Documentação do Software

Sistema de Logística de Entrega de Mercadorias

Murilo Freitas de Souza e Bernardo Leão

Belo Horizonte
Julho de 2025

Introdução

Este documento descreve o sistema Sistema de Logística de Entrega de Mercadorias, abordando seu escopo, missão, regras de negócio, funcionalidades, usuários, estrutura de dados, principais funções e casos de teste.

Escopo do software

O sistema gerencia o cadastro e controle de locais, veículos e pedidos para simular e organizar entregas de mercadorias, incluindo cálculo de rotas, backup/restauração de dados e atualização automática do status dos veículos e pedidos.

Nome do sistema e de seus componentes principais

Nome do sistema: Sistema de Logística de Entrega de Mercadorias

Componentes principais:

- Local
- Veiculo
- Pedido
- Rota Menu
- Backup/Restauração

Missão ou objetivo do software

Oferecer uma solução simples para o gerenciamento de entregas, permitindo o cadastro, atualização e rastreamento de locais, veículos e pedidos, além de simular o fluxo de entregas e garantir a integridade dos dados.

Descrição do domínio do Cliente (Regras de Negócio)

Número	Regra de Negócio	Descrição
1	Cadastro único de locais	Não permite locais com nomes duplicados. Cada local possui nome e coordenadas (x, y).
2	Cadastro único de veículos	Não permite veículos com placas duplicadas. Cada veículo possui placa, modelo, status e local atual.
3	Cadastro único de pedidos	Não permite pedidos com IDs duplicados. Cada pedido possui id, origem, destino, peso, placa do veículo e status de entrega.
4	Associação de pedido a veículo	Um pedido só pode ser associado a um veículo disponível. O status do veículo muda para "ocupado".
5	Finalização de entrega	Ao finalizar a entrega, o status do pedido muda para "entregue", o status do veículo volta para "disponível" e o local atual do veículo é atualizado para o destino do pedido.
6	Cálculo de rota	O sistema calcula uma rota ótima para entrega, baseada na menor distância entre os pontos.
7	Backup e restauração	Permite salvar e restaurar os dados de locais, veículos e pedidos em arquivos binários.

Descrição do domínio do Cliente (Regras de Negócio)

Número	Funcionalidades do Sistema
1	Cadastro, listagem, atualização e exclusão de locais
2	Cadastro, listagem, atualização e exclusão de veículos
3	Cadastro, listagem, atualização e exclusão de pedidos
4	Associação de pedidos a veículos e finalização de entregas
5	Backup e restauração dos dados do sistema

Usuários e sistemas externos

Número	Usuários/Sistemas	Definição
1	Operador	Usuário que interage com o sistema via terminal para gerenciar entregas
2	Sistema de arquivos	Utilizado para backup e restauração dos dados
3	Desenvolvedor	Responsável pela manutenção e testes do sistema

Documentação do código

Estrutura de dados geral do software

Locais: Vetor fixo de objetos Local (Local locais[MAX_LOCAIS]), cada um com nome, x, y.

Veículos: std::vector<Veiculo>, cada um com placa, modelo, status, localAtual.

Pedidos: std::vector<Pedido>, cada um com id, nomeOrigem, nomeDestino, peso, placaVeiculo, entregue.

Funções de Locais

Função **cadastrarLocal**

```
bool cadastrarLocal(const std::string& nome, float x, float y); // Parâmetros:
//nome: nome do local (std::string)
//x: coordenada X (float)
//y: coordenada Y (float)
// Retorno: true se o local foi cadastrado com sucesso, false caso contrário
(ex: nome duplicado ou limite atingido)
```

Função **listarLocais**

```
bool cadastrarLocal(const std::string& nome, float x, float y); // Parâmetros:
//nome: nome do local (std::string)
//x: coordenada X (float)
//y: coordenada Y (float)
// Retorno: true se o local foi cadastrado com sucesso, false caso contrário
(ex: nome duplicado ou limite atingido)
```

Função **atualizarLocal**

```
bool cadastrarLocal(const std::string& nome, float x, float y); // Parâmetros:
//nome: nome do local (std::string)
//x: coordenada X (float)
//y: coordenada Y (float)
// Retorno: true se o local foi cadastrado com sucesso, false caso contrário
(ex: nome duplicado ou limite atingido)
```

Função **excluirLocal**

```
bool cadastrarLocal(const std::string& nome, float x, float y); // Parâmetros:
//nome: nome do local (std::string)
//x: coordenada X (float)
//y: coordenada Y (float)
// Retorno: true se o local foi cadastrado com sucesso, false caso contrário
(ex: nome duplicado ou limite atingido)
```

Função calcularDistanciaEntreLocais

```
float calcularDistanciaEntreLocais(const Local& l1, const Local& l2);
// Parâmetros:
//  l1: objeto Local origem
//  l2: objeto Local destino
// Retorno: distância euclidiana entre os dois locais (float)
```

