ativa; 2. O sistema apresenta a página de login; 3. O Utilizador introduz o seu email e password no sistema; 4. O utilizador carrega no botão “Login”; 5. O sistema envia um comando (“Efetuar Login”) via web; 6. O sistema compara as credenciais enviadas com as que estão guardadas na base de dados; 7. A aplicação atribui as permissões consoante o tipo de utilizador que está registado na base de dados; 8. O sistema mostra mensagem de sucesso; 9. O sistema redireciona para o menu inicial; 10. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	**1.1 O Utilizador introduz o username e/ou a password incorretamente** 1. O sistema apresenta uma mensagem de erro; 2. O caso de uso termina sem sucesso 3. O utilizador repete o passo 2.

Excepções:	**1.0 E.1 O Utilizador não tem acesso à internet** 1. O Sistema mostra uma mensagem de erro a informar que não tem internet. 2. O caso de uso termina sem sucesso. **1.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 1. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 1; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez, segurança
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	2		
Nome do caso de uso:	**Recuperar password**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor, Passageiro		
Descrição:	Neste caso de uso, o condutor e o passageiro caso se tenham esquecido da *password*, devem inserir o seu *email*. A partir de um link que é enviado para o email que foi associado, o utilizador pode introduzir a palavra passe que pretende, voltando a ter acesso ao conteúdo que tinha, dependendo das permissões que tem associadas.		
Pré-condições:	1. O utilizador tem de ter acesso à internet; 2. Estar registado na aplicação		
Pós-condições:	1. Ter uma nova *password* de forma a ter acesso novamente à aplicação.		

Fluxo principal:	**2.0 Recuperar password**
	
	1. O utilizador acede ao sistema sem autenticação ativa;
	2. O sistema apresenta a página de login;
	3. O utilizador clica no link *"Recover here"* para recuperar a password;
	4. O sistema apresenta página de recuperação de password;
	5. O utilizador introduz o seu *email* no sistema;
	6. O utilizador carrega no botão "*Submit"*;
	7. O sistema reencaminha um email com um link para introduzir a nova *password* para o email do utilizador;
	8. O utilizador acede ao link enviado pelo sistema, sendo redirecionado para a página de mudança de *password*;
	    1. O utilizador introduz uma nova *password*;
	    2. O utilizador introduz novamente a *password;*
	9. O utilizador carrega no botão "*Change password";*
	10. O sistema valida as *passwords* introduzidas anteriormente pelo utilizador;
	11. O sistema mostra mensagem de sucesso;
	12. O caso de uso termina com sucesso.

Fluxo(s) alternativo(s):	**7.1 O utilizador após ter alterado a password voltar a clicar novamente no link enviado por email para o mesmo** 1. O sistema apresenta mensagem de erro a informar que o link é inválido porque possivelmente já foi usado; 2. O utilizador repete o passo 3. 3. O caso de uso termina com sucesso. **10.1 O utilizador falhar ao introduzir alguma das *passwords*** 1. O sistema mostra mensagem de erro. 2. O utilizador repete o passo 8. 3. O caso de uso termina com sucesso.

Excepções:	**2.0 E.1 O Utilizador não tem acesso à internet** 3. O Sistema mostra uma mensagem de erro a informar que não tem internet. 4. O caso de uso termina sem sucesso. **2.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 2. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 1; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez, segurança
Pressupostos:	2. Assume-se que o utilizador possui internet;

ID do caso de uso:	3		
Nome do caso de uso:	**Visualizar perfil**		
Criado por:	Carlos Afonso	Last Updated B y:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro, Condutor		
Descrição:	Este caso de uso permite ao condutor e ao passageiro visualizar os dados do seu perfil.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:			

Fluxo principal	**3.0 Visualizar o perfil** 1. O utilizador seleciona a opção de visualizar o perfil; 2. O sistema apresenta os dados ao utilizador autenticado; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	
Excepções:	**3.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	

Pontos de extensão:	
Prioridade:	Média
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	4		
Nome do caso de uso:	**Editar perfil**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro, Condutor		
Descrição:	Este caso de uso permite ao utilizador autenticado (condutor e passageiro) editar os dados do seu perfil.		

Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;
Pós-condições:	
Fluxo principal	**4.0 Editar o perfil** 1. O utilizador seleciona a opção de visualizar o perfil; 2. O sistema apresenta os dados ao utilizador autenticado; 3. O utilizador carrega no botão “Edit profile” para editar os campos que pretende (“*first name”*,*”last name”*, *“email”*) 4. O sistema valida os dados e atualiza a informação na base de dados. 5. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	

Excepções:	**4.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso. **4.0 E.2 O sistema não conseguir atualizar a informação na base de dados** 1.a) O Sistema mostra uma mensagem de erro. 1.b) O caso de uso termina sem sucesso.
Inclusões:	
Pontos de extensão:	
Prioridade:	Média
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	5		
Nome do caso de uso:	**Efetuar pedido de transporte (via web app e autenticado)**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro		
Descrição:	O passageiro pode efetuar o pedido de um transporte(via web app), sendo o mesmo registado no servidor de base de dados.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:	O pedido de transporte fica agendado.		

Fluxo principal	**5.0 Efetuar pedido de transporte**
	
	1. O Passageiro seleciona o menu “Efetuar pedido de transporte”;
	2. O Passageiro preenche o formulário:
	  1. O Passageiro introduz o local de partida;
	  2. O Passageiro introduz o local de destino;
	  3. O Passageiro introduz a data do pedido;
	3. O Passageiro carrega no botão “Criar pedido de transporte” para enviar pedido de transporte;
	4. O sistema envia um comando (“Efetuar Pedido de transporte”) via web;
	5. O Passageiro recebe confirmação do registo do pedido
	6. O Caso de uso termina com sucesso.

