ALGORITMOS PARA OTIMIZAÇÃO DE ROTAS DE DISTRIBUIÇÃO

ALGORITHMS TO OPTIMIZING DISTRIBUTION ROUTES

Tiago Duarte[1]; Daniel Carvalho[1]; Domingos Martinho[1,2]

[1]ISLA Santarém; [2]CEPESE, Porto
tiago.fmduarte87@gmail.com; djhcarvalho@gmail.com; domingos.martinho@islasantarem.pt

Resumo

Atualmente vem-se assistindo a um aumento dos serviços de entregas de mercadorias a clientes que são oferecidos pelas organizações de diversos setores de modo a aumentar a oferta dos seus produtos e serviços e a combater a concorrência no mercado. O consumo de tempo, atrasos devido a trânsito e outras circunstâncias fazem também com que este tipo de serviços aumente os seus custos de forma muito significativa.
Surge assim a oportunidade para o desenvolvimento de soluções que permitam a otimização das rotas de modo a minimizar os custos inerentes ao processo o que tem levado muitos investigadores a estudarem o problema e ao surgimento no mercado de diversas aplicações que pretendem ajudar as empresas nesse tipo de serviços, fazendo uso de algoritmos desenvolvidos para a otimização de rotas.
Neste contexto este trabalho tem como objetivo a criação de uma plataforma que permita a inserção de encomendas de clientes e posteriormente otimizar as rotas para as entregas associadas a cada veiculo. Esta plataforma vai permitir ainda que os vendedores possam efetuar as encomendas dos clientes em qualquer lugar, bastando para isso ter acesso à internet. Após as encomendas estarem inseridas, serão agrupadas por data e distribuídas pelas viaturas que as irão entregar aos clientes de áreas próximas. Através das moradas dos clientes o sistema irá calcular a melhor rota que a logística da empresa poderá utilizar, tendo em conta fatores como portagens e caminho mais rápido.
Espera-se assim implementar uma solução que possa ter uma utilização universal sem os constrangimentos das soluções proprietárias e as limitações das soluções open source.

Palavras chave: Algoritmos; Clientes; Encomendas; Otimização de rotas.

Abstract

Nowadays, there is an increase in the services of deliveries of goods to customers that are offered by organizations from various sectors in order to increase the supply of their products and services and to combat competition in the market. The consumption of time, delays due to traffic and other circumstances also make this type of services increase their costs very significantly.
This gives rise to the opportunity for the development of solutions that allow the optimization of routes in order to minimize the costs inherent to the process, which has led many researchers to study the problem and the emergence in the market of several applications that intend to help companies in this type of using algorithms developed for the optimization of routes.
In this context, this project aims to create a platform that allows the insertion of customer orders and subsequently optimize the routes for the deliveries associated with each vehicle. This platform will also allow salespeople to place customer orders anywhere, simply by having access to the internet. After the orders are inserted, they will be grouped by date and distributed by the vehicles that will deliver them to customers from nearby areas. Through customer addresses, the system will calculate the best route that the company's logistics can use, taking into account factors such as tolls and the fastest route.
It is hoped to implement a solution that can be universally used without the constraints of proprietary solutions and the limitations of open source solutions.

Keywords: Algorithms; Customers; Orders; Optimization of routes.

A sustentabilidade económica e financeira de uma organização é essencial para a sua sobrevivência e evolução, de forma a não permitir quebras que levem ao seu encerramento. A avaliação dos gastos em termos de entregas de mercadorias a clientes constitui uma peça fundamental nas tomadas de decisão sobre o preço dos transportes e das mercadorias vendidas. As empresas que necessitam de entregar os seus produtos ou as que prestam este tipo de serviços têm apresentado uma melhoria sustentada na otimização dos recursos, conseguindo melhorar e desenvolver métodos que minimizem alguns dos fatores adversos provenientes do tráfego rodoviário (Kawamura, 2006). Neste contexto a capacidade de planeamento das rotas a efetuar constitui um aspeto crítico uma vez que possibilita à organização efetuar as suas tarefas de forma eficaz e eficiente e desse modo obter os melhores resultados (Kawamura, 2006).

O objetivo que se pretende atingir com este trabalho, que se encontra em curso, consiste em planear as rotas para a frota de veículos, sem violação das restrições de tempo e capacidade, minimizando custos. Os custos normalmente estão relacionados com a distância percorrida, o número de veículos necessário para efetuar as entregas, o tempo total de espera dos veículos nos consumidores ou à combinação destes.

1. O ESTADO DA ARTE

1.1 A Solução do Problema

Para solucionar o problema da otimização de rotas é comum recorrer-se á teoria do Problema do Caixeiro Viajante e do Sistema de Formigas para elaborar os algoritmos.

O Problema do Caixeiro Viajante pode ser resumido com a seguinte pergunta: dado um número N de cidades que devem ser visitadas por um caixeiro, qual a sequência de cidades que torna o comprimento do percurso o menor possível, considerando o início e o término na mesma cidade? Sendo que, todas as cidades são interligadas umas as outras e que cada cidade deve ser visitada uma única vez (Pinheiro, Jale, & de Sousa, 2010). O Problema do Caixeiro Viajante considera-se um grafo em que os vértices representam as cidades e as arestas representam as estradas de uma dada região (a cada aresta está associada a distância entre cidades). Sob a ótica de otimização, os problemas de roteirização de veículos, incluindo o caso particular do caixeiro viajante, pertencem à categoria conhecida como NP-difícil (do inglês “NP-hard”), o que significa que possuem ordem de complexidade exponencial. Por outras palavras, o esforço computacional para a sua resolução cresce exponencialmente com o tamanho do problema (dado pelo número de pontos a serem atendidos) (Cunha, 2000).

O Sistema de Formigas como o nome indica é inspirado no método utilizador por insetos que segundo Bonabeau e Meyer (2001) assenta em três características:

- Flexibilidade. A colonia adapta-se a um ambiente em mudança.
- Robustez. Mesmo quando um ou mais indivíduos falham, o grupo ainda pode executar suas tarefas.
- Auto-organização. As atividades não são controladas centralmente nem supervisionadas localmente.

Através da auto-organização, o comportamento do grupo emerge das interações coletivas de todos os indivíduos. De fato, um dos principais temas recorrentes na inteligência de enxames (e na complexidade da ciência em geral) é que, mesmo que os indivíduos sigam regras simples, o comportamento do grupo resultante pode ser surpreendentemente complexo - e notavelmente efetivo. E, em grande medida, flexibilidade e robustez resultam da auto-organização (Bonabeau & Meyer, 2001).
Para aumentar a eficácia vários autores têm vindo a defender uma abordagem para a resolução do problema através da conjugação dos dois sistemas (Carvalho, 2007; Barbosa, Jr., & Kashiwabara, 2015).

1.2 Algoritmos

A resolução do problema da otimização de rotas sendo de elevada complexidade recorre a modelação matemática e uso tecnológico de modo a simplificar o problema através da teoria de grafos produzindo algoritmos cada vês mais eficazes (Cunha, 2000).

De forma genérica para solucionar o problema têm sido apresentados diversos algoritmos que, tendo em conta a abordagem utilizada, podem ser classificados do seguinte modo:

- Algoritmos de métodos exatos
- Algoritmos heurísticos
- Algoritmos meta-heurísticos
- Algoritmos genéticos

1.2.1 Algoritmos de métodos exatos

Os métodos exatos são algoritmos de pesquisa exaustiva que verificam todo o conjunto de soluções de determinado problema até encontrar a solução ótima. Sendo esta a sua principal vantagem, ou seja, garantem a solução ótima. No entanto, a sua modelação torna-se mais complexa e a sua aplicação a problemas mais complexos ou de grandeza maior faz com que tenha grandes dificuldades a encontrar a solução ótima num intervalo de tempo adequado. Assim sendo o esforço computacional para a sua resolução cresce exponencialmente (Júnior & Cechin, 2006).

Algoritmo Branch and Bound

Branch and Bound baseia-se em três partes distintas. Como refere Guerreiro (2009) inicia-se com a construção de árvores de nós onde divide o problema em conjuntos menores ou subproblemas menores. De seguida cria-se a estratégias a desenvolver o próximo subproblema e consequentemente realiza comparações com o seu limite superior e inferior (parâmetros a estabelecer), em que a solução tem que corresponder para que seja considerada viável. A segunda parte Branching dividir o problema principal em subproblemas menores de modo a facilitar a análise, eliminando soluções inviáveis, sem comprometer a integridade do campo de soluções. A terceira Bounding eliminar soluções de baixa qualidade através de comparações com limitantes (Guerreiro, et al., 2009).

São comumente utilizados dois tipos de limitantes: superior e inferior. Num problema de minimização, o limitante superior é um valor conhecido e viável da função objetivo, não necessariamente o valor ótimo, que tem o papel de servir como parâmetro para avaliar soluções obtidas, ou seja, soluções com valores superiores ao limitante superior são descartadas por se tratarem de soluções piores do que a atualmente conhecida (Kawamura, 2006). Por sua vez, o limitante inferior, em um problema de minimização, é uma estimativa da função objetivo tendo-se como base a solução parcial até então obtida. Note-se que o limitante inferior é sempre menor ou igual do que o valor da função objetivo, já que seu cálculo é baseado em um subconjunto da solução enquanto que a função objetivo é calculada considerando-se a solução completa. Assim sendo, é possível eliminar soluções que tenham limitantes inferiores piores do que os atuais limitantes superiores conhecidos (Kawamura, 2006).

