Documentação do Software

SISTEMA DE LOGÍSTICA DE ENTREGA DE MERCADORIAS (SLEM)

Autores:
Cleverson Resende Rosa
Cauã Thomarco Thomaz Teixeira
Gabriel Junio Nunes Viana

Belo Horizonte
2025.

Documentação do Software

Introdução

Escopo do software

Nome do sistema e de seus componentes principais

SLEM (Sistema Logístico de Entrega e Monitoramento)

Sistema de gerenciamento logístico para controle de:

- Locais (depósitos, centros de distribuição)
- Frota de veículos
- Pedidos de entrega
- Rotas de distribuição

Componentes principais:

- Módulo de Cadastro (Locais, Veículos, Pedidos)
- Módulo de Roteirização
- Módulo de Backup/Restauração
- Interface de Menu

Missão/objetivo do software

Automatizar e otimizar o processo de gestão de entregas, proporcionando:

- Controle centralizado de recursos logísticos
- Cálculo de rotas
- Persistência de dados
- Disponibilidade de veículos/pedidos

Descrição do domínio do Cliente (Regras de Negócio)

Número	*Regra de Negócio*	*Descrição*
1	Rotas são calculadas pelo veículo mais próximo.	As rotas devem ser calculadas usando a distância euclidiana entre dois pontos pegando o veículo mais próximo.
2	Backup de dados em arquivos binários	O usuário deverá poder fazer o backup dos dados e salvá-los em arquivos binários.
3	Veículos devem ter atualização de status entre disponível/ocupado	Os veículos devem poder ter seus status atualizados para "ocupado" quando estão em rotas e quando a entrega for finalizada o status deve retornar a "disponível".
4	Pedidos devem conter a atualização de status entre pendente/entregue	Os pedidos devem conter ter seus status atualizado para entregue quando uma rota for concluída.
5	Pedido não pode ser editado depois de status atualizado para entregue	Um pedido não pode ser atualizado depois que ele for entregue.

6	Atualização do local do veículo	O local do veículo deverá ser atualizado para o local do destino do pedido após a conclusão da entrega.
7	O sistema deve fornecer opção de restauração de dados	O sistema deverá fornecer uma opção no menu para restaurar os dados salvos em backup quando o sistema for reiniciado.

Funcionalidades do produto

Número	*Funcionalidade do sistema*
1	CRUD de veículos, locais e pedidos
2	Backup de dados
3	Restauração de dados
4	Calcula de rota de entrega
5	Atualização de status de veículos e pedidos

Usuários e sistemas externos

Descrição

Número	*Usuários*	*Definição*
1	Gerentes logísticos	Responsáveis pela gestão global do sistema, incluindo cadastro de locais, monitoramento de entregas e análise de relatórios. Têm acesso a todas as funcionalidades.
2	Operadores de frota	Encarregados do gerenciamento diário dos veículos (cadastro, edição, alocação). Podem calcular rotas e acompanhar entregas em andamento.
3	Atendentes de pedidos	Responsáveis pelo registro e atualização de pedidos. Têm acesso limitado às funções de cadastro e consulta de pedidos.

Documentação do código

Documentação da Estrutura de dados geral do software

class Local

Armazena informações geográficas dos pontos logísticos
Estrutura:
nome[100]: char → Nome do local (máx. 99 caracteres)
coordenadaX: float → Posição no eixo horizontal
coordenadaY: float → Posição no eixo vertical

class Veiculo

Gerencia os veículos da frota
Estrutura:
placa[20]: char → Identificação única do veículo
modelo[50]: char → Descrição do modelo
disponivel: bool → Status operacional
idLocalAtual: int → Referência ao local onde está estacionado

class Pedido

Registra as solicitações de entrega
Estrutura:
id: int → Identificador único
idOrigem: int → Referência ao local de coleta
idDestino: int → Referência ao local de entrega
peso: float → Massa da carga em kg
entregue: bool → Status de conclusão

class SistemaLogistica

Gerencia todas as operações do sistema
Estruturas de armazenamento:
locais: vector<Local> → Lista dinâmica de pontos logísticos
veiculos: vector<Veiculo> → Registro da frota disponível
pedidos: vector<Pedido> → Todas as solicitações de entrega
proximoIdPedido: int → Contador para IDs únicos

Função

LogisticaException(const char* msg)

- Parâmetros: msg - mensagem de erro a ser armazenada
- Comportamento: Construtor que inicializa a mensagem de erro
- Retorno: N/A

const char* what() const noexcept override

- Parâmetros: Nenhum
- Comportamento: Retorna a mensagem de erro armazenada
- Retorno: Ponteiro para a mensagem de erro

void adicionarLocal(const Local& local)

- **Parâmetros**: local - Objeto Local contendo dados a serem adicionados
- **Comportamento:**

    - Verifica se nome já existe.
    - Adiciona novo local ao vetor locais
    - Lança exceção se nome duplicado
- **Retorno:** N/A

void editarLocal(int id, const char* novoNome, float x, float y)