Fluxo(s) alternativo(s):	**4.1 O sistema falha ao enviar o comando “Efetuar pedido de transporte”, apresentando uma mensagem de erro** 4.1.1 O sistema indica mensagem de erro 4.1.2 Volta ao passo 2 4.1.3 O caso de uso termina sem sucesso;

Excepções:	**5.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso. **5.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 1. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 2; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	6		
Nome do caso de uso:	**Listar pedidos de transporte**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro		
Descrição:	O utilizador autenticado(passageiro) pode visualizar os pedidos de transporte que efetuou.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:			

Fluxo principal	**6.0 Listar pedidos de transporte** 1. O utilizador autenticado(passageiro) seleciona o menu “View orders”; 2. O sistema apresenta os pedidos de transporte que o utilizador efetuou; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	
Excepções:	**6.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	

Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	7		
Nome do caso de uso:	**Tornar-se condutor**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro		
Descrição:	O utilizador autenticado(passageiro) pode tornar-se um condutor se pretender.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		

Pós-condições:	
Fluxo principal	**7.0 Tornar-se condutor** 1. O utilizador seleciona o menu "*Became also a driver*"; 2. O sistema apresenta no ecrã a página para o passageiro se tornar um condutor; 2.1. O utilizador introduz o NIF; 2.2 O utilizador introduz o número da licença; 2.3. O utilizador carrega no botão "*Became also a driver*" para se tornar um condutor; 3. O sistema envia um comando ("*Became also a driver*") via web; 4. O utilizador recebe confirmação de que se tornou um condutor; 5. O caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	**2.1. Quando o sistema mostrar no ecrã a página para o passageiro se tornar um condutor, o mesmo já o ser.** 2.1.1. O sistema mostra mensagem de erro a dizer que o utilizador já é um condutor; 2.1.2. O caso de uso termina sem sucesso.

Excepções:	**7.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	8		
Nome do caso de uso:	**Registar ofertas de boleia**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor		
Descrição:	O condutor pode efetuar o registo de ofertas de boleia(via web app), sendo o mesmo registado no servidor de base de dados.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:	A oferta de boleia fica agendada.		

Fluxo principal	**8.0 Registar ofertas de boleia**
	
	1. O condutor seleciona o menu "*Create offer ride*";
	2. O condutor preenche o formulário:
	    1. O condutor introduz o local de partida;
	    2. O condutor introduz o local de destino;
	    3. O condutor introduz a data de partida;
	    4. O condutor introduz a data de chegada;
	3. O condutor carrega no botão "*Create offer ride*" para enviar oferta de boleia;
	4. O sistema envia um comando ("Criar oferta de boleia") via web;
	5. O condutor recebe confirmação do registo da oferta de boleia;
	6. O Caso de uso termina com sucesso.

Fluxo(s) alternativo(s):	**4.1 O sistema falha ao enviar o comando "Criar oferta de boleia"** 4.1.1 O sistema indica mensagem de erro 4.1.2 Volta ao passo 2 4.1.3 O caso de uso termina sem sucesso;

Excepções:	**8.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso. **8.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 1. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 2; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	9		
Nome do caso de uso:	**Listar ofertas de boleia**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor		
Descrição:	O utilizador autenticado(condutor) pode visualizar as suas ofertas de boleia.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:			

Fluxo principal	**9.0 Listar ofertas de boleia** 1. O utilizador autenticado(condutor) seleciona o menu “*List of offer rides*”; 2. O sistema apresenta as ofertas de boleia que o utilizador efetuou; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	
Excepções:	**9.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	

Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	10		
Nome do caso de uso:	**Registar veículos**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor		
Descrição:	O condutor pode registar os seus veículos (via web app), sendo o mesmo registado no servidor de base de dados.		

Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;
Pós-condições:	Os veículos ficam registados.
Fluxo principal	**10.0 Registar veículos** 1. O condutor seleciona o menu "*Create vehicles*"; 2. O condutor preenche o formulário: 1. O condutor introduz a marca do veículo; 2. O condutor introduz o modelo do veículo; 3. O condutor introduz a matrícula do veículo; 4. O condutor introduz a capacidade do veículo; 5. O condutor introduz o tipo de veículo; 3. O condutor carrega no botão "*Create new vehicle*" para enviar o registo do veículo; 4. O sistema envia um comando ("Criar registo de veículo") via web; 5. O condutor recebe confirmação do registo do veículo; 6. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	**4.1 O sistema falha ao enviar o comando "Criar registo de veículo"** 4.1.1 O sistema indica mensagem de erro 4.1.2. Volta ao passo 2 4.1.3 O caso de uso termina sem sucesso;

Excepções:	**10.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso. **10.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 1. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 2; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	11		
Nome do caso de uso:	**Listar veículos**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor		
Descrição:	O utilizador autenticado(condutor) pode visualizar os veículos que tem associado à sua conta.		
Pré-condições:	1. O utilizador tem de ter acesso à internet; 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:			

Fluxo principal	**11.0 Listar veículos** 1. O utilizador autenticado(condutor) seleciona o menu “*List of vehicles*”; 2. O sistema apresenta os veículos que o utilizador registou; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	
Excepções:	**11.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	

Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	12		
Nome do caso de uso:	**Visualizar estado das viagens dos pedidos**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Passageiro		
Descrição:	O utilizador autenticado(passageiro) pode visualizar o estado das viagens associados aos seus pedidos.		

Pré-condições:	1. O utilizador tem de ter acesso à internet; 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;
Pós-condições:	
Fluxo principal	**12.0 Visualizar estado das viagens dos pedidos** 1. O utilizador autenticado(passageiro) seleciona o menu "*View Orders*"; 2. O sistema apresenta a informação com o estado das viagens associado a cada um dos pedidos; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	

Excepções:	**12.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	13		
Nome do caso de uso:	**Visualizar estado das viagens das ofertas de boleia**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio, 2020
Ator(s):	Condutor		
Descrição:	O utilizador autenticado(condutor) pode visualizar o estado das viagens associado às ofertas de boleia.		
Pré-condições:	1. O utilizador tem de ter acesso à internet 2. O utilizador deve estar registado na aplicação; 3. O Utilizador deve estar autenticado no sistema;		
Pós-condições:			

Fluxo principal	**13.0 Visualizar estados das viagens das ofertas de boleia** 1. O utilizador autenticado(condutor) seleciona o menu "*List of offer rides*"; 2. O sistema apresenta o estado das viagens associado às ofertas de boleia que efetuou; 3. O Caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	
Excepções:	**13.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso.
Inclusões:	