Algoritmo de Dijkstra

O algoritmo de Dijkstra, cujo esquema é apresentado na figura 1, funciona de forma diferente dos algoritmos descritos anteriormente, a sua função é procurar o caminho mais curto entre duas arestas do mesmo grafo, assim calcula o custo mínimo de um vértice para todos os outros vértices dentro do mesmo grafo (Misa, 2010).

Segundo Chao (2010) o algoritmo parte de uma estimativa inicial para o custo mínimo e vai ajustando sucessivamente essa estimativa. Enquanto vai percorrendo o grafo, o algoritmo vai considerar fechar o vértice assim que obtiver o custo mínimo do vértice do caminho principal, caso não represente o custo mínimo vai permanecer em aberto. A referida análise é executada a partir dos valores entre cada vértice, e neste caso é considerado as distâncias. Quando forem executados todos os vértices, será apresentada a solução que terá o menor custo. Este algoritmo é simples e rápido embora não garanta os melhores resultados sempre que existam arcos com valores negativos, a velocidade de execução pode diminuir muito nas situações mais complexas e mais amplas.

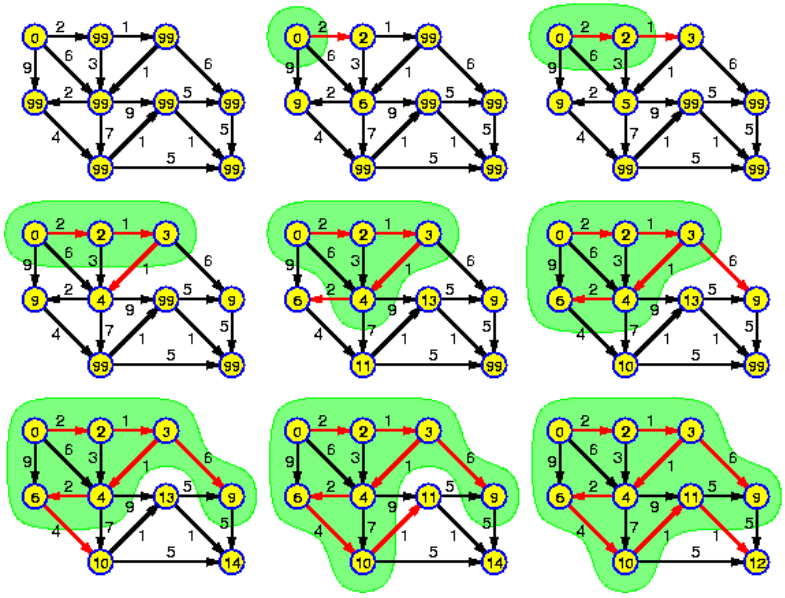

Figura 1. Algoritmo de Dijkstra (Jasika, et al., 2012)

1.2.2 Algoritmos heurísticos

Os algoritmos heurísticos foram produzidos com o objetivo de serem de fácil execução e implementação e que produzissem resultados de boa qualidade (Bonabeau & Meyer, 2001).

Num determinado problema o ideal seria analisar todas as soluções possíveis para chegar à melhor. No entanto, as heurísticas têm a desvantagem de explorar parcialmente a totalidade das soluções possíveis, ou seja, adotam uma estratégia que se apoiam numa abordagem intuitiva (Malaquias, 2006).

As heurísticas não garantem se uma solução é ótima ou o quão próximo está da solução ótima, mas encontram soluções boas num tempo razoável (Silva, 2013).

Algoritmo Savings

O algoritmo de Savings, representado na figura 2, tenta encontrar o menor custo com a combinação de duas rotas em apenas uma (Galvão, 2017).

Este algoritmo tem duas visões de abordar o problema, uma delas é visitar os clientes i e j em rotas separadas (2.4a). Em alternativa é fazê-lo na mesma rota, ou seja, visitar os clientes sequencialmente (2.4b). Visto que os custos das rotas são dados, o resultado pode ser calculado para saber qual a melhor rota. Ainda é possível dividir o algoritmo em duas versões, sequencial e paralela. Na versão sequencial apenas é formada uma rota de cada vez, enquanto na versão paralela é criada mais que uma rota de cada vez (Lysgaard, 1997).

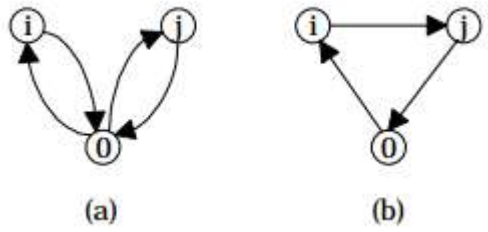

Figura 2. Algoritmo de Savings (Lysgaard, 1997)

Algoritmo de Nearest Neighbor

O principal objetivo deste algoritmo é procurar o cliente mais próximo a visitar com a finalidade de gerar poupanças de rotas, como não prevê os passos seguintes pode gerar problemas de eficácia (Galvão, 2017).

Este algoritmo baseia-se numa metodologia que segundo Silva (2003) constrói uma solução passo a passo seguindo um conjunto de requisitos pré-estabelecidos. Este processo iterativo repete-se até que não exista mais nenhum cliente a ser visitado, ou seja, todos os elementos sejam percorridos e a rota seja concluída.

No exemplo da figura 3 os círculos representam os vértices do grafo e a numeração fora dos círculos o tamanho das arestas, ou seja, a distância entre cada vértice.

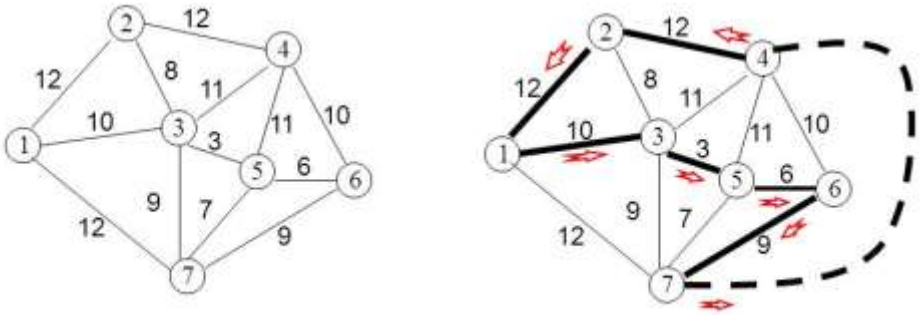

Figura 3. Nearest Neighbor adaptado (Walker, 2015)

1.2.3 Algoritmos de meta-heurística

Os algoritmos de Meta-heurísticas são algoritmos que obtém boas soluções e por vezes a melhor possível. Começam por depender da aplicação de heurística subordinada que é modificada para cada problema específico. As suas principais características são evitar as desvantagens dos métodos anteriores, ou seja, ter a capacidade de explorar um leque de soluções onde evita os locais ótimos e encontra soluções num tempo razoável (Silva, 2013).

Os procedimentos criados a partir dos métodos meta-heurísticos como se têm tornado muito eficientes são muito utilizados para este tipo de estudo.

Algoritmo Simulated Annealing

O algoritmo Simulated Annealing é utilizado para solucionar problemas de otimização discreta, com menor abrangência, e alguns problemas de natureza contínua. Exemplifica uma analogia com o comportamento termodinâmico, mais concretamente no processo de arrefecimento de sólidos (Oliveira, Vasconcelos, & Alvarenga, 2006).

Segundo Gheisari, Haghighat e Saadat (2008), o algoritmo em cada iteração compara a solução obtida com a solução existente para determinar qual a melhor. São aceites a soluções com melhor qualidade, enquanto soluções de menor qualidade são aceites como uma probabilidade que baixa com a temperatura.

Na figura 4 apresenta-se um esquema do algoritmo Simulated Annealing de Monte Carlo, que se baseia em grandes amostragens aleatórias para obter resultados, repetindo sucessivamente um elevado número de vezes a simulação para calcular probabilidades como se registassem os resultados reais em jogos de casino (Hastings, 1970).

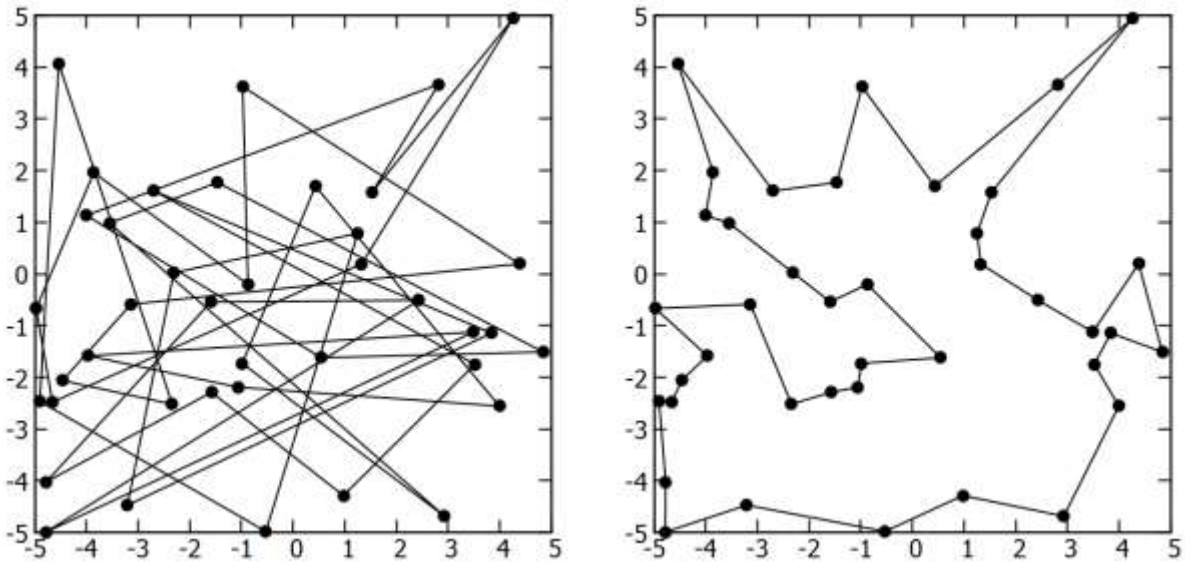

Figura 4. Algoritmo Simulated Annealing (Soligno, 2017)

Algoritmo de Tabu Shearch

O algoritmo de Tabu Shearch foi criado por Fed W. Glover em 1986 e é um método de busca local utilizado para otimizações. É uma adaptação do procedimento com a capacidade de fazer uso de muitos outros métodos, como algoritmos de programação linear e heurística especializada, que direciona para superar as limitações da otimização local (Glover, 1989).

