

Enunciado do Projeto – Sistema de Gestão de Logística e Entregas

Contexto

A empresa fictícia de logística regional realiza entregas para diversos clientes (e-commerce, farmácias, restaurantes) utilizando uma frota de motoristas próprios e terceirizados.

Atualmente, a gestão é feita de forma manual (planilhas, WhatsApp, ligações), o que gera problemas como:

- Falta de controle sobre quais entregas cada motorista está realizando.
- Planejamento manual de rotas sem otimização.
- Clientes sem acesso ao rastreamento em tempo real.
- Ausência de histórico consolidado de desempenho dos motoristas.
- Entregas atrasadas ou perdidas sem registro adequado.

O objetivo deste projeto é desenvolver uma API REST que centralize a gestão de entregas, motoristas, veículos e rotas, com autenticação e controle de acesso por perfis de usuários.

Objetivos

Implementar um sistema backend que permita:

- Cadastrar e gerenciar entregas, motoristas, veículos e rotas.
- Atribuir entregas a motoristas e rotas, validando capacidade dos veículos.
- Permitir que motoristas atualizem status das entregas.
- Permitir que clientes rastreiem suas entregas.
- Garantir segurança e organização dos dados com autenticação e permissões.
- Disponibilizar documentação clara da API para facilitar integração com outros sistemas.

Estrutura de Dados

O sistema pode conter 4 entidades principais:

1. **Entrega (A)**
 - Representa uma solicitação de entrega de mercadoria,
 - Pode estar associada a uma rota,
 - Possui um status que evolui ao longo do processo,
 - Possui capacidade_necessaria em unidades de carga.
2. **Motorista (B)**
 - Profissional responsável por realizar entregas,
 - Pode estar associado a várias rotas,

- Histórico de entregas realizadas,
- Quando ativo, está vinculado a um veículo específico.

3. Rota (C)

- Agrupamento lógico de entregas,
- Pertence a um único motorista e utiliza um único veículo,
- Contém várias entregas em uma sequência planejada,
- Controle de capacidade total utilizada.

4. Veículo (D)

- Representa o veículo utilizado nas entregas,
- Possui capacidade máxima em unidades de carga,
- Controle de disponibilidade e quilometragem,
- Pode estar vinculado a um motorista ativo.

Relacionamentos obrigatórios:

- 1:N Motorista ↔ Entrega
- 1:N Motorista ↔ Rota
- 1:N Veículo ↔ Rota
- 1:N Rota ↔ Entrega
- 1:N Cliente ↔ Entrega
- 1:1 Veículo ↔ Motorista
- N:N Rota ↔ Cliente

1:N Motorista ↔ Entrega

- Um motorista realiza várias entregas ao longo do tempo,
- Uma entrega pode ter um motorista atribuído.

1:N Motorista ↔ Rota

- Um motorista pode ter várias rotas,
- Cada rota pertence a um único motorista.

1:N Veículo ↔ Rota

- Um veículo pode ser utilizado em várias rotas (ao longo do tempo),
- Cada rota utiliza um único veículo.

1:N – Rota ↔ Entrega

- Uma rota contém várias entregas,
- Cada entrega pertence a no máximo uma rota (ou pode estar sem rota ainda),
- Regra de negócio: Soma(capacidade_necessaria das entregas) ≤ capacidade_maxima do veículo.

1:N Cliente ↔ Entrega

- Um cliente pode possuir várias entregas,

- Cada entrega possui apenas um cliente (o cliente pode visualizar o status da entrega).

1:1 Veículo ↔ Motorista

- N:N Rota ↔ Cliente
- Uma rota possui vários clientes,
- Um cliente pode ter entregas em várias rotas.

Principais Usuários e Atores

1. Gestor de Logística (Administrador)
 - Maior privilégio no sistema,
 - Cadastra e gerencia entregas,
 - Cadastra e gerencia motoristas e veículos,
 - Cria e edita rotas,
 - Atribui entregas a rotas validando capacidade,
 - Vincula/desvincula veículos a motoristas,
 - Visualiza relatórios gerais (desempenho, entregas pendentes, capacidade de frota, etc.).
2. Motorista
 - Consulta suas rotas e entregas atribuídas,
 - Atualiza status das entregas (ex.: "saiu para entrega", "entregue", "cliente ausente"),
 - Visualiza detalhes de endereços e observações,
 - Pode adicionar observações sobre a entrega,
 - Acesso restrito apenas aos seus dados,
3. Cliente
 - Pode rastrear suas entregas via código de rastreio,
 - Visualiza status e previsão de entrega,
 - Acesso somente-leitura,

Requisitos Mínimos

- Rotas de backend implementando métodos **GET, POST, PUT, DELETE**.
 - CRUD completo para cada entidade.
 - No mínimo 3 rotas de relacionamento:
 - Motorista - Entrega (A-B)
 - Motorista - Rota (B-C)
 - Veiculo - Entrega (D-C)
 - Motorista - Veiculo (B-D)
 - Rota - Entrega (C-A)
 - Rota que Envolva A-B-C-D (Composição)
- Criar pelo menos uma rota "composição" que cruze as quatro entidades principais na mesma resposta.