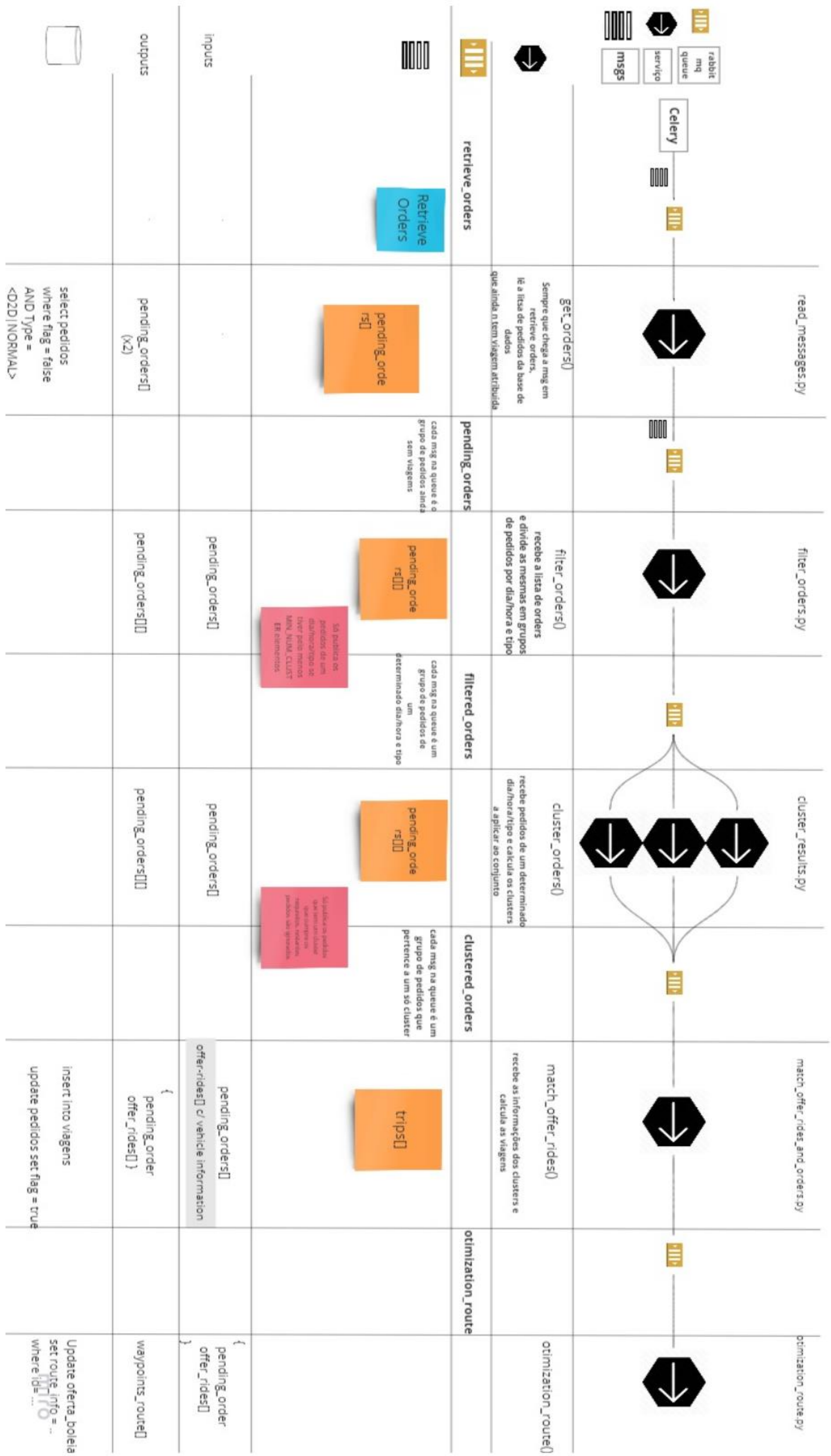
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

ID do caso de uso:	14		
Nome do caso de uso:	**Efetuar logout**		
Criado por:	Carlos Afonso	Last Updated By:	Carlos Afonso
Data de criação:	1 de Fevereiro,2020	Date Last Updated:	6 de Maio,2020
Ator(s):	Passageiro, Condutor		

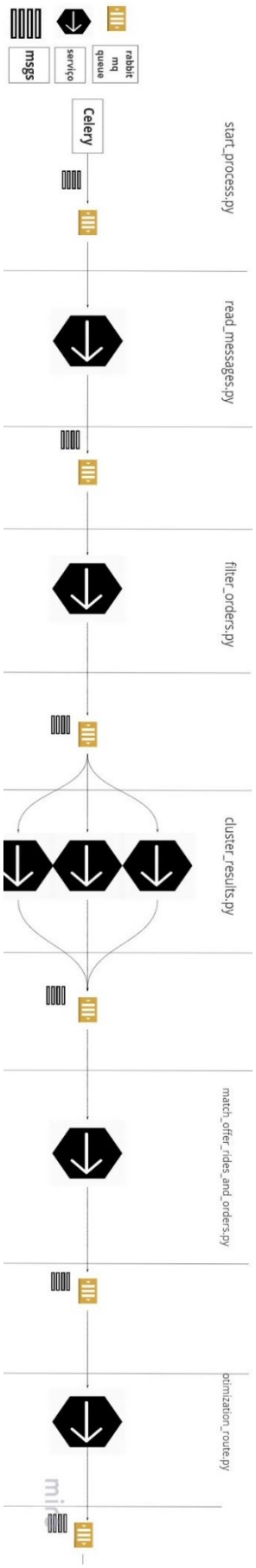
Descrição:	O utilizador (passageiro e condutor) efetuam o *logout* não tendo assim mais acesso a qualquer tipo de funcionalidade do sistema.
Pré-condições:	1. O utilizador tem de ter acesso à internet; 2. O utilizador tem que estar autenticado no sistema;
Pós-condições:	1. O Utilizador termina sessão na aplicação,
Fluxo principal:	**14.0 Efetuar logout** 1. O utilizador efetua o "*Logout*", saindo assim da aplicação; 2. O sistema envia um comando ("Efetuar Logout") via web; 3. O caso de uso termina com sucesso.
Fluxo(s) alternativo(s):	

Excepções:	**14.0 E.1 O Utilizador não tem acesso à internet** 1.a) O Sistema mostra uma mensagem de erro a informar que não tem internet. 1.b) O caso de uso termina sem sucesso. **14.0 E.2 Ocorreu um erro do sistema ao enviar o comando** 1. O Sistema apresenta uma mensagem informando que ocorreu um erro do Sistema ao enviar o comando 2. a) O Utilizador retorna ao ponto 1; 2. b) O caso de uso termina.
Inclusões:	
Pontos de extensão:	
Prioridade:	Alta
Requisitos especiais:	Usabilidade, eficiência, rapidez
Pressupostos:	1. Assume-se que o utilizador possui internet;

Anexo B – Diagrama de Workflow do módulo de clustering e otimização de rota


rabbit mq queue
serviço
msgs
Celery
read_messages.py
filter_orders.py
cluster_results.py
match_offer_rides_and_orders.py
otimization_route.py
retrieve_orders
pending_orders
filtered_orders
clustered_orders
otimization_route
get_orders()
filter_orders()
cluster_orders()
match_offer_rides()
otimization_route()
Retrieve Orders
trips[]
inputs
outputs
pending_orders[]
pending_orders[][]
waypoints_route[]
insert into viagens
update pedidos set flag = true
select pedidos where flag = false AND Type = <D2D | NORMAL>