Segundo Charbonneau e Vokkarane (2010), de início deve-se gerar uma solução aleatória ou com origem noutro algoritmo. Partindo da solução inicial serão produzidas novas soluções sendo a melhor será selecionada como solução atual. Os algoritmos por vezes bloqueiam em ótimos locais, para solucionar o problema este algoritmo cria uma lista tabu. Guan, Cao e Shi (2010) referem que a lista tabu regista movimentos produzidos para formar as soluções selecionadas que não geram repetições enquanto estiverem na lista. A lista tabu é um fator essencial e determinante na qualidade do algoritmo Tabu Search (figura 5).

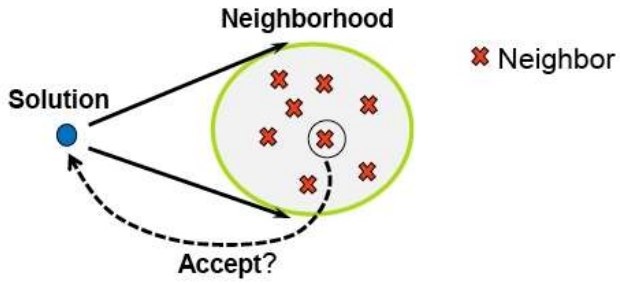

Figura 5. Tabu Search (Paradiseo, s.d.)

Algoritmo Greedy Randomized Adaptive Search Procedure (GRASP)

O Algoritmo Greedy Randomized Adaptive Search Procedure (GRASP), introduzido por Feo e Resende (1995), é um algoritmo meta-heurístico de processos iterativos que gera normalmente soluções perto do ótimo. A principal característica do algoritmo, apresentado na figura 6, é criar soluções novas independentes das anteriores. Na sua versão básica divide-se em duas fases em cada iteração, a fase de construção e a fase de procura local. Na fase de construção é pretendido encontrar uma solução para o problema e na fase de procura local melhorar a solução encontrada.

Em cada passo do algoritmo é escolhida a melhor componente, identificado por um indicador de sensibilidade, que utiliza uma componente que tem características aleatória e adaptativa, controlado pelo parâmetro α para indicar as componentes de melhor qualidade para fazerem parte da solução do problema (Resende & FEO, 1989). Na fase de melhoria local é utilizado um algoritmo de busca local com o objetivo de procurar na vizinhança da solução fornecida pela fase construtiva, uma melhor solução para o problema. O GRASP pode contar ainda com uma fase de pré-processamento na qual, as informações sobre o problema são pré-processadas e avaliadas com o objetivo de diminuir o espaço de busca do problema caso seja possível (Souza S. S., 2013).

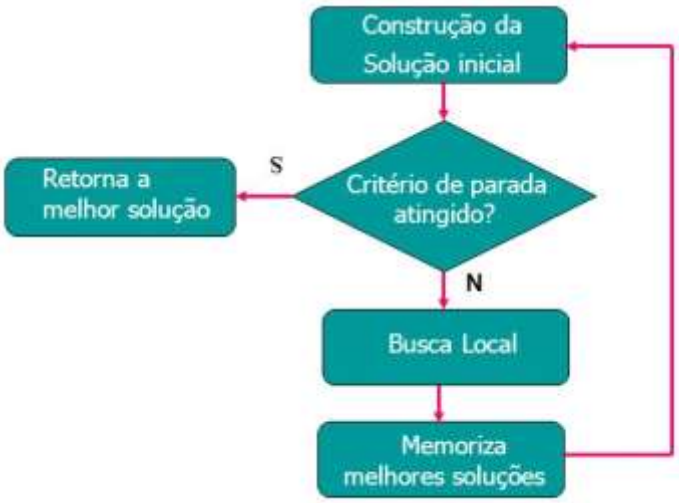

Figura 1. Algoritmo GASP (Souza, 2014)

1.2.4 Algoritmos genéticos

Algoritmos Genéticos (AG) são algoritmos robustos, usados nos mais variados problemas de diferentes domínios. São baseados em procedimentos de seleção natural e de genética (Botassoli, Furtado, & Alberti, 2015).

Os AG constituem uma técnica de busca e otimização inspirada no princípio Darwiniano de seleção natural e reprodução genética. Os princípios da natureza nos quais os AG se inspiram são simples. De acordo com a teoria de C. Darwin, o princípio de seleção privilegia os indivíduos mais aptos com maior longevidade e, portanto, com maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chance de perpetuarem seus códigos genéticos nas próximas gerações. Tais códigos genéticos constituem a identidade de cada indivíduo e estão representados nos cromossomas. Estes princípios são imitados na construção de algoritmos computacionais que buscam uma melhor solução para um determinado problema, através da evolução de populações de soluções codificadas através de cromossomas artificiais (Pacheco, 1999).

Os AG utilizam uma analogia direta deste fenômeno de evolução na natureza (figura 7), onde cada indivíduo representa uma possível solução para um problema dado. A cada indivíduo se atribui uma pontuação de adaptação, dependendo da resposta dada ao problema por este indivíduo. Aos mais adaptados é dada uma maior oportunidade de se reproduzirem mediante cruzamentos com outros indivíduos da população, produzindo descendentes com características de ambas as partes (Palma-Chilla, Lazzarus, & Ponce, 2011).

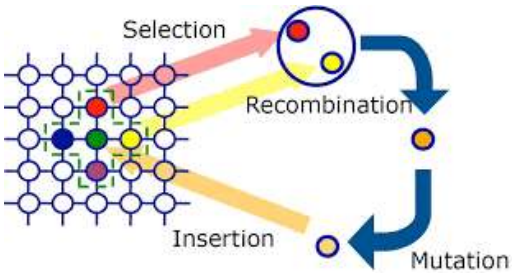

Figura 2. Algoritmos Genéricos (Botassoli, Furtado, & Alberti, 2015)

1.3 As soluções tecnológicas

Sendo a otimização de rotas um problema de elevada complexidade que corresponde a uma parte substancial dos custos de muitas empresas, fez com que surgissem no mercado inúmeras soluções tecnológicas para tentar simplificar/resolver o problema.

Os principais desenvolvedores de aplicações direcionados para as organizações empresariais como os ERP´s com módulos vocacionados para a área de logística incluem na sua oferta sistemas integrados para gestão de encomendas e logística com otimização de rotas que são utilizados principalmente pelas grandes empresas de distribuição e logística. São disso exemplo os softwares Primavera Gestão de Armazéns Eye Peak e o SAP Transportation Manager (Gomes, 2015). Estão disponíveis também soluções gratuitas para otimização de rotas não integradas no software empresarial, mais simples e com menos opções, não permitem por exemplo gerir rotas de vários veículos em simultâneo e a introdução dos pontos de entrega é manual duplicando assim esse trabalho. Alguns dos softwares gratuitos mais conhecidos são o Google Maps, Open Source Routing Machine ou Open Street Map (Huber & Rust, 2016).

O Primavera Gestão de Armazéns Eye Peak é um software especifico para gestão de armazéns e distribuição com possibilidade de integração com o ERP Primavera, a solução divide-se em duas áreas de atuação, sendo WMS Gestão Avançada de Armazéns a parte de front-end relacionado com todo o processo administrativo da gestão de armazéns e modo wifi que permite fazer processos no armazém, o DMS Gestão de Serviços Distribuição e Entrega que contem uma parte de backoffice que permite a configuração e parametrização dos serviços e o Terminal GSM para utilização no exterior das instalações. As principais características são a criação de serviços que permitem a tipificação de clientes, o planeamento de rotas e entregas, confirmação de recolhas e entregas, gestão de carga da viatura, classificação das causas das não entregas e devoluções, registo eletrónico de entregas para maior controlo de qualidade, reportes construção de relatórios de informação e exploração do negócio (Primavera BSS, s.d.).

O SAP Transportation Manager (STM) é um software construído como um add-on do SAP ERP e outros softwares de gestão da SAP que necessitem da sua integração, estando construído para lidar com leque muito abrangente de fatores que podem influenciar os custos e prazos de entrega de mercadorias tendo em conta todos os fenómenos influenciados pela globalização havendo unidades de software específicos para área de atuação de cada empresa. O STM permite: determine o plano de transporte mais eficiente, enquanto cumpre as restrições fornecidas (como acordos de nível de serviço, custos e disponibilidade de recursos), identificar oportunidades de redução de custos (como possibilidades de consolidação e a escolha do melhor meio de transporte), maximizar o uso de recursos existentes (como usar sua própria frota), reagir a eventos de execução e resolver possíveis conflitos com o plano inicial (SAP, 2017).