Funções de Veículos

Função **cadastrarVeiculo**

```
bool cadastrarVeiculo(std::string placa, std::string modelo, std::string status,
std::string localAtual);
// Parâmetros:
//   placa: placa do veículo (std::string)
//   modelo: modelo do veículo (std::string)
//   status: status do veículo ("disponivel" ou "ocupado") (std::string)
//   localAtual: nome do local atual do veículo (std::string)
// Retorno: true se o veículo foi cadastrado com sucesso, false caso
contrário (ex: placa duplicada)
```

Função **listarVeiculo**

```
void listarVeiculos();
// Parâmetros: nenhum
// Retorno: nenhum (imprime a lista de veículos cadastrados)
```

Função **atualizarVeiculo**

```
bool atualizarVeiculo(std::string placa, std::string novoModelo, std::string
novoStatus, std::string novoLocal);
// Parâmetros:
//   placa: placa do veículo a ser atualizado (std::string)
//   novoModelo: novo modelo (std::string)
//   novoStatus: novo status (std::string)
//   novoLocal: novo local atual (std::string)
// Retorno: true se atualizado com sucesso, false caso contrário
```

Função **excluirVeiculo**

```
bool excluirVeiculo(std::string placa);
// Parâmetros: //   placa: placa do veículo a ser excluído (std::string)
// Retorno: true se excluído com sucesso, false caso contrário
```

Funções de Pedidos

Função **cadastrarPedido**

```
bool cadastrarPedido(int id, const std::string& origem, const std::string&
destino, float peso);
// Parâmetros:
//   id: identificador do pedido (int)
//   origem: nome do local de origem (std::string)
//   destino: nome do local de destino (std::string)
//   peso: peso do pedido (float) // Retorno: true se o pedido foi cadastrado
com sucesso, false caso contrário (ex: id duplicado)
```

Função **listarPedidos**

```
void listarPedidos();
// Parâmetros: nenhum
// Retorno: nenhum (imprime a lista de pedidos cadastrados)
```

Função **atualizarPedido**

```
bool atualizarPedido(int id, const std::string& novaOrigem, const
std::string& novoDestino, float novoPeso);
// Parâmetros:
//   id: identificador do pedido (int)
//   novaOrigem: novo local de origem (std::string)
//   novoDestino: novo local de destino (std::string)
//   novoPeso: novo peso (float)
// Retorno: true se atualizado com sucesso, false caso contrário
```

Função **excluirPedido**

```
bool excluirPedido(int id);
// Parâmetros:
//   id: identificador do pedido a ser excluído (int)
// Retorno: true se excluído com sucesso, false caso contrário
```

Função **associarPedidoVeiculo**

```
bool associarPedidoVeiculo(int idPedido, const std::string& placaVeiculo);
// Parâmetros:
//   idPedido: identificador do pedido (int)
//   placaVeiculo: placa do veículo (std::string)
//Retorno: true se o pedido foi associado com sucesso, false caso contrário
// Retorno: bool (true se associado, false se erro)
```

Função **finalizarEntrega**

```
bool finalizarEntrega(int idPedido, const std::string& placaVeiculo);
// Parâmetros:
//   idPedido: identificador do pedido (int)
//   placaVeiculo: placa do veículo (std::string)
// Retorno: true se a entrega foi finalizada com sucesso, false caso
contrário
```

Funções de Rotas

Função **calcularRotaEntrega**

```
std::vector<std::string> calcularRotaEntrega(const std::vector<std::string>&
nomesLocais);
// Parâmetros:
//   nomesLocais: vetor de nomes dos locais a serem visitados
(std::vector<std::string>)
// Retorno: vetor de nomes dos locais na ordem ótima de entrega
(std::vector<std::string>)
```

Função **calcularDistanciaEntreLocaisPorNome**

```
float calcularDistanciaEntreLocaisPorNome(const std::string& nome1, const
std::string& nome2);
// Parâmetros:
//   nome1: nome do local de origem (std::string)
//   nome2: nome do local de destino (std::string)
// Retorno: distância euclidiana entre os dois locais (float)
```

Teste de Software

Casos de Teste

Função: **cadastrarLocal**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nome, x, y	"Hospital", 12.3, 45.6	true	"Hospital", 13.0, 50.0 (nome já existe)	false
2	nome, x, y	"Supermercado", -8.0, 22.1	true	" ", 1.0, 1.0 (nome vazio)	false
3	nome, x, y	"Escola", 0.0, -15.0	true	"Shopping", 99999, 99999 (fora do mapa)	false