Anexo C – Diagrama da pipeline de processamento


msgs
serviço
rabbit mq queue
Celery
start_process.py
read_messages.py
filter_orders.py
cluster_results.py
match_offer_rides_and_orders.py
otimization_route.py

Anexo D – Screenshots do backoffice

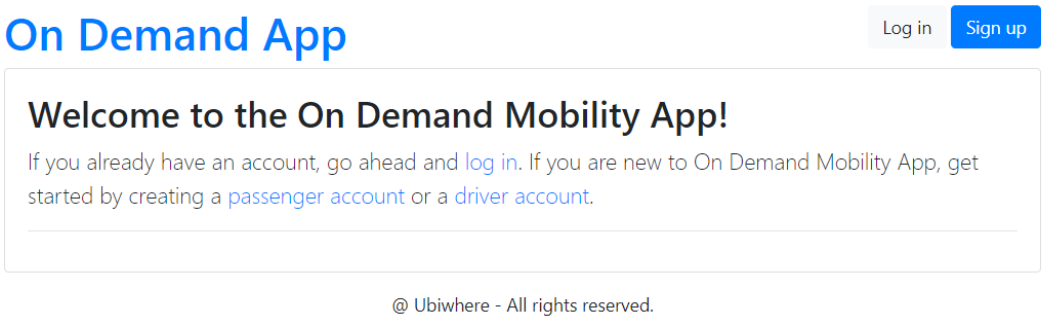


Figura 34 - Página inicial da aplicação

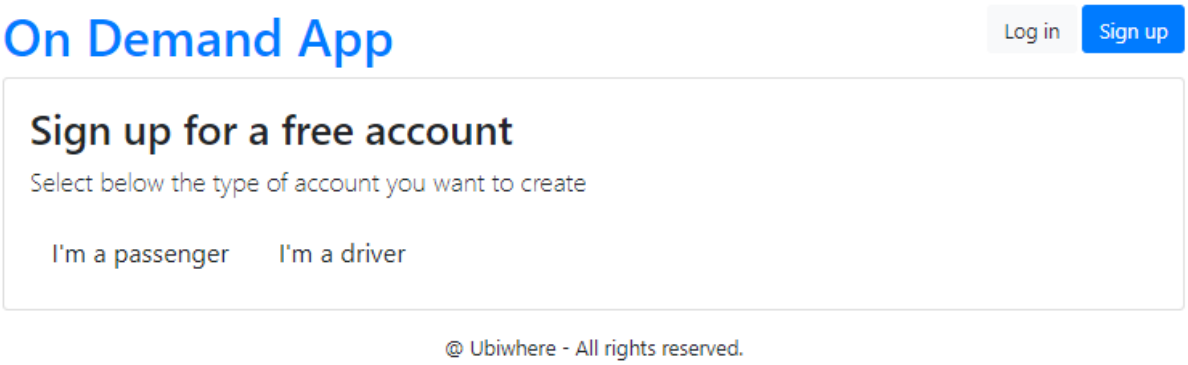


Figura 35 - Selecionar tipo de utilizador para o registo na aplicação.