O Google Maps é um serviço de mapas online desenvolvido pela Google com aplicações para sistemas operativos IOS e Android, dispõem de diversas funcionalidades (GoogleMaps, s.d.):

- Monitorizar as informações de transito em tempo real e encontrar o melhor trajeto para o destino, navegação curva a curva e a orientação da faixa de rodagem ao longo de todo o trajeto.
- Indicar dinamicamente um novo trajeto com base no padrão de transito atual de modo a evitar engarrafamentos de trânsito.
- Permite comentários e fotos de habitantes para tomar uma decisão mais informada de locais a visitar, analise do destino antes de lá chegar com o Street View e os mapas interiores, imagens de satélite.
- Art Project para percorrer os melhores locais do mundo, partilha de conhecimentos locais, guardar endereços de casa e trabalho para uma pesquisa mais rápida, comentários e críticas de locais.

O Open Source Routing Machine também conhecido por OSRM é um comando que calcula a distância e o tempo de viagem entre dois pontos usando informações de latitude e longitude, é usado o OSRM em conjunto com o OpenStreetMap (OSM) para encontrar a rota ideal, o procedimento é especialmente construído para grandes conjuntos de dados georreferenciados (Huber & Rust, 2016). Porque é rápido, o comando usa toda a capacidade computacional de um PC, permite ao utilizador fazer pedidos ilimitados, e é independente da Internet e fornecedores comerciais on-line. Portanto, não há risco de o comando se tornar obsoleto. Além disso, os resultados podem ser replicados a qualquer momento (Huber & Rust, 2016).

O OSM é um mapa digital gerido pela Fundação OpenStreetMap e desenvolvido por uma comunidade voluntária de mapeadores que contribuem e mantêm atualizados os dados sobre estradas, trilhos, cafés, estações ferroviárias e muito mais por todo o mundo. Os colaboradores utilizam fotografias aéreas, dispositivos GPS, e mapas do terreno para verificar que a informação no OSM é rigorosa e atualizada (OpenStreetMap, s.d.). O OSM é constituído por dados abertos por isso qualquer pessoa tem a liberdade de usar os dados para qualquer fim desde credite a autoria do OSM e os seus colaboradores.

2. METODOLOGIA

Neste trabalho adotou-se uma metodologia ágil - o scrum – que podendo ser utilizada para qualquer tipo de projeto complexo (Greem, 2016), tem sido utilizada em todo o mundo, desde o FBI, agências de Marketing e construtores. Para qualquer tipo de produto que esteja a ser desenvolvido o scrum pode ser aplicado de forma a ajudar no planeamento e desenvolvimento do produto em questão.

O Scrum é descrito como uma metodologia ágil para a gestão e planeamento de projetos de software (Scrum, 2018). Nesta metodologia os projetos são divididos em ciclos, por norma com a duração máxima de um mês, que são denominados de sprints. O sprint representa um espaço temporal dentro do qual uma atividade ou conjunto de atividades será executado.

O Scrum tem designado um Product Owner (dono do produto) que representa quem está a criar o projeto e tem autoridade para dizer o que vai ou não fazer o produto final. É encarregado de elaborar uma lista de tarefas, necessidades e exigências do produto final. Neste projeto o Product Owner é a equipa de desenvolvimento do projeto. Terá também de ser definido um Scrum Master que irá ajudar a equipa a agir com base no planeamento do desenvolvimento do projeto.

Seguidamente vêm os sprints, que são o período de tempo determinado que a equipa de desenvolvimento do projeto terá para efetuar as tarefas programadas. O período de tempo para a realização destas tarefas depende das necessidades que terão para o seu desenvolvimento. A equipa de desenvolvimento do projeto tem necessidade de reunir pelo menos uma vez por semana com o nosso Scrum Master de forma a reportar o estado de desenvolvimento do projeto. Cada sprint termina com a conclusão da tarefa e com uma revisão ou retrospetiva onde será analisado o trabalho efetuado e onde serão discutidas melhorias a efetuar no próximo sprint (figura 8).

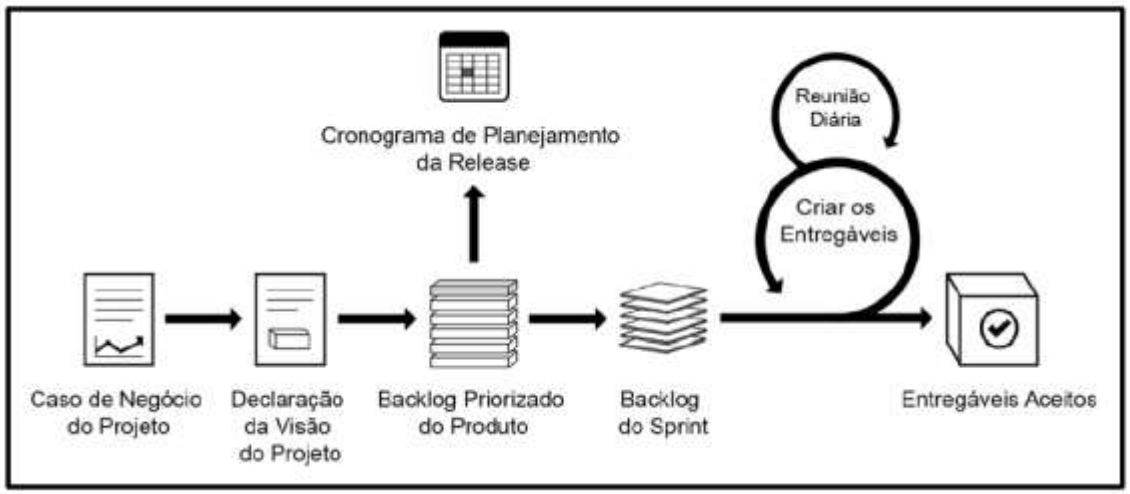

Figura 8. Fluxo do Scrum (SCRUMstudy, 2016)

3. O MODELO DO SISTEMA DE INFORMAÇÃO

3.1 Modelo de Negócio

A organização será constituída por uma unidade central (Sistema de Gestão de Encomendas) que fará a gestão de duas unidades organizativas: Unidade de Internet para Vendedores, Central de Gestão de Encomendas (figura 9).

3.1.1 Unidade de Internet para Vendedores

Nesta unidade os Vendedores registados e com devida permissão pode realizar as encomendas de produtos dos seus clientes e adicionar novos clientes, consultar produtos, alterar editar e anular encomendas. Os produtos disponíveis poderão ser consultados e inseridos na encomenda. Cada produto terá um código ID Produto, nome, designação, preço com e sem iva e a respetiva quantidade disponível em stock.

3.1.2 Central de Gestão de Encomendas (CGE)

Na Central de Gestão de Encomendas o Responsável de Logística registado e com permissão pode agrupar encomendas efetuadas pelos Vendedores para distribuição assim como consultar as rotas e proceder aos registos das viaturas.

- A CGE recebe as encomendas dos Vendedores no seu terminal que é adicionado á lista de encomendas a satisfazer.
- O Responsável de Logística vai agrupando as encomendas por área geográfica para cada viatura.
- Os funcionários de armazém recolhem as quantidades necessárias dos produtos para satisfazer as encomendas.

- É gerada uma rota por cada viatura.
- As encomendas são entregues na morada indicada pelo cliente.

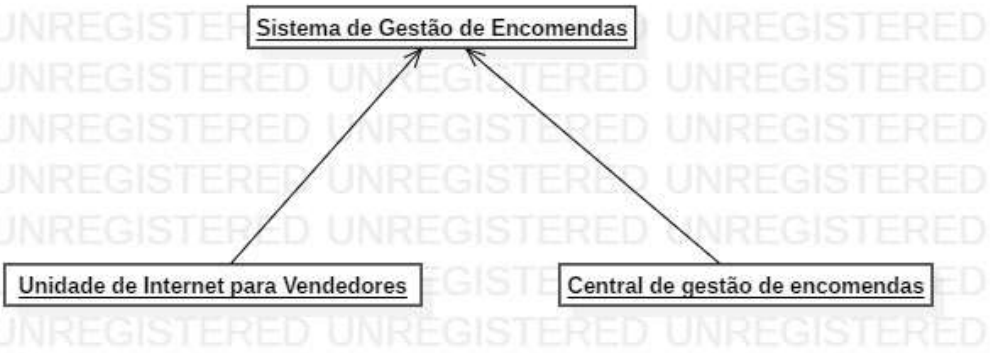

Figura 9. Estrutura Organizativa

3.2 Modelo de Domínio

No seguimento do Modelo de Negócio o sistema de informação foi organizado em 2 subsistemas: Subsistema de Internet e Subsistema central. Pretende-se que esta arquitetura tenha tenho uma elevada autonomia e que mesmo que uma falhe a outra continuara a funcionar mesmo sem comunicação entre subsistemas.

Os subsistemas terão uma base de dados comum onde poderão ter acesso aos dados necessários para realizar as diferentes tarefas. Esta arquitetura obriga a comunicação através de transações em XML para fazer a interligação entre diferentes subsistemas.

3.2.1 Subsistema Internet

É o sistema responsável pela realização das encomendas junto dos clientes através da sua identificação e morada de entrega, identificação dos produtos e respetivas quantidades.

3.2.2 Subsistema Central

A função deste sistema é gerir toda a informação gerada no processo de encomenda e respetiva distribuição, gerar as rotas para a distribuição e gerir a manutenção dos veículos. Por consequência deve de manter toda a informação atualizada referente a produtos, stocks, preços, encomendas, clientes, viaturas, manutenção de viaturas e respetiva quilometragem, pagamentos das encomendas.