- **Parâmetros:**
    - id - Índice do local a editar
    - novoNome - Novo nome para o local
    - x, y - Novas coordenadas
- **Comportamento:**
    - Valida ID
    - Atualiza dados do local
    - Lança exceção se ID inválido
- **Retorno:** N/A

void removerLocal(int id)

- **Parâmetros:**
- id - Índice do local a remover
- **Comportamento:**
    - Valida ID
    - Remove local do vetor
    - Lança exceção se ID inválido
- **Retorno:** N/A

void listarLocais() const

- **Parâmetros:** Nenhum
- **Comportamento:** Exibe todos os locais cadastrados ou mensagem de vazio
- **Retorno:** N/A

void adicionarVeiculo(const Veiculo& veiculo)

- **Parâmetros:**
    - veiculo - Objeto Veiculo a ser adicionado
- **Comportamento:**
    - Verifica placa duplicada
    - Adiciona ao vetor veiculos
    - Lança exceção se placa existir
- **Retorno:** N/A

void editarVeiculo(int id, const char* novaPlaca, const char* novoModelo, int novoLocal)

- **Parâmetros:**
    - id - Índice do veículo
    - novaPlaca - Nova identificação
    - novoModelo - Novo modelo

    - novoLocal - Novo local de estacionamento
- **Comportamento:**
    - Valida ID e placa única
    - Atualiza dados do veículo
    - Lança exceções para erros
- **Retorno:** N/A

void removerVeiculo(int id)

- **Parâmetros:**
    - id - Índice do veículo a remover
- **Comportamento:**
    - Verifica disponibilidade
    - Remove do vetor
    - Lança exceção se em uso ou ID inválido
- **Retorno:** N/A

void listarVeiculos() const

- **Parâmetros:** Nenhum
- **Comportamento:** Exibe frota cadastrada ou mensagem de vazio
- **Retorno:** N/A

void adicionarPedido(int origem, int destino, float peso)

- **Parâmetros:**
    - origem, destino - IDs de locais
    - peso - Massa da carga (kg)
- **Comportamento:**
    - Valida locais e peso
    - Cria pedido com ID sequencial
    - Lança exceção para dados inválidos
- **Retorno:** N/A

void editarPedido(int id, int novaOrigem, int novoDestino, float novoPeso)

- **Parâmetros:**
    - id - ID do pedido
    - novaOrigem, novoDestino - Novos locais
    - novoPeso - Novo peso
- **Comportamento:**
    - Valida ID, status e novos dados
    - Atualiza pedido
    - Lança exceções para restrições
- **Retorno:** N/A

void removerPedido(int id)

- **Parâmetros:**

    - id - ID do pedido
- **Comportamento:**
    - Verifica se não foi entregue
    - Remove do vetor
    - Lança exceção para restrições
- **Retorno:** N/A

void listarPedidos() const

- **Parâmetros:** Nenhum
- **Comportamento:** Exibe todos os pedidos ou mensagem de vazio
- **Retorno:** N/A

void calcularRotaEntrega(int idPedido)

- **Parâmetros:**
    - idPedido - ID do pedido a ser atendido
- **Comportamento:**
    - Encontra veículo mais próximo disponível
    - Calcula distâncias
    - Atualiza status do veículo
    - Registrar pedido em rota
    - Exibe detalhes da rota
- **Retorno:** N/A

void listarEntregasEmRota()

- **Parâmetros**: Nenhum
- **Comportamento:**
    - Mostra pedidos em andamento
    - Permite confirmar/cancelar entregas
    - Atualiza status conforme ação
- **Retorno:** N/A

void fazerBackup() const

- **Parâmetros:** Nenhum
- **Comportamento:**
    - Salva dados em arquivos binários:
    - locais.bin, veiculos.bin, pedidos.bin, id_pedido.bin
    - Lança exceção em falha de I/O
- **Retorno:** N/A

void restaurarDados()

- **Parâmetros:** Nenhum
- **Comportamento:**
    - Carrega dados dos arquivos de backup
    - Recupera estado anterior do sistema

- Lança exceção se arquivos corrompidos
- **Retorno:** N/A

float calcularDistancia(int idLocal1, int idLocal2) const

- **Parâmetros:**
  - idLocal1, idLocal2 - IDs dos locais
- **Comportamento:**
  - Calcula distância euclidiana entre coordenadas
  - Valida IDs dos locais
  - Lança exceção para locais inválidos
- **Retorno:** Distância calculada (float)

Testes do software

Número	*Função*
CT001	void adicionarLocal(const Local& local)
CT002	void editarLocal(int id, const char* novoNome, float x, float y)
CT003	void removerLocal(int id)
CT004	void adicionarVeiculo(const Veiculo& veiculo)
CT005	void editarVeiculo(int id, const char* novaPlaca, const char* novoModelo, int novoLocal)
CT006	void removerVeiculo(int id)
CT007	void adicionarPedido(int origem, int destino, float peso)
CT008	void editarPedido(int id, int novaOrigem, int novoDestino, float novoPeso)
CT009	void calcularRotaEntrega(int idPedido)
CT010	void listarEntregasEmRota()
CT011	void fazerBackup() const
CT012	void restaurarDados()
CT013	void listarLocais() const, void listarVeiculos() const, void listarPedidos() const