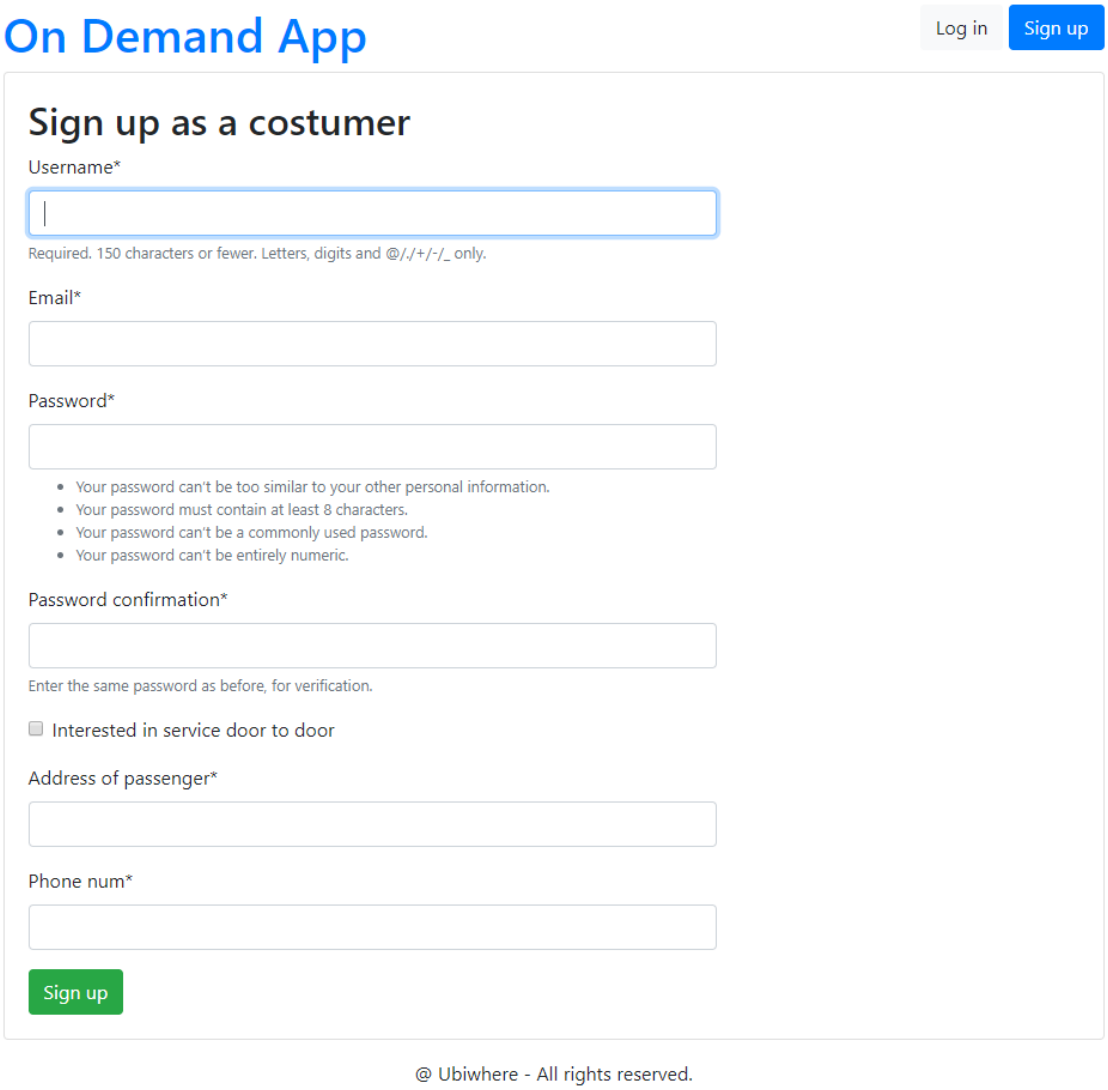


Figura 36 - Formulário de registo do passageiro

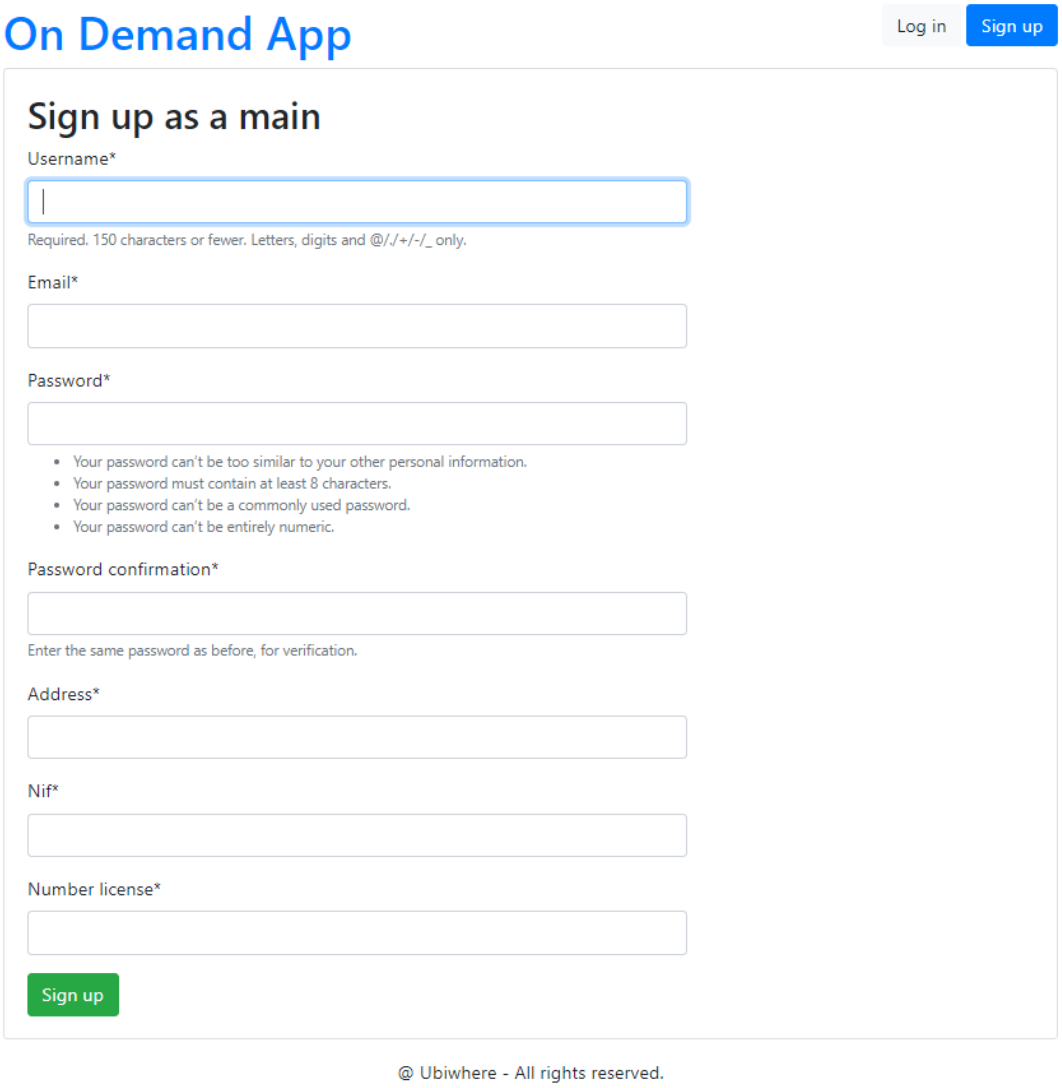


Figura 37 - Formulário de registo do condutor

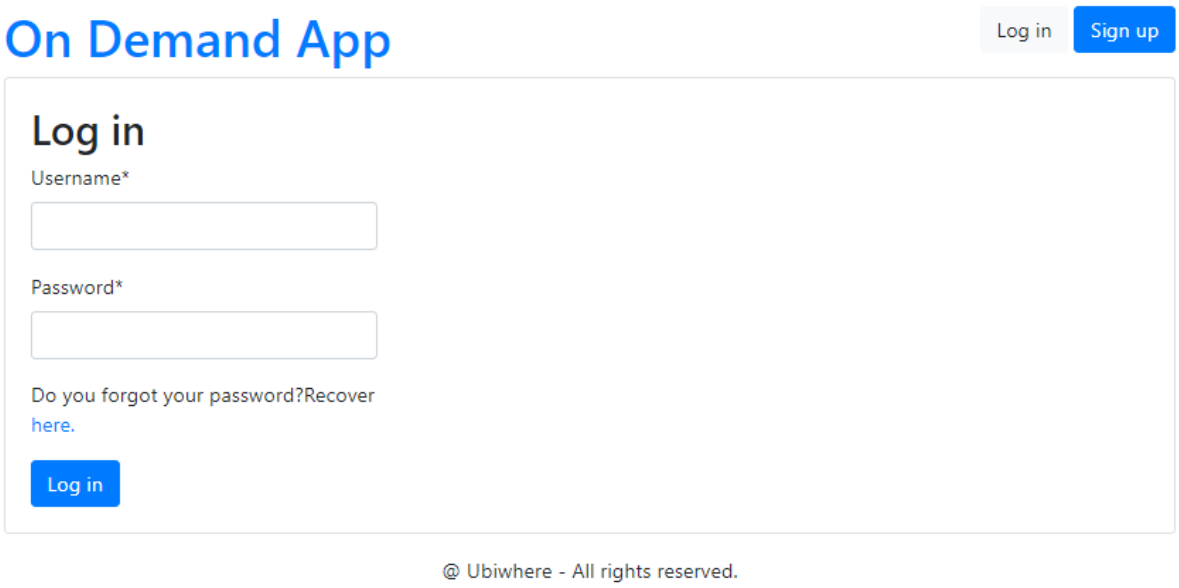


Figura 38 - Página de autenticação da aplicação.