3.3 Modelo Casos de Uso

Os use cases, ou traduzindo à letra “casos de utilização” ou “casos de uso”, constituem a técnica em UML para representar o levantamento de requisitos de um sistema. (Silva &

Videira, 2001). Na figura 10 apresentamos o modelo de casos de uso do sistema a implementar.

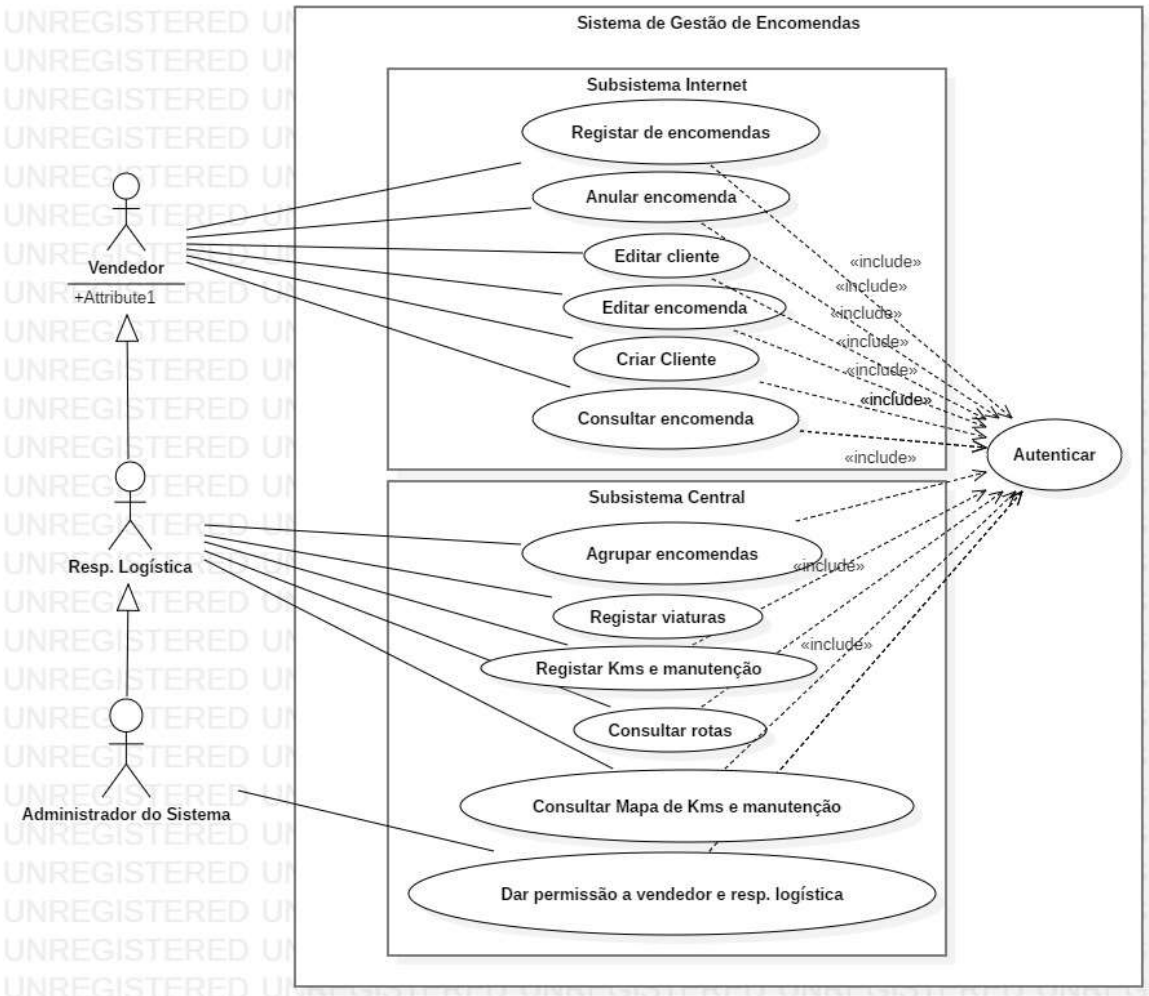

Figura 3. Modelo Casos de Uso

3.4 Diagrama de Classes

Segundo Nunes & O´Neill (2011) o digrama de classes é uma descrição formal da estrutura de objetos num determinado sistema. Para cada objeto é descrito a sua identidade e os seus relacionamentos com outros objetos, são descritos os seus atributos e as suas operações. Na figura 11 apresenta-se o diagrama de classes considerado adequado à resolução do problema.

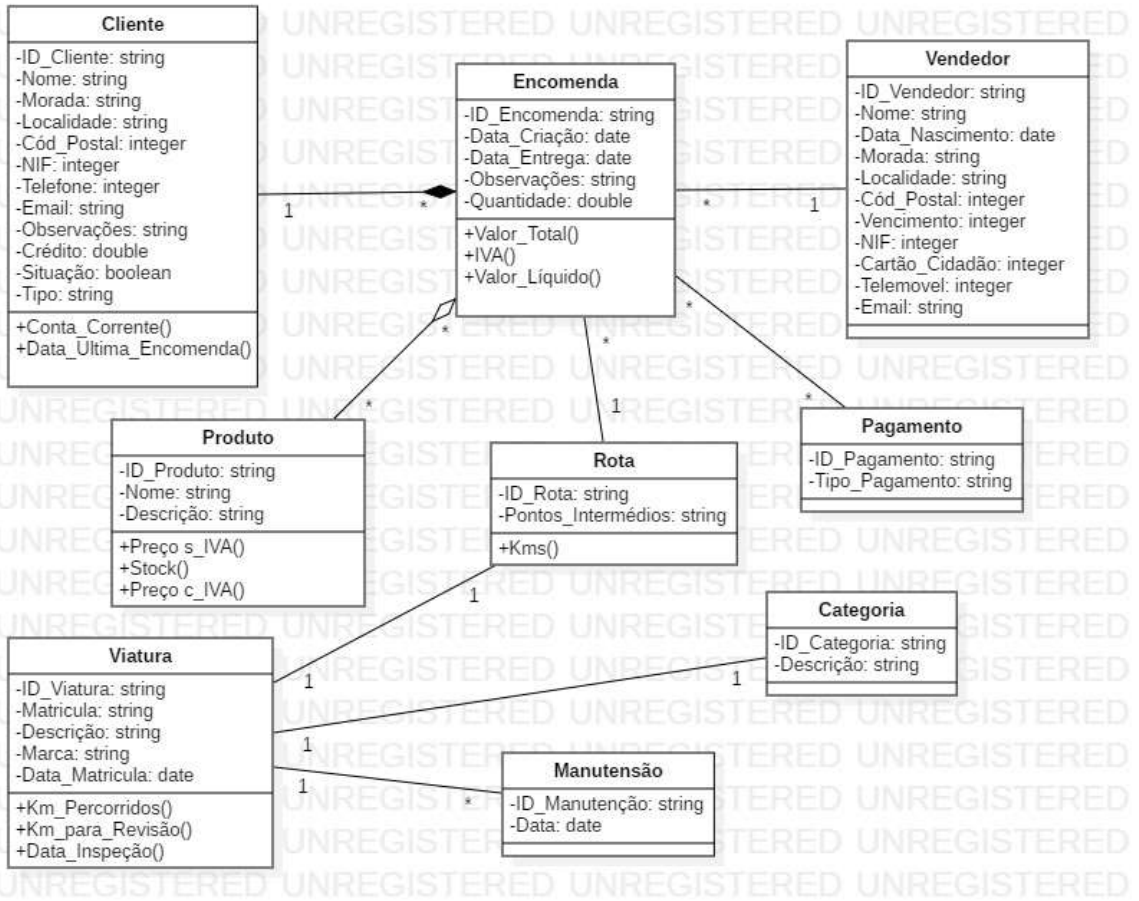

Figura 11. Diagrama de Classes

3.5 Diagrama de Atividades

O diagrama de atividades constitui um elemento de modelação simples, mas eficaz, para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamentos que possuam processamento paralelo (Nunes & O´Neill, 2011).

O exemplo, apresentado na figura 12, representa o processo de criação de uma encomenda com todas as opções disponíveis incluindo a possibilidade de cancelamento durante várias fazes do processo.