- **Exemplos de proposta:**
- Opção 1:
 - GET /api/motoristas/{id}/visao-completa/
 - Retorna:
 - Dados do motorista (B),
 - Veículo atual vinculado (D) com capacidade e km,
 - Todas as rotas (C) desse motorista,
 - Para cada rota, as entregas (A) incluídas e capacidade utilizada.
- Opção 2:
 - GET /api/rotas/{id}/dashboard/
 - Retorna:
 - Dados da rota (C),
 - Dados do motorista (B) responsável,
 - Dados do veículo (D) utilizado com capacidade disponível/utilizada,
 - Lista completa de entregas (A) com status e capacidade individual.
- Autenticação:
 - Login via web (Django Auth ou similar).
 - Token Authentication (DRF Token Authentication).
- Controle de acesso por perfis:
 - **Gestores** → CRUD completo em todas as entidades.
 - **Motoristas** → acesso apenas às suas rotas e entregas.
 - **Clientes** → acesso somente-leitura às suas entregas.
- Registro no admin: todas as tabelas e tokens devem estar disponíveis.
- Documentação da API: gerada com drf-spectacular e disponível em Redoc.

Sugestão de Modelagem

Entidade: Entrega (A)

Campos principais:

- id (PK)
- codigo_rastreio
- cliente_id (FK → Cliente)
- endereco_origem
- endereco_destino
- status (pendente, em trânsito, entregue, cancelada)
- capacidade_necessaria

- valor_frete
- data_solicitacao
- data_entrega_prevista
- data_entrega_real
- observacoes

Entidade: Motorista (B)

Campos principais:

- id (PK)
- nome
- cpf
- cnh
- telefone
- status (ativo, inativo, em_rota, disponível)
- data_cadastro

Relacionamentos:

- 1:N com Entrega
- 1:N com Rota
- 1:1 com Veículo

Entidade: Rota (C)

Campos principais:

- id (PK)
- nome
- descricao
- motorista_id (FK → Motorista)
- veiculo_id (FK → Veículo)
- data_rota
- status (planejada, em_andamento, concluída)
- capacidade_total_utilizada
- km_total_estimado
- tempo_estimado

Entidade: Veículo (D)

Campos principais:

- id (PK)
- placa
- modelo
- tipo (carro, van, caminhão)
- capacidade_maxima
- km_atual
- status (disponível, em uso, manutenção)
- motorista_id (FK → Motorista, opcional)

Sugestão de Rotas da API

Rota principal

- `/api/` → ponto de entrada da API.

Entregas

- `/api/entregas/` → CRUD de entregas.
- `/api/entregas/{id}/` → detalhes de uma entrega.
- `/api/entregas/{id}/atribuir-motorista/` → atribui motorista.
- `/api/entregas/{id}/rastreamento/` → rastreamento completo da entrega.

Motoristas

- `/api/motoristas/` → CRUD de motoristas (apenas gestores).
- `/api/motoristas/{id}/entregas/` → listar entregas de um motorista.
- `/api/motoristas/{id}/rotas/` → listar rotas de um motorista.
- `/api/motoristas/{id}/atribuir-veiculo/` → vincular veículo.

Veículos

- `/api/veiculos/` → CRUD de veículos.
- `/api/veiculos/{id}/rotas/` → listar rotas que utilizaram o veículo.
- `/api/veiculos/disponiveis/` → listar veículos disponíveis.

Rotas

- `/api/rotas/` → CRUD de rotas.
- `/api/rotas/{id}/entregas/` → listar/adicionar/remover entregas da rota.
- `/api/rotas/{id}/dashboard/` → visão completa da rota (motorista, veículo, entregas).

Autenticação

- [/api/auth/token/](#) → obtenção de token.

Documentação

- [/api/schema/](#) → schema OpenAPI.
- [/api/docs/](#) → documentação interativa.
- [/api/docs/redoc/](#) → documentação interativa.

Sugestão de Melhorias

- Implementação de relatórios de desempenho de motoristas.
- Implementação de rastreamento em tempo real.
- Implementação de filtros avançados (django-filter).
- Validação automática de capacidade de veículos nas rotas.

Entregáveis

- Código fonte.
- Repositório GitHub organizado.
- Modelos, rotas, controllers/services.
- Arquivo de dependências (requirements.txt).
- Scripts de migração.
- Estrutura de pastas limpa.
- Documentação:
 - README com instruções de instalação e uso.
 - Documentação da API (Markdown ou Swagger).
 - Modelo de dados (MER + DER).
 - Diagrama de casos de uso (opcional).