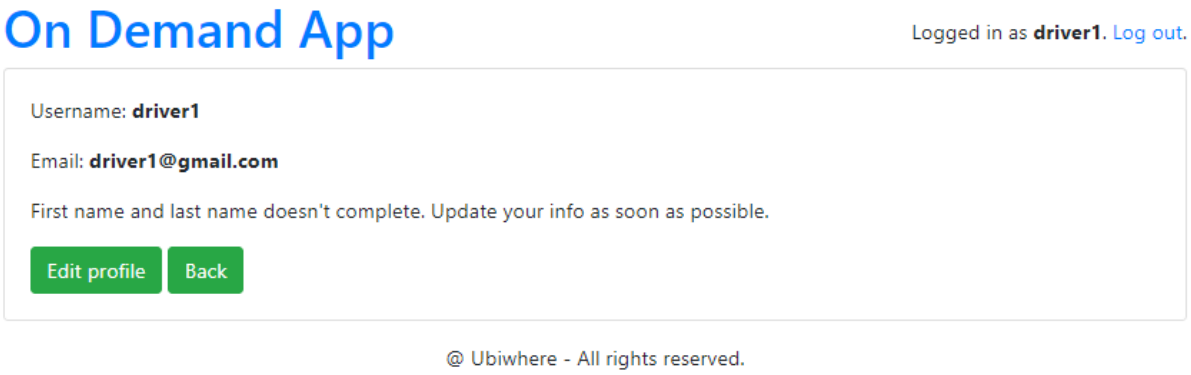


Figura 39 - Visualizar perfil

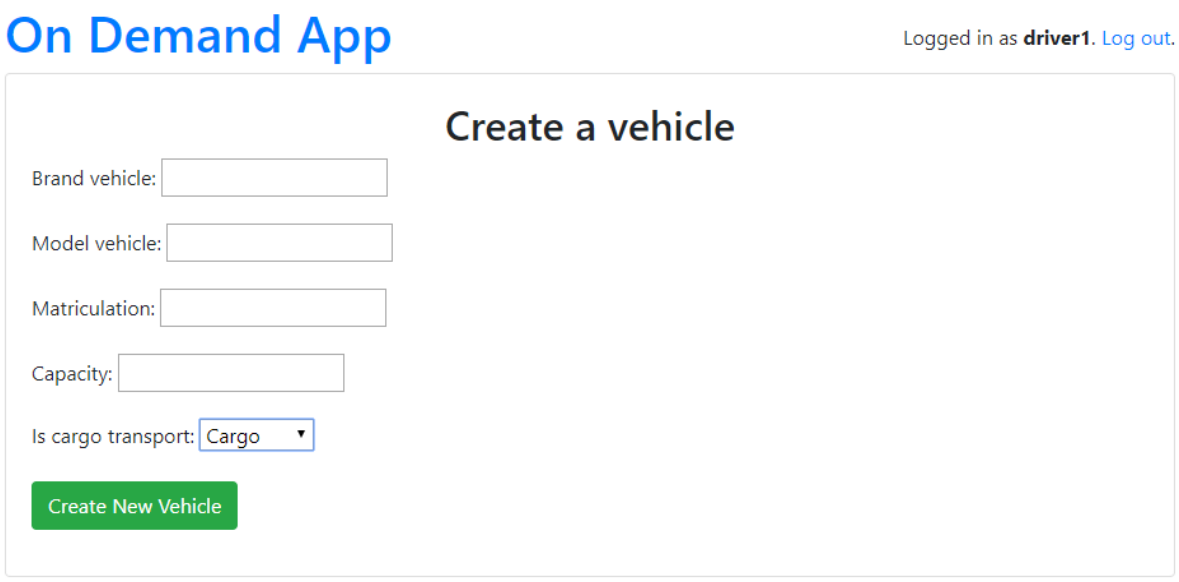


Figura 40 - Registo de veículos por parte do condutor.

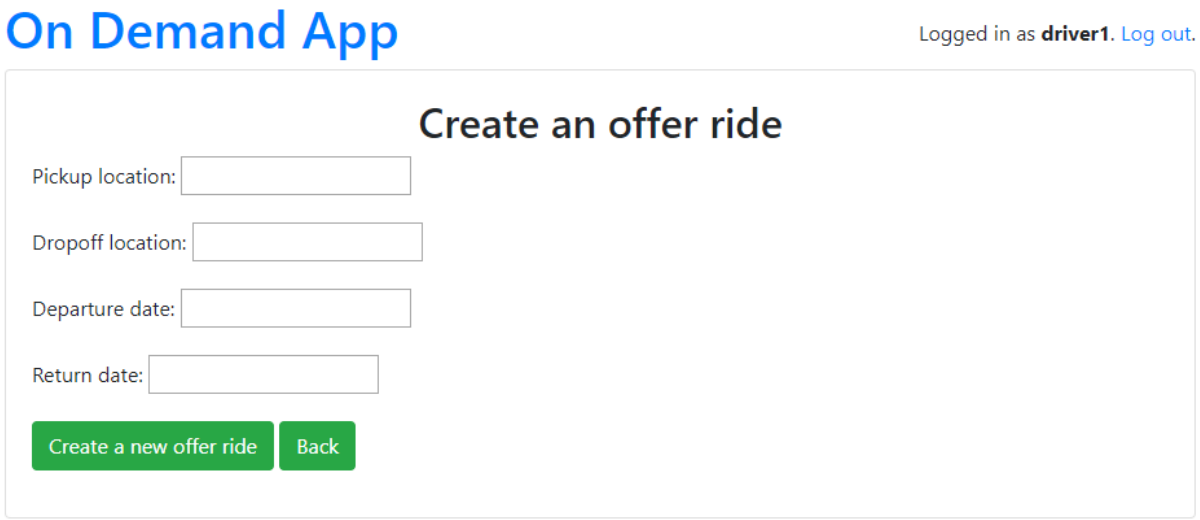


Figura 41-Registo de ofertas de boleia por parte do condutor.