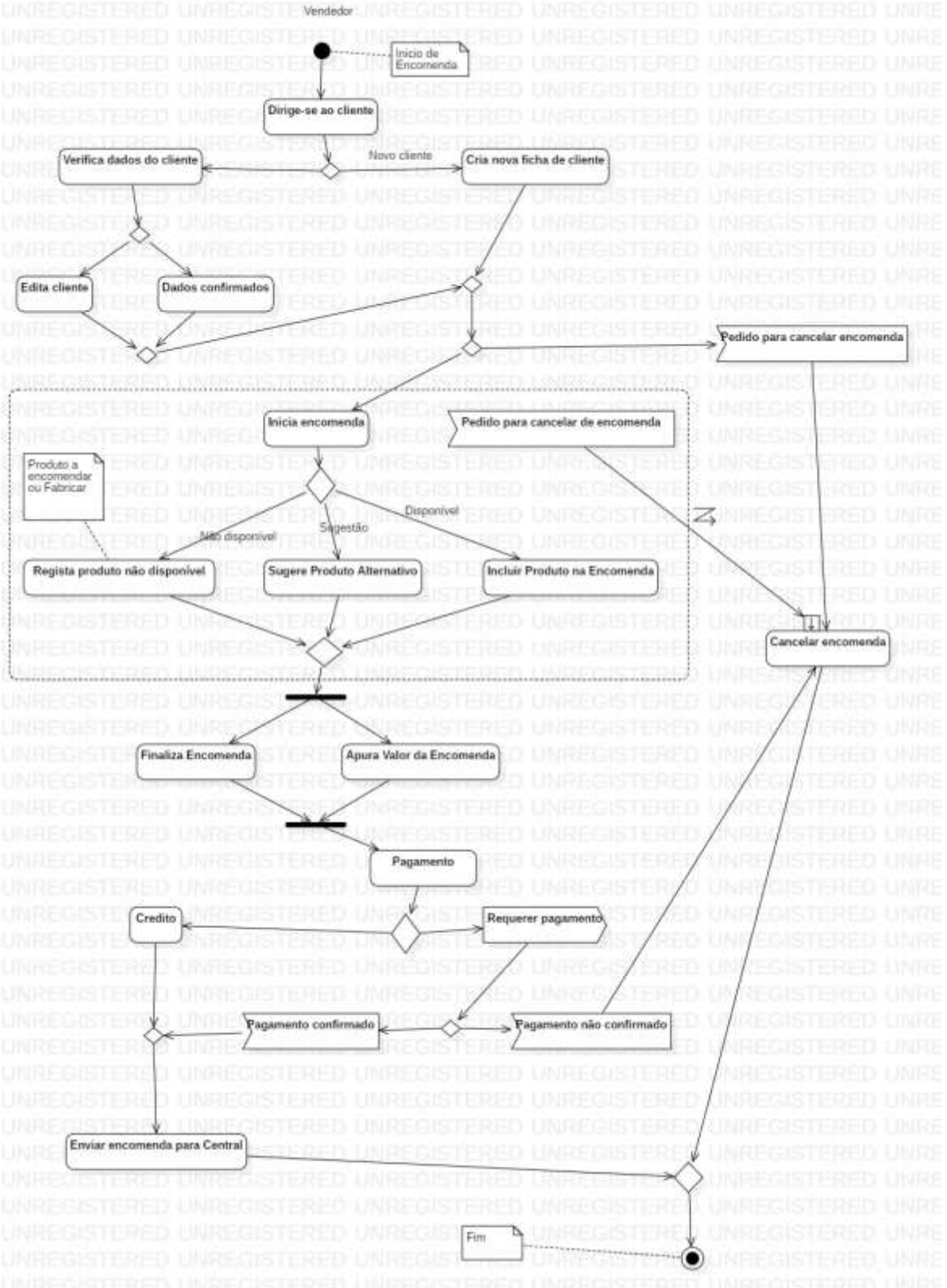

Figura 4. Diagrama de Atividade Encomenda

3.6 Diagramas de Sequência

Um diagrama de sequência apresenta as interações entre objetos a partir do encadeamento temporal das mensagens (Nunes & O´Neill, 2011). As mensagens trocadas entre objetos representam a invocação de um serviço (operação) disponibilizado por um objeto, com o objetivo de despoletar uma ação ou atividade. Uma definição mais formal descreve uma mensagem como a especificação da comunicação entre objetos (Nunes & O´Neill, 2011). Ilustrando o referido anteriormente, na figura 13 apresenta-se o processo de criação de encomenda.

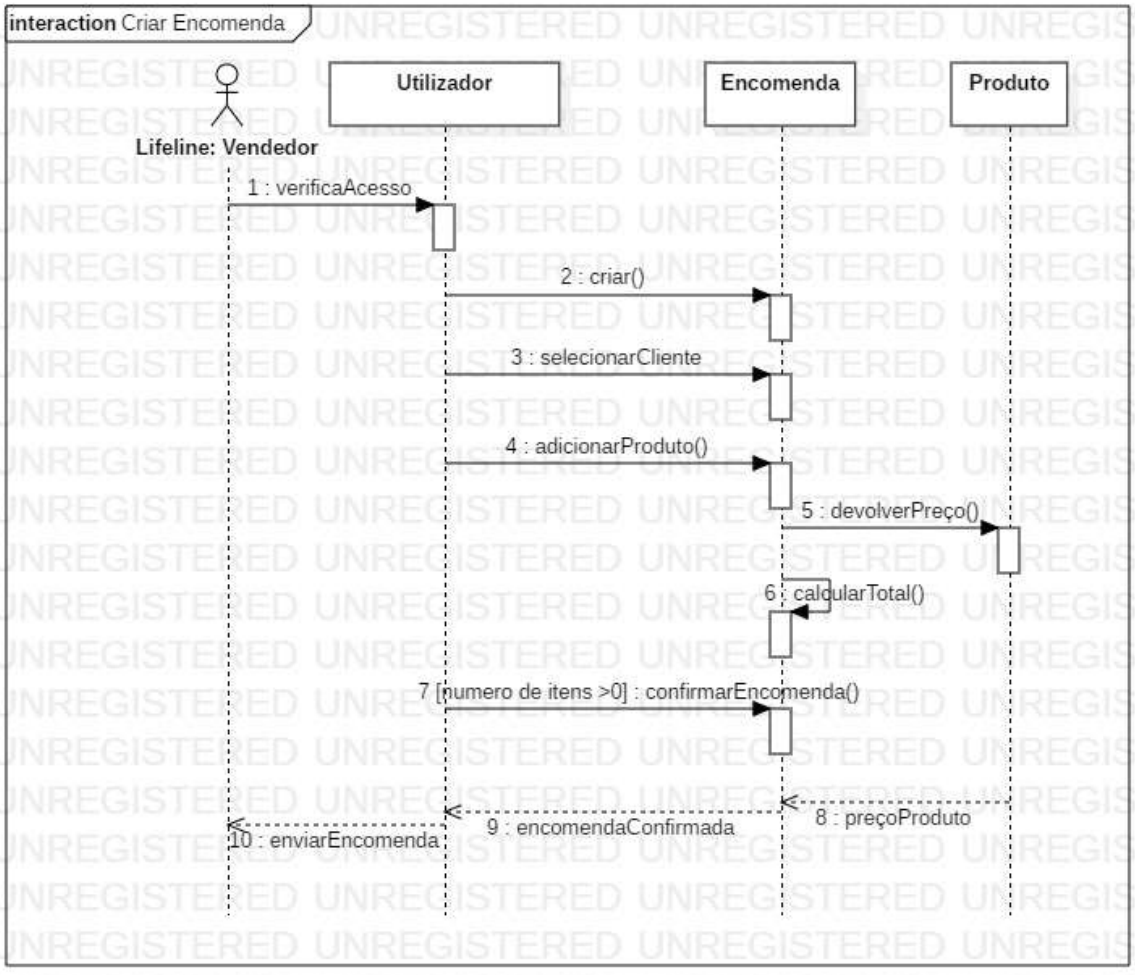

Figura 13. Diagramas de sequência Criar Encomenda

3.7 Modelo de Implementação

Segundo Nunes & O´Neill (2011) o diagrama de instalação pretende descrever a arquitetura de hardware do sistema e as suas e a relação com os diferentes componentes (software). Na figura 14 apresenta-se o diagrama de instalação.

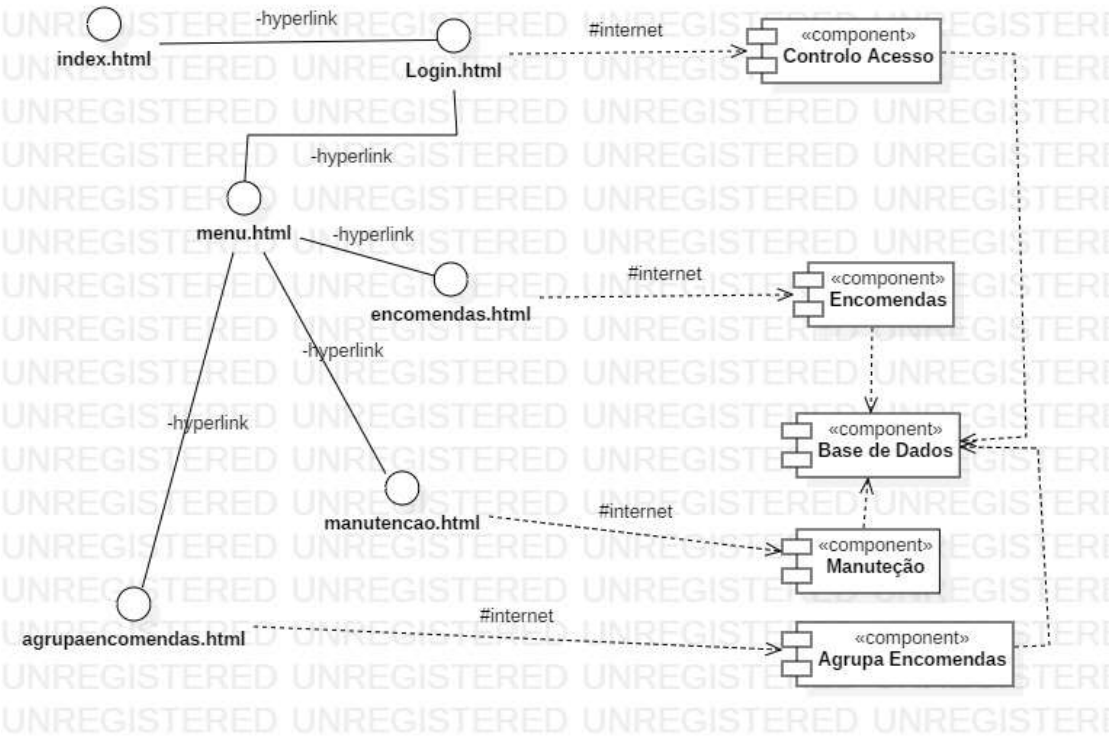

Figura 14. Diagrama de Componentes

3.8 Modelo de Instalação

Este diagrama ilustra a arquitetura do sistema em termos de nós (nodes) que efetuam o processamento de componentes. Na prática, permite demostrar como o hardware estará organizado e como os componentes (software) estarão distribuídos, estabelecendo assim a sua relação física (Nunes & O´Neill, 2011). Na figura 15 apresenta-se o diagrama geral de instalação.

