Estrutura de Dados Avançadas

Relatório Projeto Semestre 2022/2023

Nome: Victor Destefani

Nº de aluno: a18586

Índice

Introdução3

Visão geral da implementação…4

Descrição das funcionalidades5

Desafios enfrentados6

Testes e validação6

Conclusão6

1. Introdução

O âmbito deste trabalho consiste em responder os conhecimentos adquiridos durante a disciplina Estruturas de Dados Avanças do curso Engenharia de Sistemas Informáticos – IPCA.

O projeto reside no desenvolvimento de uma solução de software na área da micromobilidade. O crescente ecossistema de novas formas de mobilidade social, nomeadamente aquelas que ocorrem entre distâncias curtas, tem promovido a necessária integração de múltiplos meios de deslocação. Esta transformação na forma como a mobilidade é realizada, fator essencial para o desenvolvimento dos espaços, cidades e outros, irá depender de ações que permitam agilizar a utilização dos meios de transporte que suportem uma mobilidade mais fácil, rápida, limpa e económica, como por exemplo os meios de mobilidade elétrica (trotinetes, bicicletas, etc.).

A essência deste projeto prende-se com o desenvolvimento de uma solução de software que permita agilizar a gestão (registo, partilha, utilização) de meios de mobilidade urbana num contexto de uma smart-city. A implementação da solução deverá considerar estruturas de dados dinâmicas, armazenamento em ficheiro, modularização e apresentar uma estruturação e documentação adequadas.

Contemplando dois tipos de utilizadores, a saber, gestores e clientes registados, pretende-se considerar as funcionalidades seguintes:

● Agilizar o aluguer dos meios de mobilidade, disponibilizando: informação sobre os meios disponíveis (tipo, carga da bateria, custo, etc); identificar a localização dos meios existentes através de geocódigo (what3words.com); registar o pagamento através de um saldo recarregável associado ao cliente; entre outros;

● Permitir aos responsáveis pela plataforma (gestores) a gestão dos respetivos meios de mobilidade permitindo registar, alterar, cancelar, localizar e outros serviços;

● Admitir somente clientes registados considerando o NIF, saldo, nome, morada, etc.;

● Permitir aos gestores da plataforma a possibilidade de uma visão integrada permitindo operações como consultas de históricos, estatísticas, validações, etc.;

**Fase 1**:

1. Definição de uma estrutura de dados dinâmica (a validar previamente com o docente), para a representação dos meios de mobilidade elétrica, clientes e gestores;

2. Armazenamento/leitura dos dados em ficheiro de texto (valores de simulação) e binários (preservar dados);

3. Inserção de novos dados (cliente/gestor/meio de mobilidade elétrica);

4. Remoção de determinados dados (cliente/gestor/meio de mobilidade elétrica);

5. Alteração de determinados dados (cliente/gestor/meio de mobilidade elétrica);

6. Registo do aluguer de um determinado meio de mobilidade elétrica;

7. Listagem dos meios de mobilidade elétrica por ordem decrescente de autonomia;

8. Listagem dos meios de mobilidade elétrica existentes numa localização com determinado geocódigo.

**Fase 2**

1. Definição de uma estrutura de dados dinâmica para representação da localização de um conjunto de clientes e meios de mobilidade elétrica, recorrendo a um grafo;

2. Armazenamento/leitura dos dados em ficheiro de texto (valores de simulação) e binários (preservar dados). Dado a localização de um cliente, listar os meios de mobilidade elétrica de um determinado tipo existentes num determinado raio;

3. Calcular uma solução (adaptação do problema do caixeiro-viajante) do trajeto com a menor distância a percorrer, com recurso a um camião, para recolha de todos os meios de mobilidade elétrica com carga da 3 bateria abaixo de 50%. O ponto de partida e chegada é o mesmo. A capacidade de transporte do camião é limitada e dependente dos tipos de meios de mobilidade transportados, pelo que poderá ser necessário efetuar vários circuitos de modo a completar a recolha dos referidos meios de mobilidade elétrica.

Para cada fase do trabalho foi acompanhada dos seguintes critérios:

● Documentação de código fonte, com geração da respetiva API (ex. DoxyGen, DocX, outros).

● Utilização de ferramentas apropriadas para o controlo de versões (Git, GitHub, outras).

● Utilização de ferramentas apropriadas para a gestão do desenvolvimento (Trello, outras).

1. Visão geral da implementação:

A linguagem determinada para realizar o projeto foi a linguagem C e procedeu-se a realização do trabalho utilizando o software *Visual Studio*. A princípio foi muito complicado perceber o funcionamento do software, pois antes de iniciar a compilação, foi preciso realizar algumas configurações internas nosoftware. O projeto teve toda a sua implementação inserida no Github, no final deste documento será inserido o link do repositório Git e também o do Trello. Para cada nova tarefa exercida, optei por criar branchs distintas para saber diferenciar corretamente os pontos do projeto, assim como também foi empregado no Trello as tarefas que precisavam de ser realizadas.