Função: **atualizarLocal**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nomeAntigo, nomeNovo, x, y	"Hospital", "Hospital Central", 13, 46	true	"Padaria", "Padaria Nova", 1, 1 (nomeAntigo não existe)	false
2	nomeAntigo, nomeNovo, x, y	"Supermercado", "Mercado 24h", 5, 20	true	"Hospital", "Mercado 24h", 5, 20 (nomeNovo já existe)	false
3	nomeAntigo, nomeNovo, x, y	"Escola", "Escola", 1, 2	true	"Escola", "", 1, 2 (nomeNovo vazio)	false

Função: **excluirLocal**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nome	"Hospital"	true	"Posto de Gasolina"	false
2	nome	"Supermercado"	true	"" (nome vazio)	false

Função: **cadastrarVeículo**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa, modelo, status, localAtual	"QWE9876", "Sprinter", "disponivel", "Hospital"	true	"QWE9876", "Ducato", "disponivel", "Hospital" (placa duplicada)	false
2	placa, modelo, status, localAtual	"JKL4321", "HR", "ocupado", "Supermercado"	true	"", "HR", "disponivel", "Supermercado" (placa vazia)	false
3	placa, modelo, status, localAtual	"MNO2468", "Fiorino", "disponivel", "Escola"	true	"MNO2468", "Fiorino", "em uso", "Escola" (status inválido)	false

Função: **atualizarVeículo**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa, novoModelo, novoStatus, novoLocal	"QWE9876", "Sprinter Nova", "ocupado", "Hospital Central"	true	"ZZZ0000", "X", "X", "X" (placa não existe)	false
2	placa, novoModelo, novoStatus, novoLocal	"JKL4321", "HR Nova", "disponivel", "Mercado 24h"	true	"QWE9876", "Uno", "em uso", "Hospital" (status inválido)	false

Função: **excluirVeículo**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa	"QWE9876"	true	"ZZZ0000" (placa não existe)	false
2	placa	"JKL4321"	true	"" (placa vazia)	false

Função: **cadastrarPedido**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id, origem, destino, peso	101, "Hospital", "Supermercado", 12.5	true	101, "Hospital", "Supermercado", 5.0 (id já existe)	false
2	id, origem, destino, peso	102, "Supermercado", "Escola", 8.0	true	103, "", "Escola", 2.0 (origem vazia)	false
3	id, origem, destino, peso	103, "Escola", "Hospital", 1.0	true	104, "Escola", "", 1.0 (destino vazio)	false
4	id, origem, destino, peso	104, "Hospital", "Escola", 0.5	true	105, "Hospital", "Escola", -1.0 (peso negativo)	false

Função: **atualizarPedido**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id, novaOrigem, novoDestino, novoPeso	101, "Hospital Central", "Escola", 10.0	true	999, "A", "B", 1.0 (id não existe)	false
2	id, novaOrigem, novoDestino, novoPeso	102, "Mercado 24h", "Hospital", 5.0	true	101, "", "B", 1.0 (origem vazia)	false

Função: **excluirPedido**()

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id	101	true	999 (id não existe)	false
2	id	102	true	-1 (id inválido)	false

Função: **mostrarDistanciaPedido**( )

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id	101 (pedido existente, locais válidos)	imprime distância correta	999 (pedido não existe)	imprime "Pedido não encontrado."
2	id	103 (origem ou destino inexistente)	imprime "Origem ou destino não encontrados."	-1 (id negativo)	imprime "Pedido não encontrado."

Função: **associarPedidoVeiculo**( )

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	idPedido, placaVeiculo	101, "QWE9876"	true	999, "QWE9876" (pedido inexistente)	false
2	idPedido, placaVeiculo	102, "ZZZ0000"	false	102, "" (placa vazia)	false

Função: **calcularDistanciaEntreLocais**( )

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	l1, l2	("Hospital", 12.3, 45.6), ("Supermercado", -8.0, 22.1)	valor calculado	("Hospital", 12.3, 45.6), ("Hospital", 12.3, 45.6)	0.0
2	l1, l2	("Escola", 0.0, -15.0), ("Hospital", 12.3, 45.6)	valor calculado	("Escola", 0.0, -15.0), ("Shopping", 99999, 99999)	-1.0

Função: **finalizarEntrega**( )

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	idPedido, placaVeiculo	101, "QWE9876"	true	101, "ZZZ0000" (veículo não existe)	false
2	idPedido, placaVeiculo	999, "QWE9876"	false	101, "QWE9876" (pedido já entregue)	false

Função: **calcularRotaEntrega( )**

Nº	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nomesLocais	{"Hospital", "Supermercado", "Escola"}	Ordem ótima	{} (vetor vazio)	vetor vazio
2	nomesLocais	{"Hospital", "Supermercado", "Escola", "Mercado 24h"}	Ordem ótima	{"Hospital", "Aeroporto"} (Aeroporto não existe)	Ordem parcial ou erro