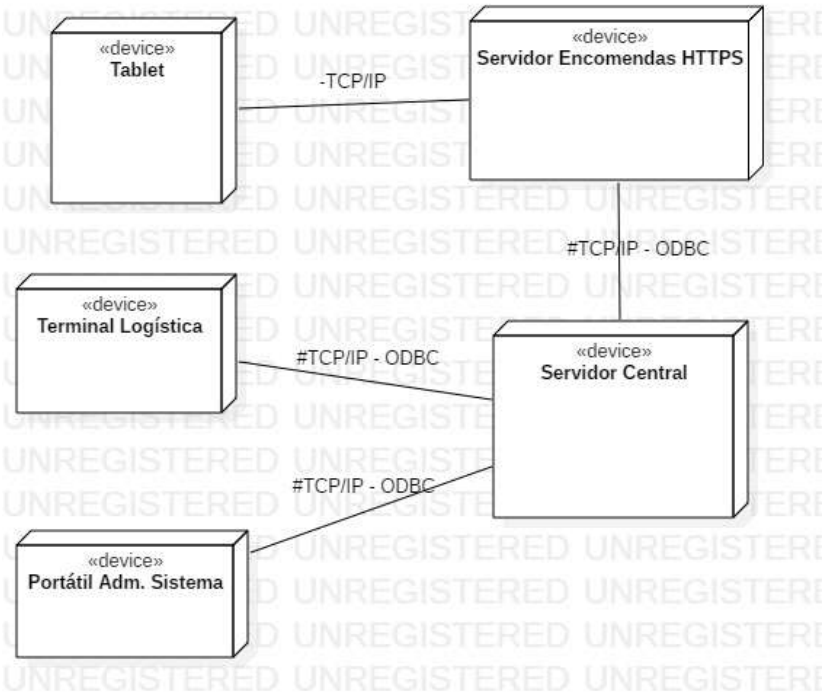

Figura 15. Diagrama geral da Instalação

3.9 Modelo de Dados

Na figura 16 apresenta-se o modelo de dados para dar resposta às necessidades identificadas para a resolução do problema.

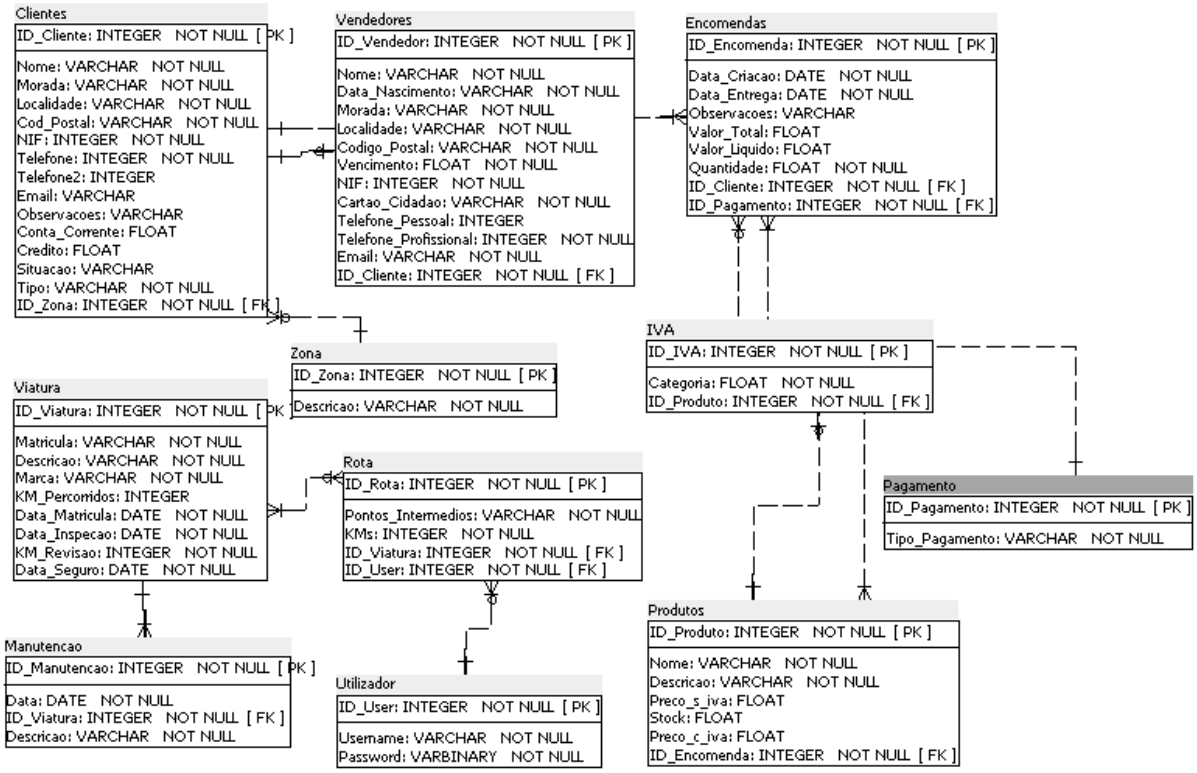

Figura 16. Modelo de Dados (Entidade – Relação)

4. INFRAESTRUTURA DE DESENVOLVIMENTO

Serão utilizadas diversas tecnologias de modo a suportar o processo de criação e desenvolvimento da plataforma que se pretende implementar

O modelo de dados foi desenhado com recurso ao software SQL Power Architect que consiste numa ferramenta que permite desenhar o modelo de base de dados de modo a facilitar a sua criação, pois permite trabalhar em ambiente gráfico de uma forma simples. É um software com versão de utilização gratuita que está disponível para qualquer utilizador (SQLPower Architect, 2019).

A implementação do modelo de dados será concretizada através do sistema de bases de dados relacional Microsoft SQL Server 2014 Standard que incorpora a linguagem Structured Query Language (SQL) (Microsoft, 2014).

Para a criação da plataforma, que ficará online, será utilizada a linguagem de programação HTML que consiste numa linguagem de programação utilizada para a criação de páginas web que permite a criação de documentos que podem ser lidos em praticamente qualquer dispositivo com ligação à internet. Para a criação de documentos em HTML basta um editor de texto e conhecimentos do código que compõe a linguagem de programação HTML. Atualmente o HTML vai na versão HTML5. Funciona de forma simples, sendo que os browsers identificam as tags e apresentam as páginas conforme estas estão especificadas (Sorgetz & Pretto, 2010).

Para a personalização do HTM será utilizado o Cascading Style Sheets (CSS) que é um mecanismo simples para adicionar estilos e personalização a documentos Web, como por exemplo, fontes, cores e espaçamento (W3C, 1994).

Para a ligação à base de dados iremos utilizar a linguagem Hypertext Preprocessor (PHP). O PHP é uma linguagem de script, open source, gratuita, muito utilizada no desenvolvimento de plataformas web, especialmente do lado do servidor e que pode ser incorporada dentro do HTML (PHP, 2001). Irá permitir fazer a ligação entre a plataforma web e a base de dados, onde irão ser geridas as encomendas e as rotas e os dados. O que distingue o PHP de algo como o JavaScript no lado do cliente é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador. O navegador recebe os resultados da execução desse script, mas não sabe qual era o código fonte. Pode-se inclusive configurar o servidor web para processar todos os arquivos HTML com o PHP (PHP, 2001).

5. CONCLUSÕES

A pesquisa efetuada permitiu-nos concluir que existem diversas ferramentas e algoritmos associados à resolução dos problemas da definição de rotas.

Concluímos ainda que devido à complexidade e dificuldade associadas à definição de rotas otimizadas de forma eficiente e eficaz, existe uma grande variedade de algoritmos disponíveis de forma a tentar ajudar a resolução deste problema e que estão divididos por quatro grupos grandes grupos:

- Métodos exatos, onde de forma geral encontram uma solução tida como ideal, mas que demoram demasiado tempo na realização desta operação.
- Heurísticas, que consistem em algoritmos que tendem em encontrar uma solução ideal ou muito perto dela e que elaboram a rota ideal ou a sua procura um pouco pela intuição.
- Meta-heurísticas que tentam evitar os erros descritos nos dois grupos anteriores e procuram encontrar uma solução num tempo razoável.
- Genéticos baseados em procedimentos de seleção natural e de genética.

Do mesmo modo existem diversas tecnologias, proprietárias e open source, que pretendem resolver este tipo de problemas. Contudo os serviços disponíveis para além de terem muitas restrições associadas, também não partilham muita informação sobre as funções que disponibilizam nem como a recolha de dados que efetuam.

Apesar de existirem muitas aproximações à solução do problema a solução que se pretende implementar com este trabalho terá em conta que a forma de facilitar a determinação de uma rota ideal é a prévia identificação das zonas onde as organizações têm clientes com encomendas para entregar, e agrupar esses mesmos clientes num determinado veiculo.

Este processo permitirá que esse veiculo possa efetuar as entregas todas a clientes da mesma zona, evitando assim efetuar rotas por zonas muito dispersas e por onde poderá passar outro veiculo no mesmo dia ou num dia muito próximo, permitindo assim efetuar uma rota mais curta, mas também ela eficaz. A implementação desta solução permitirá acrescentar valor às soluções já existentes e constituirá um contributo para a melhoria da gestão dos processos associados.