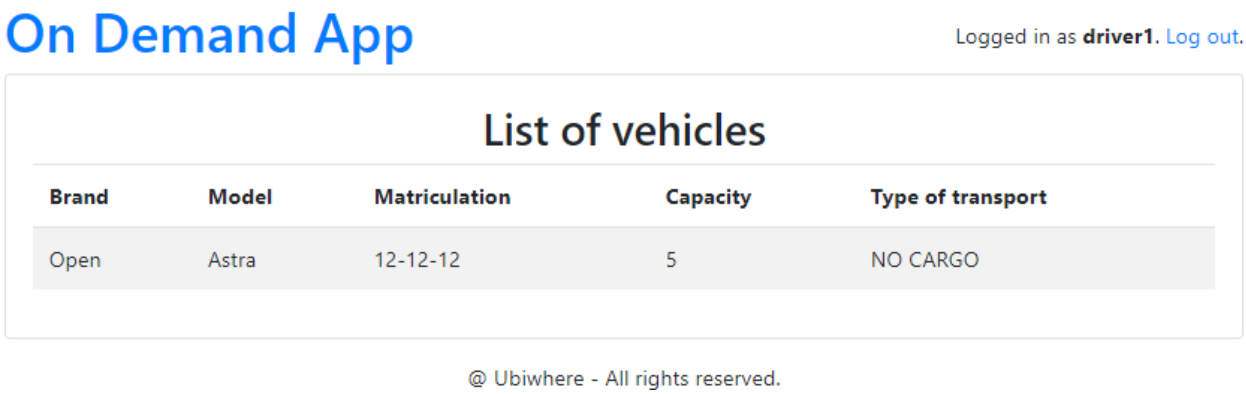


Figura 42 - Lista de veículos.

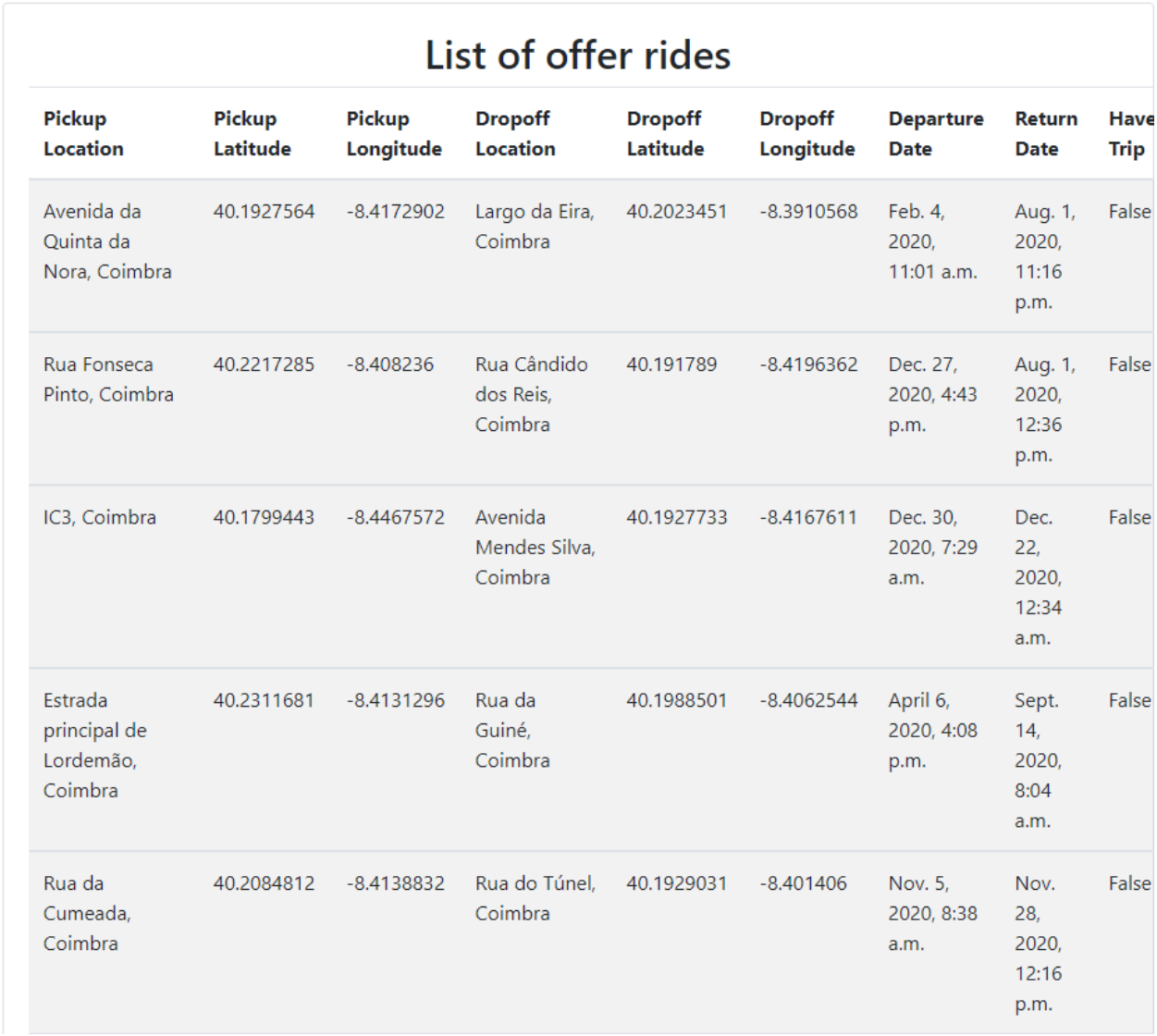


Figura 43 - Lista de ofertas de boleia.

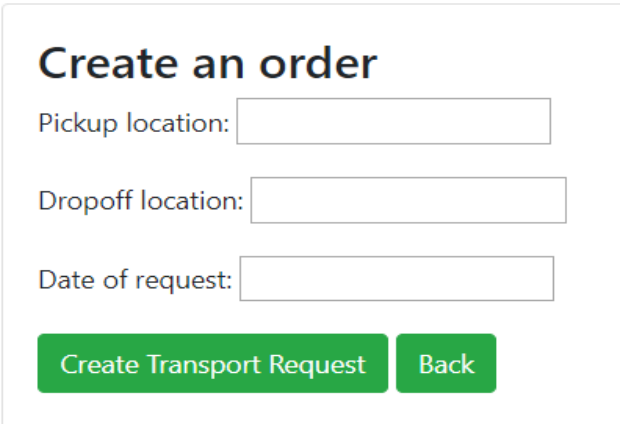


Figura 44 - Registo de pedidos de transporte por parte do passageiro.

Pickup Location	Pickup Latitude	Pickup Longitude	Dropoff Location	Dropoff Latitude	Dropoff Longitude	Date of the request	Have trip
Rua das Figueiras, Coimbra	40.1863057	-8.4105761	Rua Eduardo Alle Alvarez, Coimbra	40.20564	-8.41955	May 31, 2020, 3:05 p.m.	False
Rua António Maia, Coimbra	40.2034775	-8.3917008	Quinta de São Miguel, Coimbra	40.2080882	-8.4046205	Nov. 9, 2020, 10:29 p.m.	True
Rua da Fonte da Talha, Coimbra	40.20564	-8.41955	Rua dos Estudos, Coimbra	40.2089489	-8.4236162	May 3, 2020, 11:08 a.m.	False
Rua das Romeiras, Coimbra	40.1965523	-8.3968609	Rua das Laranjeiras, Coimbra	40.1908926	-8.4011303	June 12, 2020, 8 p.m.	False

Figura 45 - Lista de pedidos efetuados por parte do passageiro.

