Documentação de Software

Sistema de Logística de Entrega de Mercadorias (SLEM)

Autores:

Arthur Souza Fernandes

Victor Roberto Chagas Alves

Vinícius dos Santos Gonçalves

Belo Horizonte
Julho de 2025

1. Introdução

1.1. Escopo do software

Este documento detalha a estrutura, funcionalidades e testes do Sistema de Logística de Entrega de Mercadorias (SLEM). O sistema foi desenvolvido como parte da disciplina de Algoritmos e Estruturas de Dados I e abrange o gerenciamento de locais, veículos e pedidos, além de um algoritmo para cálculo de rotas e persistência de dados em arquivos.

1.2. Nome do sistema e de seus componentes principais

- **Nome do Sistema:** Sistema de Logística de Entrega de Mercadorias (SLEM).
- **Componentes Principais:**
  - Módulo de Gerenciamento de Locais
  - Módulo de Gerenciamento de Veículos
  - Módulo de Gerenciamento de Pedidos
  - Módulo de Roteamento
  - Módulo de Persistência de Dados

1.3. Missão ou objetivo do software

O objetivo principal do SLEM é simular um sistema de logística básico, permitindo ao usuário gerenciar as entidades essenciais de uma operação de entrega e calcular a rota mais eficiente para um pedido com base no veículo disponível mais próximo.

1.4. Descrição do domínio do Cliente (Regras de Negócio)

Número	Regra de Negócio	Descrição
1	Identificadores Únicos	Locais (por nome) e Veículos (por placa) não podem ser cadastrados em duplicidade.
2	Dependência de Locais	Um Veículo ou Pedido só pode ser cadastrado se os locais

		associados (local atual, origem, destino) já existirem no sistema.
3	Disponibilidade de Veículos	Apenas veículos com status "Disponível" podem ser alocados para uma nova rota de entrega.
4	Atualização Pós-Entrega	Após uma entrega simulada, o status do veículo é atualizado e sua localização passa a ser o destino final do pedido.
5	Persistência de Dados	Os dados do sistema devem ser salvos ao sair e carregados ao iniciar para garantir a continuidade entre sessões.
6	Cálculo de Rota	A rota é sempre calculada para o veículo disponível que se encontra mais próximo do ponto de coleta do pedido.
7	Distância Euclidiana	A distância entre dois pontos é calculada utilizando a fórmula da distância euclidiana.

1.5. Funcionalidades do produto

Número	**Funcionalidade do sistema**
1	Gerenciamento completo (CRUD) de Locais.
2	Gerenciamento completo (CRUD) de Veículos.
3	Gerenciamento completo (CRUD) de Pedidos.
4	Cálculo e exibição de rota de entrega para um pedido.
5	Backup (salvamento) e restauração de dados em arquivos.

2. Usuários e sistemas externos

2.1. Descrição

Número	Usuários	Definição
1	Operador de Logística	Usuário principal do sistema, responsável por inserir, consultar e gerenciar todos os dados de locais, veículos e pedidos, além de solicitar o cálculo das rotas.

3. Documentação do código

3.1. Documentação da Estrutura de dados geral do software

O sistema armazena todos os seus dados em memória principal através de vetores (arrays) de structs. Foram definidas três structs principais no ficheiro slem.h para representar as entidades do sistema:

- **struct Local**: Armazena os dados de um ponto geográfico, contendo um nome (identificador) e suas coordenadas X e Y.
- **struct Veiculo**: Representa um veículo da frota, com placa (identificador), modelo, seu local atual e um status (Disponível ou Ocupado), definido pelo enum StatusVeiculo.
- **struct Pedido**: Contém as informações de uma solicitação de entrega, incluindo um ID numérico, locais de origem e destino, e o peso da carga.

Todos os vetores são dimensionados pela constante global MAX_ENTIDADES.

3.2. Assinatura das Funções

Função buscarLocalPorNome

int buscarLocalPorNome(const char* nome, const Local locais[], int numLocais);

- **Descrição:** Procura por um local no vetor de locais usando o nome como chave.
- **Parâmetros:** nome (o nome a ser buscado), locais[] (o vetor de locais), numLocais (tamanho atual do vetor).
- **Retorno:** O índice do local no vetor se encontrado; -1 caso contrário.

Função buscarVeiculoPorPlaca

int buscarVeiculoPorPlaca(const char* placa, const Veiculo veiculos[], int numVeiculos);

- **Descrição:** Procura por um veículo no vetor de veículos usando a placa como chave.
- **Parâmetros:** placa (a placa a ser buscada), veiculos[] (o vetor de veículos), numVeiculos (tamanho atual do vetor).
- **Retorno:** O índice do veículo no vetor se encontrado; -1 caso contrário.

