LogiTrack: Sistema de Gestão de Transportes

Documentação Técnica

Versão: 1.0

Data: 15 de Junho de 2025

Desenvolvido por: Estudante de Desenvolvimento de Sistemas

Índice

---

1. Visão Geral do Sistema

O **LogiTrack** é um sistema completo de gestão para transportadoras que integra:

- **Sistema CRUD** para gerenciamento de dados (peças, fornecedores, caminhões, funcionários, clientes e saídas)
- **Módulo de Sensores IoT** para monitoramento de segurança e ambiente
- **Sistema de Controle de Luzes** com comunicação serial para Arduino

- **Interface Gráfica Moderna** desenvolvida em CustomTkinter

Objetivos do Sistema

- Facilitar o gerenciamento operacional da transportadora
- Automatizar o controle de segurança através de sensores
- Permitir controle remoto da iluminação dos setores
- Manter registros organizados em arquivos de texto
- Fornecer interface intuitiva e moderna

---

2. Arquitetura do Sistema

O sistema segue uma arquitetura modular com separação clara entre:

Camadas do Sistema

1. **Camada de Apresentação (View)**
2. Interface gráfica em CustomTkinter
3. Telas específicas para cada módulo
4. Navegação intuitiva com sidebar
5. **Camada de Lógica de Negócio (Controller)**
6. Classes CRUD para manipulação de dados
7. Simulador de sensores IoT
8. Controlador de luzes com comunicação serial
9. **Camada de Dados (Model)**
10. Arquivos TXT como banco de dados
11. Estrutura CSV para armazenamento
12. Operações de leitura e escrita
13. **Camada de Hardware (Arduino)**
14. Sensores de presença, luminosidade e temperatura
15. Controle de LEDs representando luzes dos setores
16. Comunicação serial com o sistema Python

Padrões de Design Utilizados

- **MVC (Model-View-Controller)**: Separação clara entre dados, lógica e apresentação
- **Singleton**: Para controladores únicos de sensores e luzes
- **Observer**: Para atualização em tempo real dos dados dos sensores

---

3. Módulos Implementados

3.1 Sistema CRUD

O sistema CRUD permite operações completas de Create, Read, Update e Delete para as seguintes entidades:

Entidades Gerenciadas:

- **Peças**: Controle de estoque de peças para manutenção
- **Fornecedores**: Cadastro de fornecedores com dados completos
- **Caminhões**: Registro da frota com status e quilometragem
- **Funcionários**: Gestão de recursos humanos
- **Clientes**: Base de clientes da transportadora
- **Saídas de Caminhões**: Registro de viagens e cargas

Funcionalidades:

- Criação de novos registros
- Consulta e listagem de dados
- Atualização de informações existentes
- Exclusão de registros
- Validação de dados de entrada
- Interface gráfica intuitiva com Treeview

3.2 Sistema de Sensores IoT

Módulo responsável pelo monitoramento automatizado do ambiente da transportadora.

Sensores Simulados:

- **Sensor de Presença**: Detecta movimento nos galpões
- **Sensor de Luminosidade**: Monitora níveis de luz ambiente
- **Sensor de Temperatura**: Controla temperatura dos ambientes
- **Detector de Fumaça**: Sistema de segurança contra incêndios

Atuadores Automáticos:

- **Alarme**: Ativado quando há detecção de movimento
- **Controle de Luzes**: Automático baseado na luminosidade
- **Sirene de Emergência**: Ativada por alta temperatura ou fumaça
- **Sistema de Ventilação**: Acionado automaticamente

Características:

- Simulação em tempo real
- Interface gráfica com dados ao vivo
- Log de eventos
- Configurações personalizáveis
- Teste de emergência

3.3 Sistema de Controle de Luzes

Sistema para controle manual das luzes dos diferentes setores da transportadora.