O projeto foi inicializado utilizando arrays, contudo, de acordo como íamos avançando no semestre, aplicamos o projeto utilizando listas ligadas. Também para a gravação de ficheiros em binário, uma solução posta em causa foi a criação de outra struct para armazenar os dados, seja dos *Veiculos*, *Clientes*, *Gestores* e *Aluguéis* como podemos ver a seguir, apenas um exemplo da struct *Veículos*, a mesma ideia foi aplicada para as demais structs.

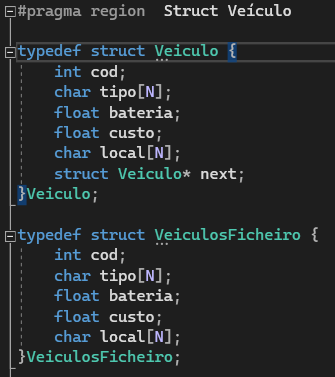


Figura 1 - Exemplo Struct Veículo.

O projeto foi desenvolvido com base no código de aula e fornecidos pelo professor Luís Ferreira, ao longo do percurso foram implementadas funções que seriam extremamente necessárias para todas as estruturas funcionarem de acordo com o que se foi solicitado.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteAssinaturas dos métodos também foram implementadas de acordo com cada estrutura. A utilização de booleano para validar funções também foram implementadas em diversas funções.

Figura - Assinaturas de Funções

1. Descrição das funcionalidades:

Basicamente todas as estruturas carregam as mesmas funções e para cada uma destas funções, foram definidos os seus nomes para melhor leitura e compreensão do código. Dentre as principais estão as funções de *CriaObjeto*, o próprio nome já traduz no sentido da função, a *InsereObjeto,* que insere determinado objeto na lista para posteriormente fazermos a utilização. *VerificaDuplicado* que nada mais verifica se existe um objeto duplicado nas listas, está função foi utilizada como recursividade na função *InsereObjeto,* assim antes de inserir determinado objeto na lista, já será feita a verificação se o objeto existe ou não. Também das principais funcionalidades está o algoritmo DFS (Depth-First Search), que realiza os cálculos de caminhos diretos e caminhos possíveis entre dois vértices, no projeto, foi implementado algumas funções para tanto ser encontrado pelo nome do vértice, quanto ser encontrado pelo seu código de vértice, ou mesmo receber um código de vértice e converter em string e vice-versa.

1. Desafios enfrentados:

O projeto por si foi um enorme desafio, sem muito conhecimento da linguagem, poder ultrapassar situações, como por exemplo, na função *InsereObjeto* inseria normalmente os objetos criados, mas ao inserir o último, era passado como NULL e só carregava lixo em memória. Nessa situação, o uso do depurador foi essencial, pois com muita análise, percebi que estava em falta o return inicio;pois sem isso ela não percebia onde era seu fim. Em termos de biblioteca e perceber o correto funcionamento foi bastante custoso, como era preciso unificar estruturas, por vezes encontrei-me perdido, quando tive de perceber os grafos e adjacências, ao chamar a struct Adj; dentro da struct Vertice; estava a passar o ponteiro Adj\* adjacentes; mas esquecendo de criar esse mesmo parâmetro dentro de struct Adj. Também numa das funções de contar os caminhos, um dos parâmetros de entrada estava a ser passado como int e sendo utilizado como char, o que fazia criar lixo em memória.

1. Testes e validação:

Todos os testes foram feitos utilizando o depurador do software. No início, a validação da criação de objetos, estava a ser feita a verificar manualmente os arquivos bins que eram criados, utilizando o software **Daemon Tools**, para verificar se realmente os dados eram inseridos no ficheiro. Contudo, quando eram criados e inseridos, as primeiras gravações e alterações, no main já não era passado como parâmetro para criar, assim preservando melhor a integridade dos ficheiros que estavam a ser gerados. O sistema precisa de melhorias nas funções de alugar veículos, onde tive maior dificuldade em implementá-las.

1. Conclusão:

O trabalho foi muito gratificante executá-lo e poder entrar de vez no novo mundo da programação, eu concluo que, mesmo não conseguindo executar determinadas funções, eu ultrapassei os meus limites tentando buscar o conhecimento e me sinto bastante realizado, pois no início deste projeto, eu estava totalmente perdido e com intenções de desistir pois estava tudo a ser demasiado complexo. Com o tempo e focando sempre em perceber corretamente o que o professor Luís Ferreira queria transmitir em aula e mesmo com os seus documentos apresentados para estudo, consegui fazer coisas que jamais imaginei que conseguiria fazer e entender. Sempre fui fascinado pela informática, como trabalho na área de help desk, essa cadeira me fez ter ampla visão não só de programação em C, mas sim como funciona por trás o sistema, e isso literalmente me fascinou, poder concluir atividades e perceber o que estava acontecendo naquele momento em determinada função. Os meus agradecimentos ao professor Luís Ferreira pela disponibilidade e orientação para conclusão do projeto.

Link repositório git: [git@github.com:Skunkkzenn/EstruturadeDados.git](mailto:git@github.com:Skunkkzenn/EstruturadeDados.git)

LinkTrello: https://trello.com/invite/b/F3O3dLmH/ATTId9bd0ecd6881e0db029602b73edb33936003E049/project-eda