Função buscarPedidoPorId

int buscarPedidoPorId(int id, const Pedido pedidos[], int numPedidos);

- **Descrição:** Procura por um pedido no vetor de pedidos usando o ID como chave.
- **Parâmetros:** id (o ID a ser buscado), pedidos[] (o vetor de pedidos), numPedidos (tamanho atual do vetor).
- **Retorno:** O índice do pedido no vetor se encontrado; -1 caso contrário.

Função calcularDistancia

double calcularDistancia(const Local &l1, const Local &l2);

- **Descrição:** Calcula a distância euclidiana em linha reta entre dois locais.
- **Parâmetros:** l1 (struct do primeiro local), l2 (struct do segundo local).
- **Retorno:** Um double representando a distância calculada.

Função salvarDados

void salvarDados(const Local locais[], int numLocais, const Veiculo veiculos[], int numVeiculos, const Pedido pedidos[], int numPedidos);

- **Descrição:** Salva os dados atuais dos vetores de locais, veículos e pedidos em ficheiros binários. Não possui retorno.

Função restaurarDados

void restaurarDados(Local locais[], int &numLocais, Veiculo veiculos[], int &numVeiculos, Pedido pedidos[], int &numPedidos);

- **Descrição:** Carrega os dados dos ficheiros binários para os vetores em memória ao iniciar o programa. Não possui retorno.

4. Testes do software

Os testes foram divididos em **Testes Unitários**, para funções com retornos diretos que podem ser validados de forma isolada, e **Testes de Comportamento**, para funcionalidades que envolvem interação com o usuário ou com o sistema de ficheiros.

4.1. Casos de Teste Unitários

Casos de testes do software: função buscarLocalPorNome

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nome = "Centro"	Nome existe na primeira posição.	Retorna 0.	-	-
2	nome = "Bairro"	Nome existe em outra posição.	Retorna 1.	-	-
3	-	-	-	nome = "Inexistente"	Retorna -1.

Casos de testes do software: função buscarVeiculoPorPlaca

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa = "ABC1234"	Placa existe na primeira posição.	Retorna 0.	-	-
2	placa = "XYZ9876"	Placa existe em outra posição.	Retorna 1.	-	-
3	-	-	-	placa = "ZZZ9999"	Retorna -1.

Casos de testes do software: função buscarPedidoPorId

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id = 1	ID existe na primeira posição.	Retorna 0.	-	-
2	id = 2	ID existe em outra posição.	Retorna 1.	-	-
3	-	-	-	id = 999	Retorna -1.

Casos de testes do software: função calcularDistancia

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	l1 = {0,0}, l2 = {3,4}	Coordenadas formam um triângulo retângulo 3-4-5.	Retorna 5.0.	-	-

4.2. Testes de Comportamento

Testes de Comportamento: Funções salvarDados e restaurarDados

Número do Teste	Ação do Utilizador	Comportamento Esperado do Sistema	Resultado Obtido	Aprovado?
1	1. Iniciar o programa. 2. Cadastrar 2 locais. 3. Sair do programa (opção 0).	O programa deve invocar salvarDados. O ficheiro locais.dat deve ser criado/atualizado na pasta	O ficheiro foi criado com sucesso.	Sim

		dados_bin/.		
2	1. Iniciar novamente o programa após o teste anterior. 2. Listar os locais.	O programa invoca restaurarDados no início. A listagem deve mostrar os 2 locais cadastrados na sessão anterior.	Os dados foram carregados e listados corretamente.	Sim
3	1. Iniciar o programa sem que os ficheiros .dat existam. 2. Listar os locais.	O programa tenta restaurar os dados, mas não encontra os ficheiros. A listagem deve mostrar "Nenhum local cadastrado".	O sistema iniciou corretamente sem dados prévios.	Sim

Testes de Comportamento: Navegação de Menus

Número do Teste	**Ação do Utilizador**	**Comportamento Esperado do Sistema**	**Resultado Obtido**	**Aprovado?**
1	No menu principal, digitar a opção 1.	O sistema deve limpar a tela e exibir o submenu "Gerenciar Locais".	O submenu correto foi exibido.	Sim
2	No menu principal, digitar uma opção inválida (ex: 9).	O sistema deve exibir uma mensagem de "Opção inválida" e mostrar o menu principal novamente.	A mensagem de erro foi exibida e o menu recarregado.	Sim
3	No menu "Gerenciar	O sistema deve retornar ao	O programa retornou ao	Sim

	Locais", digitar a opção 0.	menu principal.	menu principal corretamente.	
4	No menu principal, digitar um caractere não numérico (ex: a).	O sistema deve detetar a entrada inválida, solicitar um número e aguardar nova entrada sem encerrar a execução.	O sistema tratou o erro e pediu nova entrada.	Sim