Setores Controlados:

- Oficina
- Galpão (3 blocos independentes)
- Escritório
- Corredor
- Área de Serviço
- Área Externa

Funcionalidades:

- Controle individual por setor
- Controle geral (ligar/desligar todas)
- Comunicação serial com Arduino
- Status em tempo real
- Log de comandos
- Modo simulação quando Arduino não conectado

---

4. Estrutura de Arquivos

Arquivos Principais:

```
LogiTrack/
├── main_app.py              # Aplicação principal
├── crud_operations.py       # Classes CRUD para todas as
entidades
├── crud_view.py            # Interface gráfica para CRUD
├── sensor_simulator.py     # Simulador de sensores IoT
├── sensor_view.py          # Interface gráfica para sensores
├── light_controller.py     # Controlador de luzes
├── light_view.py           # Interface gráfica para luzes
├── arduino_sensores.ino    # Código Arduino para sensores
├── arduino_luzes.ino       # Código Arduino para luzes
├── estrutura_txt.md        # Documentação da estrutura dos
arquivos
├── instrucoes_tinkercad.md # Instruções para implementação no
Tinkercad
└── documentacao_tecnica.md # Esta documentação
```

Arquivos de Dados (TXT):

```
├── pecas.txt               # Dados das peças
├── fornecedores.txt       # Dados dos fornecedores
├── caminhoes.txt          # Dados dos caminhões
├── funcionarios.txt       # Dados dos funcionários
├── clientes.txt           # Dados dos clientes
└── saidas_caminhoes.txt   # Registro de saídas
```

Dependências:

- **customtkinter**: Interface gráfica moderna
- **pyserial**: Comunicação serial com Arduino
- **threading**: Processamento paralelo para sensores
- **tkinter**: Componentes adicionais da interface

---

5. Banco de Dados (Arquivos TXT)

O sistema utiliza arquivos de texto no formato CSV para armazenamento de dados, conforme especificado nos requisitos.

5.1 Estrutura dos Arquivos

pecas.txt

```
ID_Peca,Nome,Descricao,Fabricante,Preco,Quantidade
P001,Pneu,Pneu para caminhão,Michelin,550.0,8
```

fornecedores.txt

```
ID_Fornecedor,Nome,Contato,Endereco_Rua,Endereco_Bairro,Endereco_Cidade,En
F001,Fornecedor A,Contato A,Rua A,Bairro A,Cidade A,Estado A,
1111-2222,a@email.com
```

caminhoes.txt

```
ID_Caminhao,Marca,Modelo,Ano,Placa,Quilometragem,Status
C001,Volvo,FH,2020,ABC-1234,150000.0,em viagem
```

funcionarios.txt

```
ID_Funcionario,Nome,Cargo,Telefone,Email,Endereco_Rua,Endereco_Bairro,Ende
FUNC001,João Silva,Motorista Senior,
3333-4444,joao@email.com,Rua X,Bairro Y,Cidade Z,Estado W
```

clientes.txt

```
ID_Cliente,Nome,Contato,Endereco_Rua,Endereco_Bairro,Endereco_Cidade,Ende
CLI001,Empresa Alfa,Contato Alfa,Av. Principal,Centro,Cidade
A,Estado B,7777-9999,alfa@email.com
```

saidas_caminhoes.txt

```
ID_Saida,ID_Caminhao,ID_Cliente,Tipo_Carga,Destino,Horario_Entrada,Horari
SAIDA001,C001,CLI001,Eletrônicos,Belo Horizonte,2025-06-15
08:00,2025-06-15 08:30,150000.0,150050.0
```

5.2 Operações de Dados

Características:

- **Formato CSV**: Facilita importação/exportação

- **Validação**: Verificação de tipos e formatos
- **Integridade**: Prevenção de IDs duplicados
- **Backup**: Arquivos podem ser facilmente copiados
- **Portabilidade**: Compatível com qualquer sistema

---

6. Interface Gráfica

6.1 Tecnologia Utilizada

O sistema utiliza **CustomTkinter**, uma biblioteca moderna que oferece: - Design contemporâneo e profissional - Temas claro/escuro - Componentes responsivos - Escalabilidade da interface

6.2 Estrutura da Interface

Tela Principal:

- **Sidebar de Navegação**: Acesso rápido aos módulos
- **Área Principal**: Conteúdo dinâmico baseado na seleção
- **Configurações**: Tema e escala da interface

Módulos da Interface:

1. **Home**: Tela de boas-vindas com visão geral
2. **CRUD**: Acesso aos gerenciadores de dados
3. **Sensores IoT**: Monitoramento em tempo real
4. **Controle de Luzes**: Interface de controle manual

6.3 Características da Interface

- **Responsiva**: Adapta-se a diferentes tamanhos de tela
- **Intuitiva**: Navegação clara e organizada
- **Moderna**: Design profissional com CustomTkinter
- **Acessível**: Controles grandes e texto legível
- **Consistente**: Padrão visual em todos os módulos

