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 testes do software:

Função**: cadastrarLocal**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	nome, x, y	"A", 0, 0	true	"A", 1, 1 (nome já existe)	false
2	nome, x, y	"B", 2, 3	true	" ", 1, 1 (nome vazio)	false
3	nome, x, y	"C", -5, 10	true	"D", 100000, 100000 (fora do limite do mapa)	false

Função**: atualizarLocal**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	nomeAntigo, nomeNovo, x, y	"A", "A1", 1, 1	true	"Z", "Z1", 1, 1 (nomeAntigo não existe)	false
2	nomeAntigo, nomeNovo, x, y	"B", "B2", 2, 2	true	"A", "B2", 2, 2 (nomeNovo já existe)	false
3	nomeAntigo, nomeNovo, x, y	"C", "C", -5, 10	true	"C", "", 1, 1 (nomeNovo vazio)	false

Função**: excluirLocal**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	nome	"A"	true	"Z" (nome não existe)	false
2	nome	"B"	true	"" (nome vazio)	false

Função: **cadastrarVeículo**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa, modelo, status, localAtual	"ABC1234", "Fiorino", "disponivel", "A"	true	"ABC1234", "Uno", "disponivel", "A" (placa duplicada)	false
2	placa, modelo, status, localAtual	"XYZ5678", "Kombi", "ocupado", "B"	true	"", "Kombi", "disponivel", "B" (placa vazia)	false
3	placa, modelo, status, localAtual	"QWE1234", "Van", "disponivel", "C"	true	"QWE1234", "Van", "em uso", "C" (status inválido)	false

Função: **atualizarVeículo**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa, novoModelo, novoStatus, novoLocal	"ABC1234", "Fiorino Novo", "ocupado", "Bairro A"	true	"ZZZ9999", "X", "X", "X" (placa não existe)	false
2	placa, novoModelo, novoStatus, novoLocal	"XYZ5678", "Kombi Nova", "disponivel", "Centro"	true	"ABC1234", "Uno", "em uso", "Centro" (status in	false

Função: **excluirVeículo**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	placa	"ABC1234"	true	"ZZZ9999" (placa não existe)	false
2	placa	"XYZ5678"	true	"" (placa vazia)	false

Função: **cadastrarPedido**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id, origem, destino, peso	1, "A", "B", 10.0	true	99, "A", "B", 1.0 (id não existe)	false
2	id, origem, destino, peso	2, "B", "C", 8.0	true	1, "", "B", 1.0 (origem vazia)	false
3	id, origem, destino, peso	3, "C", "D", 1.0	true	5, "A", "C", -1.0 (peso negativo)	
4	id, origem, destino, peso	4, "A", "C", 0.5	true	5, "A", "C", -1.0 (peso negativo)	

Função: **atualizarPedido**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	id,novaOrigem, novoDestino, novoPeso	1, "Centro", "Bairro B", 10.0	true	99, "A", "B", 1.0 (id não existe)	false
2	id, novaOrigem, novoDestino, novoPeso	2, "Bairro A", "Centro", 5.0	true	1, "", "B", 1.0 (origem vazia)	false

Função: **excluirPedido**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	id	1	true	99 (id não existe)	false
2	id	2	true	-1 (id inválido)	false

Função: **mostrarDistanciaPedido**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	id	1 (pedido existente, locais válidos)	imprime distância correta	99 (pedido não existe)	false
2	id	2 (pedido com origem/destino inexistente)	imprime "Origem ou destino não encontrado."	-1 (id negativo)	imprime "Pedido não encontrado."

Função: **associarPedidoVeiculo**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	idPedido, placaVeiculo	1, "ABC1234"	true	99, "ABC1234" (pedido inexistente)	false
2	placa, modelo, status, localAtual	1, "ZZZ9999"	false	1, "" (placa vazia)	false

Função: **calcularDistanciaEntreLocais**

Número	Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
1	l1, l2	(0,0), (3,4)	5.0	(0,0), (0,0)	0.0
2	l1, l2	(1,2), (4,6)	5.0	(1,2), (999,999) (local inexistente)	-1.0

Função: **finalizarEntrega**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	idPedido, placaVeiculo	1, "ABC1234"	true	1, "ZZZ9999" (veículo que não existe)	false
2	idPedido, placaVeiculo	99, "ABC1234"	false	1, "ABC1234" (pedido já entregue)	false

Função: **calcularRotaEntrega**

Número	**Variáveis de Entrada**	**Valores Válidos**	**Resultado Esperado**	**Valores Inválidos**	**Resultado Esperado**
1	nomesLocais	{"A", "B", "C"}	Ordem ótima (ex: A,B,C)	{} (vetor vazio)	vetor vazio
2	nomesLocais	{"A", "B", "C", "D"}	Ordem ótima (ex: A,B,C,D)	{"A", "Z"} (Z não existe)	Ordem parcial ou erro