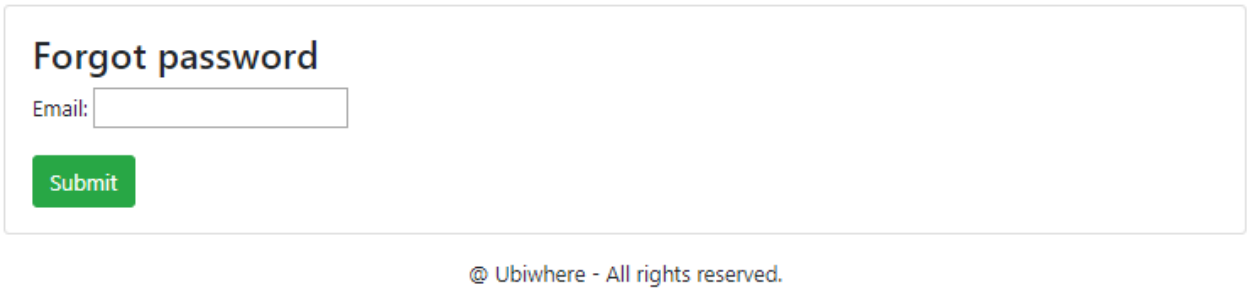


Figura 46 - Recuperação de password caso o utilizador se tenha esquecido.

Anexo E – Proposta de estágio

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E DE SISTEMAS

UBIWHERE, LDA

PROPOSTA DE ESTÁGIO

Ano Letivo de 2019/2020

em Mestrado em Informática e Sistemas Desenvolvimento de Software

TEMA

On-demand Mobility Platform

SUMÁRIO

O estágio proposto foca-se no vetor da Mobilidade Inteligente, neste caso, com uma solução para otimizar os pontos de recolha de pessoas numa cidade, ou num percurso até à cidade. O principal objetivo do estágio é compreender e agregar pedidos de transporte (num sistema *on-demand*), agrupar as pessoas e garantir a otimização da rota e respetivos pontos de recolha.

1. ÂMBITO

A Ubiwhere tem vindo a desenvolver diversos produtos na área das Smart Cities, que produzem serviços de valor acrescentado aos gestores das cidades. Neste âmbito, pretende-se aproveitar todo o conhecimento já desenvolvido para ajudar as regiões que necessitam de infraestruturas e opções de mobilidade a criar soluções de mobilidade urbana sustentável para a população mais envelhecida ou que não tenha acesso a veículo próprio. Assim, a presente proposta tem como visão a criação de um serviço de mobilidade *on-demand* capaz de agregar todos os serviços de mobilidade presentes numa cidade e fazer a correlação de disponibilidade e pedidos de boleia.

2. OBJECTIVOS

O presente projecto pretende atingir os seguintes objectivos:

- Produção de um documento de estágio que inclua estado da arte e decisões tecnológicas para o desenvolvimento da solução;
- Implementação de clustering de pedidos de transporte;
- Implementação do registo de pedidos de transporte;
- Criação de um mecanismo de atribuição automática pedido de transporte, pessoa e disponibilidade do veículo;
- Integração na solução já presente na empresa.

Departamento de Engenharia Informática e de Sistemas

Ubiwhere, Lda

3. PROGRAMA DE TRABALHOS

O estágio consistirá nas seguintes actividades e respectivas tarefas:

- ***T1*** *- Estudo do problema e Elaboração do estudo do Estado da Arte* - Nesta tarefa pretende-se que se faça o levantamento do estado da arte e tecnologias necessárias à concepção da solução;
- ***T2*** *- Levantamento e especificação de requisitos* - Esta tarefa tem como objetivo, fazer o levantamento dos requisitos da solução sendo estes alicerçados no estado da arte. Deverá também contribuir-se com a arquitetura lógica do sistema;
- ***T3*** *- Desenvolvimento da solução* - Nesta tarefa serão implementados os requisitos da solução;
- ***T4*** *- Testes* - Nesta tarefa são realizados os testes ao protótipo desenvolvido;
- ***T5*** *- Elaboração do Relatório Final do Projeto* - Nesta tarefa é feito o relatório final do projeto.

4. CALENDARIZAÇÃO DAS TAREFAS

As Tarefas acima descritas, incluindo os testes de validação de cada módulo, serão executadas de acordo com a seguinte calendarização:

	Meses											
Tarefas		N		N+1		N+2		N+3		N+4		N+5	
T1													
T2													
T3													
T4													
T5													
Metas	INI			M1		M2						M3	M4

INI		Início dos trabalhos
M1	(INI + 6 Semanas)	Tarefa T1 terminada
M2	(INI + 10 Semanas)	Tarefa T2 terminada
M3	(INI + 20 Semanas)	Tarefa T3 terminada
M4	(INI + 22 Semanas)	Tarefa T4 terminada
M5	(INI + 24 Semanas)	Tarefa T5 terminada

5. RESULTADOS

Os resultados dos estágio serão consubstanciados num conjunto de documentos a elaborar pelo estagiário de acordo com o seguinte plano:

M1 - Estado da arte
  R1.1: Relatório do estado da arte
M2 - Especificação de requisitos e arquitetura da solução
  R2.1: Relatório de especificação de requisitos
M3 - Solução funcional
  R3.1: Relatório da implementação
M4 - Relatório final de estágio
  R4.1: Relatório final de estágio

6. LOCAL DE TRABALHO

Ubiwhere's R&I Center

Rua Pedro Nunes - IPN , Escritório 2.18

3030-199 Coimbra.

7. METODOLOGIA

Metodologia baseada em SCRUM.

Organização de um **Dossier de Projecto**, e reuniões.

8. ORIENTAÇÃO

ISEC:

Nome Ana Alves (aalves@isec.pt)
Categoria: Professor Adjunto

Entidade de Acolhimento: Ubiwhere
Nome André Duarte (aduarte@ubiwhere.com)
Cargo: Future Mobility Engineer & DevOps

9. CARACTERIZAÇÃO E REMUNERAÇÃO

- Data de início - 16 setembro 2019
- Data de fim - 29 junho 2020
- Horário - Horário Flexível

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E DE SISTEMAS

UBIWHERE, LDA

- Tipo de regalias oferecidas - Bolsa de estágio, equivalente ao subsídio de alimentação
- Tipo de formação oferecida aos estagiários - Formação mensal contínua