---

7. Sistema de Sensores IoT

7.1 Arquitetura do Sistema

O módulo de sensores simula um ambiente IoT completo com:

Componentes:

- **SensorSimulator**: Classe principal de simulação
- **SensorView**: Interface gráfica para visualização
- **Threading**: Processamento paralelo para tempo real

7.2 Sensores Implementados

Sensor de Presença/Distância:

- **Tipo**: Ultrassônico (simulado)
- **Função**: Detectar movimento nos galpões
- **Ação**: Ativar alarme quando detectado

Sensor de Luminosidade:

- **Tipo**: LDR (simulado)
- **Função**: Medir níveis de luz ambiente
- **Ação**: Controlar luzes automaticamente

Sensor de Temperatura:

- **Tipo**: TMP36 (simulado)
- **Função**: Monitorar temperatura ambiente
- **Ação**: Ativar ventilação e sirene se necessário

Detector de Fumaça:

- **Tipo**: MQ-2 (simulado)
- **Função**: Detectar presença de fumaça
- **Ação**: Ativar sirene de emergência

7.3 Lógica de Automação

Regras Implementadas:

1. **Presença detectada** → Ativar alarme
2. **Luminosidade < 30%** → Ligar luzes automáticas

3. **Temperatura > 35°C** → Ativar ventilação e sirene
4. **Fumaça detectada** → Ativar sirene de emergência

Configurações:

- Temperatura máxima configurável
- Luminosidade mínima configurável
- Timeout de presença ajustável
- Controle automático de luzes liga/desliga

---

8. Controle de Luzes

8.1 Mapeamento de Setores

O sistema controla 8 setores independentes:

Setor	Comando Ligar	Comando Desligar	Pino Arduino
Oficina	A	a	2
Galpão Bloco 1	B	b	3
Galpão Bloco 2	C	c	4
Galpão Bloco 3	D	d	5
Escritório	E	e	6
Corredor	F	f	7
Área de Serviço	G	g	8
Área Externa	H	h	9

8.2 Comunicação Serial

Protocolo:

- **Baud Rate**: 9600
- **Formato**: Caracteres ASCII simples
- **Comandos**: Maiúscula = Ligar, Minúscula = Desligar
- **Especiais**: 'T' = Teste, 'S' = Status

Características:

- **Modo Simulação**: Funciona sem Arduino conectado
- **Detecção Automática**: Identifica portas disponíveis
- **Feedback**: Confirmação de comandos enviados
- **Log**: Registro de todas as operações

8.3 Interface de Controle

Funcionalidades:

- **Conexão Arduino**: Campo para porta serial
- **Controles Gerais**: Ligar/desligar todas as luzes
- **Controles Individuais**: Botões para cada setor
- **Status Visual**: Indicadores coloridos do estado
- **Log de Comandos**: Histórico de operações

---

9. Códigos Arduino

9.1 Arduino para Sensores (arduino_sensores.ino)

Funcionalidades:

- Leitura de sensor ultrassônico (presença)
- Leitura de LDR (luminosidade)
- Leitura de TMP36 (temperatura)
- Simulação de detector de fumaça
- Controle de LED de alarme
- Controle de buzzer para sirene
- Envio de dados via Serial

Formato de Saída:

```
PRESENCA,LUMINOSIDADE,TEMPERATURA,FUMACA
1,45,28.5,0
```

Pinos Utilizados:

- Pino 2: Trigger do sensor ultrassônico
- Pino 3: Echo do sensor ultrassônico
- Pino A0: Sensor de luminosidade (LDR)

- Pino A1: Sensor de temperatura (TMP36)
- Pino 13: LED de alarme
- Pino 12: Buzzer para sirene

9.2 Arduino para Luzes (arduino_luzes.ino)

Funcionalidades:

- Recepção de comandos via Serial
- Controle de 8 LEDs independentes
- Feedback de status
- Comandos de teste
- Relatório de status geral

Comandos Aceitos:

- **A-H**: Ligar luzes dos setores
- **a-h**: Desligar luzes dos setores
- **T**: Teste de conexão (piscar todas as luzes)
- **S**: Mostrar status de todas as luzes

Pinos Utilizados:

- Pinos 2-9: LEDs representando luzes dos setores

---

10. Instalação e Configuração

10.1 Requisitos do Sistema

Software:

- **Python 3.11+**: Linguagem principal
- **Sistema Operacional**: Windows, Linux ou macOS
- **Arduino IDE**: Para programação dos microcontroladores
- **Tinkercad**: Para simulação (opcional)

Hardware (Opcional):

- **Arduino Uno**: Para controle físico
- **Sensores**: HC-SR04, LDR, TMP36, MQ-2
- **LEDs e Resistores**: Para indicação visual
- **Buzzer**: Para alarmes sonoros

10.2 Instalação das Dependências

Passo 1: Instalar Python

```
# Verificar versão do Python
python --version

# Deve ser 3.11 ou superior
```

Passo 2: Instalar Bibliotecas

```
# Instalar CustomTkinter
pip install customtkinter

# Instalar PySerial
pip install pyserial
```

Passo 3: Baixar o Sistema

```
# Clonar ou baixar os arquivos do sistema
# Extrair em uma pasta dedicada
```

10.3 Configuração Inicial

Estrutura de Pastas:

```
LogiTrack/
├── Arquivos Python (.py)
├── Códigos Arduino (.ino)
├── Documentação (.md)
└── Dados (arquivos .txt criados automaticamente)
```

Primeira Execução:

1. Executar `python main_app.py`
2. O sistema criará automaticamente os arquivos TXT
3. Testar cada módulo individualmente
4. Configurar Arduino se disponível

10.4 Configuração do Arduino

Para Sensores:

1. Abrir `arduino_sensores.ino` no Arduino IDE
2. Conectar componentes conforme diagrama
3. Fazer upload do código
4. Testar no Monitor Serial

Para Luzes:

1. Abrir `arduino_luzes.ino` no Arduino IDE
2. Conectar LEDs nos pinos especificados
3. Fazer upload do código
4. Testar comandos via Serial

---

11. Manual do Usuário

11.1 Iniciando o Sistema

1. **Executar o Programa**:
2. Duplo clique em `main_app.py` ou

3. Terminal: `python main_app.py`

4. **Tela Inicial**:

5. Bem-vindo ao LogiTrack
6. Visão geral das funcionalidades
7. Navegação pela sidebar

11.2 Usando o Sistema CRUD

Acessar CRUD:

1. Clicar em "CRUD" na sidebar
2. Escolher a entidade desejada
3. Usar os botões para operações

Operações Disponíveis:

- **Criar**: Preencher campos e clicar "Criar"
- **Consultar**: Selecionar item na lista

- **Atualizar**: Modificar campos e clicar "Atualizar"
- **Excluir**: Selecionar item e clicar "Excluir"

Dicas:

- Campos obrigatórios devem ser preenchidos
- IDs devem ser únicos
- Use "Limpar" para resetar formulário
- Lista atualiza automaticamente

11.3 Monitoramento de Sensores

Acessar Sensores:

1. Clicar em "Sensores IoT" na sidebar
2. Clicar "Iniciar Simulação"
3. Observar dados em tempo real

Funcionalidades:

- **Dados dos Sensores**: Valores atuais
- **Status dos Atuadores**: Estado dos dispositivos
- **Configurações**: Ajustar parâmetros
- **Log de Eventos**: Histórico de atividades

Controles:

- **Iniciar/Parar**: Controlar simulação
- **Teste de Emergência**: Simular situação crítica
- **Aplicar Config**: Salvar configurações
- **Toggle Luzes Auto**: Ligar/desligar automação
- **Reset Alarme**: Resetar alarme manualmente

11.4 Controle de Luzes

Acessar Controle:

1. Clicar em "Controle de Luzes" na sidebar
2. Configurar porta serial (se Arduino conectado)
3. Clicar "Conectar" ou usar modo simulação

Operações:

- **Conectar Arduino**: Inserir porta (ex: COM3) e conectar

- **Controles Gerais**: Ligar/desligar todas as luzes
- **Controles Individuais**: Botões para cada setor
- **Status**: Visualizar estado atual das luzes
- **Log**: Acompanhar comandos enviados

Setores Disponíveis:

- Oficina
- Galpão (3 blocos)
- Escritório
- Corredor
- Área de Serviço
- Área Externa

---

12. Testes Realizados

12.1 Testes Unitários

Sistema CRUD:

✅ **Criação de registros**: Todos os tipos de entidade ✅ **Leitura de dados**: Consulta individual e listagem ✅ **Atualização**: Modificação de campos específicos ✅ **Exclusão**: Remoção de registros ✅ **Validação**: Verificação de dados de entrada ✅ **Persistência**: Dados salvos em arquivos TXT

Sistema de Sensores:

✅ **Simulação**: Geração de dados realistas ✅ **Automação**: Ativação correta de atuadores ✅ **Interface**: Atualização em tempo real ✅ **Configuração**: Ajuste de parâmetros ✅ **Teste de Emergência**: Simulação de situações críticas

Controle de Luzes:

✅ **Comunicação Serial**: Envio de comandos ✅ **Modo Simulação**: Funcionamento sem Arduino ✅ **Controles**: Individual e geral ✅ **Status**: Acompanhamento do estado ✅ **Interface**: Feedback visual adequado

12.2 Testes de Integração

✅ **Navegação**: Transição entre módulos ✅ **Dados Compartilhados**: Consistência entre telas ✅ **Performance**: Responsividade da interface ✅ **Estabilidade**: Sistema sem travamentos ✅ **Usabilidade**: Interface intuitiva

12.3 Testes de Sistema

✅ **Funcionalidade Completa**: Todos os requisitos atendidos ✅ **Interface Gráfica**: CustomTkinter funcionando ✅ **Arquivos TXT**: Criação e manipulação corretas ✅ **Arduino**: Códigos testados no Tinkercad ✅ **Documentação**: Completa e atualizada

---

13. Limitações e Melhorias Futuras

13.1 Limitações Atuais

Técnicas:

- **Banco de Dados**: Arquivos TXT limitam consultas complexas
- **Concorrência**: Sem controle de acesso simultâneo
- **Backup**: Não há sistema automático de backup
- **Validação**: Validações básicas implementadas

Funcionais:

- **Relatórios**: Não há geração de relatórios
- **Gráficos**: Ausência de visualizações gráficas
- **Histórico**: Limitado aos logs de eventos
- **Usuários**: Sistema single-user

13.2 Melhorias Futuras

Curto Prazo:

- **Validações Avançadas**: Regras de negócio mais complexas
- **Backup Automático**: Sistema de backup periódico
- **Relatórios Básicos**: Geração de relatórios simples
- **Importação/Exportação**: Funcionalidades de dados

Médio Prazo:

- **Banco de Dados**: Migração para SQLite ou MySQL
- **Multi-usuário**: Sistema de login e permissões
- **API REST**: Interface para integração externa
- **Dashboard**: Painel de controle com gráficos

Longo Prazo:

- **Web Application**: Versão web do sistema
- **Mobile App**: Aplicativo para dispositivos móveis
- **IoT Real**: Integração com sensores físicos
- **Cloud**: Hospedagem em nuvem

13.3 Considerações de Segurança

Implementadas:

- **Validação de Entrada**: Prevenção de dados inválidos
- **Tratamento de Erros**: Captura de exceções
- **Modo Simulação**: Funcionamento seguro sem hardware

Recomendadas:

- **Criptografia**: Para dados sensíveis
- **Autenticação**: Sistema de login
- **Auditoria**: Log de todas as operações
- **Backup**: Estratégia de recuperação

Conclusão

O **LogiTrack** representa uma solução completa e funcional para gestão de transportadoras, atendendo a todos os requisitos especificados:

Objetivos Alcançados:

✅ **Sistema CRUD completo** com interface gráfica moderna ✅ **Armazenamento em arquivos TXT** conforme especificado ✅ **Módulo de sensores IoT** com simulação realista ✅ **Controle de luzes** com comunicação serial ✅ **Códigos Arduino** funcionais no Tinkercad ✅ **Interface em CustomTkinter** exclusivamente ✅ **Separação View/Funcional** em arquivos distintos ✅ **Documentação completa** do sistema

Diferenciais Implementados:

- **Design Moderno**: Interface profissional com CustomTkinter
- **Simulação Realista**: Sensores IoT com comportamento real
- **Modo Híbrido**: Funciona com ou sem Arduino
- **Arquitetura Modular**: Código organizado e extensível
- **Documentação Completa**: Manual técnico e do usuário

O sistema está pronto para uso e pode ser facilmente expandido conforme as necessidades futuras da transportadora.

Desenvolvido com dedicação para a disciplina de Desenvolvimento de Sistemas
Data: 15 de Junho de 2025