REFERÊNCIAS BIBLIOGRÁFICAS

Barbosa, D. F., Jr., C. N., & Kashiwabara, A. Y. (2015). Aplicação da otimização por colônia de formigas ao problema de múltiplos caixeiros viajantes no atendimento de ordens de

serviço nas empresas de distribuição de energia eletrica. *XI Brazilian Symposium on Information System, GO, May 26-29, 2015.*, (pp. 23-30). Goiana

Bonabeau, E., & Meyer, C. (Maio de 2001). *Swarm Intelligence: A Whole New Way to Think About Business.* Obtido de Harvard Business Review: https://hbr.org/2001/05/swarm-intelligence-a-whole-new-way-to-think-about-business

Botassoli, G. T., Furtado, J. C., & Alberti, R. A. (2015). Simulação computacional para otimização de filas em processos. *Revista Geintec Gestão, Inovação e Tecnologia*, 2121-2135.

Carvalho, M. d. (2007). *Aplicações de meta-heuristica genetica e fuzzy no sistema de colonia de formigas para o problema do caixeiro viajante.* S.Paulo: Tese de Mestrado.

Charbonneau, N., & Vokkarane, V. (2010). Tabu Search Meta-Heuristic for Static Manycast Routing and Wavelength Assignment over Wavelength-Routed Optical WDM Networks. *International Conference on Communications.* Cape Town, South Africa: IEEE.

Cunha, C. B. (2000). Aspectos Práticos da Aplicação de Modelos de Roterização de Veículos a Problemas Reais. São Paulo, Brazil: Escola Politécnica da Universidade de São Paulo.

SQLPowerArchitect (2019). Data Modeling & Profiling Tool: SQL Power Architect. Obtido de http://www.bestofbi.com/page/architect: http://www.bestofbi.com/page/ architect

Feo, T. A., & Resende, M. G. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 103-133.

Galvão, R. R. (2017). *Algoritmos de Optimização para o Problema de Roteamento de Veículos.* Limeira: Universidade Estadual de Campinas.

Gheisari, S., Haghighat, A. T., & Saadat, S. (2008). A Simulated Annealing-Based Mobicast Routing Protocol for Wireless Sensor Networks. *3rd International Symposium on Wireless Pervasive Computing* (pp. 529-534). Santorini, Greece: IEEE.

Glover, F. (1989). Tabu Search – Part 1. *ORSA Jornal on Computing*, 190-206.

Gomes, C. G. (2015). *Avaliação de Implementação do SAP ERP na Logística - Estudo de Caso.* Braga: Universidade do Minho.

GoogleMaps. (s.d.). *Conhecer o Google Maps*. Obtido de Google Maps: https://www.google.com/intl/pt-PT/maps/about/

Greem, M. D. (2016). *Scrum - Novice To Ninja.* SITEPOINT PTY LTD.

Guan, C.-h., Cao, Y., & Shi, J. (2010). Tabu Search Algorithm for Solving the Vehicle Routing Problem. *Third International Symposium on Information Processing.* Qingdao, China: IEEE.

Guerreiro, A. F., Almeida, P. A., Relvas, P. I., Povoa, P. P., Monteiro , P. M., & Figueiredo, E. A. (2009). *Construção de uma Metaheurística de Optimização de Rotas de Veículos.* Lisboa: Instituto Superior Técnico.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Oxford University Press*, 97-109.

Huber, S., & Rust, C. (2016). Calculate travel time and distance with OpenStreetMap data using the Open Source Routing Machine (OSRM). *The Stata Journal*, 416-423.

Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., & Nosovic, N. (2012). Dijkstra's shortest path algorithm serial and parallel execution performance analysis. *Proceedings of the 35th International Convention MIPRO* (pp. 1811–1815). Opatija, Croatia: IEEE.

Kawamura, M. S. (2006). *Aplicação Do Método Branch-and-Bound Na Programação De Tarefas Em Uma Única Máquina Com Data De Entrega Comum Sob Penalidades de Adiantamento e Atraso.* São Paulo.

Kunigami. (2010). *Algoritmo de Branch and Cut*. Obtido de Blog do Kunigami doses semanais de computação e matemática:https://kuniga.wordpress.com/2010/10/15/ algoritmo-de-branch-and-cut/

Lysgaard, J. (1997). *Clarke & Wright's Savings Algorithm.* Aarhus: The Aarhus School of Business.

Malaquias, N. G. (2006). *Uso dos algoritmos genéticos para a otimização de rotas de distribuição.* Uberlândia: Universidade Federal de Uberlândia.

Microsoft. (2014). *Microsoft SQL Server*. Obtido de microsoft.com: https://msdn.microsoft.com/pt-br/library/bb545450.aspx

Misa, T. J. (2010). An Interview With Edsger W. Dijkstra. *Communications of the ACM*, 41-47.

Nunes, M., & O´Neill, H. (2011). *Fundamental de UML.* Lisboa: FCA - Editora de Informática, Lda.

Oliveira, H. C., Vasconcelos, G. C., & Alvarenga, G. B. (2006). A Multi-Start Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows. *Ninth Brazilian Symposium on Neural Networks (SBRN'06)* (pp. 1-6). Ribeirao Preto, Brasil: IEEE.

OpenStreetMap. (s.d.). *Sobre Nós*. Obtido de Open Street Map: https://www.openstreetmap.org/about

Pacheco, M. A. (1999). *Algoritmos Genéticos: Principios e Aplicações.* Rio de Janeiro, Brasil : ICA: Laboratório de Inteligência Computacional Aplicada.

Palma-Chilla, L., Lazzarus, J., & Ponce, A. (2011). Gas-solid phase calculations of binary mixtures using optimization of evolutionary algorithms. *Journal of Engineering Thermophysics*, Vol. 20.

Paradiseo. (s.d.). *How to implement your first hill-climber algorithm?* Obtido de Paradiseo a Software Framework for Metaheuristics: http://paradiseo.gforge.inria.fr/index.php?n=Doc.TutoMOLesson1

PHP, T. (2001). *O que é o PHP?* Obtido de http://php.net/: http://php.net/manual/pt_BR/intro-whatis.php

Pinheiro, D. d., Jale, J. d., & de Sousa, P. J. (2010). *Sistemas de Formigas Aplicados ao Problema do Caixeiro Viajante.* Pernambuco: Universidade Federal Rural de Pernambuco.

PrimaveraBSS. (s.d.). *Eyepeak*. Obtido de Primavera Business Software Solutions: https://pt.primaverabss.com/pt/software/solucoes-especializadas/logistica/eyepeak/

Resende, M. G., & FEO, T. A. (1989). A Probabilistic Heuristic for a Computationally Difficult Set Coving Problem. *Operations Research Letters*, 67-71.

SAP. (2017). Master Guide SAP TM 9.5 SP00. *Master Guide for SAP TM*.

Scrum (2018). Obtido de Desenvolvimento Ágil: https://www.desenvolvimentoagil.com.br/scrum/

SCRUMstudy. (2016). *Um Guia para o conhecimento em Scrum.* Phoenix.

Silva, A. M., & Videira, C. A. (2001). *UML, Metodologias e Ferramentas CASE.* V. N. Famalicão: Edições Centro Atlântico.

Silva, B. d. (2013). *Otimização De Rotas Utilizando Abordagens Heurísticas Em Um Ambiente Georreferenciado.* Fortaleza: UNIVERSIDADE ESTADUAL DO CEARÁ.

Soligno, D. (2017). *CLUe Training #3 (Monte Carlo simulations/simulated annealing algorithm)*. Obtido de Universiteit Utrecht: https://www.uu.nl/en/events/clue-training-3-monte-carlo-simulationssimulated-annealing-algorithm

Sorgetz, L., & Pretto, R. (2010). *HTML.*

Souza, L. V. (2014). *GRASP Greedy Randomized Adaptative Search Procedure.* Obtido de SlidePlayer: https://slideplayer.com.br/slide/347822/

Souza, S. S. (2013). *Algoritmo GRASP Eespecializado Aplicado ao Problema de Reconfiguraçãi de Alimentadores em Sistemas de Distribuição Radial.* Ilha Solteira: UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”.

W3C (1994). *https://www.w3.org/Style/CSS/*. Obtido de World Wide Web Consortium (W3C): https://www.w3.org/

Walker, R. (2015). *Traveling Salesman Problem.* Obtido de slideplayer: https://slideplayer.com/slide/8783733/

Wilson, N., & Edgar, C. (2015). Agile Methodology for Modeling and Design of Data Wharehouses. *International Journal of Commputer and Information Enginerring vol.9 n.º 9*, 2132-2137

PERFIL ACADÉMICO E PROFISSIONAL DOS AUTORES

Tiago Duarte é estudante finalista do curso de licenciatura em Informática de Gestão do ISLA Santarém.

Daniel Carvalho é estudante finalista do curso de licenciatura em Informática de Gestão do ISLA Santarém.

Domingos Martinho é Professor Coordenador e Diretor do ISLA Santarém. Licenciado em Informática de Gestão, Mestre em informática, doutorado com especialização em tecnologias da informação e comunicação na educação pela Universidade de Lisboa. Professor Especialista em Ciências Informáticas. Desenvolve investigação nas áreas da aplicação das metodologias e tecnologias de e-learning, aprendizagem baseada em computador e aplicação de business intelligence. Membro do CEPESE e da UI&D do ISLA Santarém onde coordena o núcleo de Informação e Tecnologia.

Endereço postal:
ISLA Santarém
Largo Cândido dos Reis, 2000-241 Santarém
Portugal