CT014	void exibirMenu(), void menuLocais(), void menuVeiculos(), void menuPedidos()
CT015	float calcularDistancia(int idLocal1, int idLocal2) const
CT016	void adicionarPedido(int origem, int destino, float peso)
CT017	void adicionarVeiculo(const Veiculo&), void editarVeiculo(...)
CT018	void fazerBackup() const, void restaurarDados()
CT019	void listarEntregasEmRota()
CT020	void editar*() (Local, Veículo, Pedido)

Casos de testes do software

Número	*Varáveis de Entrada*	*Valores válidos*	*Resultado Esperado*	*Valores inválidos*	*Resultado Esperado*
CT001	Local: Nome "Depósito A", X=10.5, Y=20.3	Nome único, coordenadas numéricas	Local adicionado com sucesso	Nome já existente, coordenadas não numéricas	Erro: "Local com este nome já existe" ou "Entrada inválida para coordenadas"
CT002	Editar Local: ID existente, novo nome "Depósito B", X=15.0, Y=25.0	ID válido, nome único, coordenadas válidas	Local atualizado com sucesso	ID inválido, nome duplicado	Erro: "ID de local inválido" ou "Local com este nome já existe"
CT003	Remover Local: ID existente sem dependências	ID válido, local não referenciado por veículos ou pedidos	Local removido com sucesso	ID inválido, local em uso por veículo/pedido	Erro: "ID de local inválido" ou "Local está em uso"
CT004	Veículo: Placa "ABC1234", Modelo "Fiat Ducato", Local ID 0	Placa única, modelo válido, local ID existente	Veículo adicionado com sucesso	Placa duplicada, local ID inválido	Erro: "Veículo com esta placa já existe" ou "ID de local inválido"
CT005	Editar Veículo: ID existente, nova placa "XYZ5678", novo modelo, local	ID válido, placa única (se alterada), local existente	Veículo atualizado com sucesso	Placa duplicada, veículo em uso	Erro: "Placa já existe" ou "Não é possível editar veículo em uso"
CT006	Remover Veículo: ID existente disponível	ID válido, veículo disponível	Veículo removido com sucesso	ID inválido, veículo em uso	Erro: "ID de veículo inválido" ou "Não é possível remover veículo em uso"
CT007	Pedido: Origem ID 0, Destino ID 1, Peso 5.5kg	IDs locais existentes, peso positivo	Pedido criado com ID sequencial	IDs locais inválidos, peso negativo/zero	Erro: "Local de origem/destino inválido" ou "Peso deve ser positivo"
CT008	Editar Pedido: ID existente não entregue, novos dados	ID válido, pedido não entregue, novos dados válidos	Pedido atualizado com sucesso	ID inválido, pedido já entregue	Erro: "ID de pedido inválido" ou "Não é possível editar pedido já entregue"
CT009	Calcular Rota: Pedido ID 1 existente não entregue	ID válido, pedido não entregue, veículo disponível	Rota calculada com veículo designado	ID inválido, pedido já entregue, sem veículos disponíveis	Erro: "Pedido não encontrado" ou "Nenhum veículo disponível"
CT010	Confirmar Entrega: Após cálculo de rota	Pedido em transporte, veículo designado	Pedido marcado como entregue, veículo liberado	Tentativa de confirmação sem cálculo prévio	Erro: Operação não disponível
CT011	Backup: Sistema com dados	Arquivos podem ser	Binários gerados com	Permissões de arquivo	Erro: "Falha ao criar arquivo de backup"

		criados	sucesso	insuficientes	
CT012	Restauração: Arquivos de backup válidos	Arquivos existentes e íntegros	Dados restaurados com sucesso	Arquivos corrompidos ou inexistentes	Erro: "Falha ao abrir arquivo de backup"
CT013	Listagens: Locais/Veículos/Pedidos	Dados existentes ou vazios	Lista exibida corretamente ou mensagem "Nenhum X cadastrado"	-	-
CT014	Menu: Opções numéricas válidas	Números de 0-9 conforme menu	Ação correspondente executada	Caracteres não numéricos, números fora do range	Mensagem: "Opção inválida!"
CT015	Distância: Cálculo entre dois locais válidos	IDs de locais existentes	Distância calculada corretamente	IDs inválidos	Erro: "Local inválido para cálculo de distância"
CT016	Pedido com peso extremo: 0.001kg	Peso positivo mínimo	Pedido criado com sucesso	Peso zero ou negativo	Erro: "Peso deve ser positivo"
CT017	Veículo em local inexistente	-	-	Local ID inválido	Erro ao listar/criar rota
CT018	Backup/Restauração com dados máximos	Grande volume de dados	Todos dados preservados	-	-
CT019	Cancelamento de rota após cálculo	Rota calculada não confirmada	Veículo volta a disponível	-	-
CT020	Edição para dados idênticos	Novos valores iguais aos atuais	Operação concluída sem erro	-	-