

Programa de Desenvolvimento Tecnológico

Hackthon Educacional

GRUPO – Os Desinformados

PROJETO: Sistema de Gestão de Transporte Universitário (SIGTU)

Prefeitura Municipal de Caraguatatuba

2025

Sistema de Gestão de Transporte Universitário (SIGTU)

PROJETO: Sistema de Gestão de Transporte Universitário (SIGTU) Prefeitura Municipal de Caraguatatuba

2025: Detalhamento dos Requisitos Funcionais e Arquitetura do SGTU-C

1. Módulo de Cadastro de Estudantes (RF1)

O cadastro do estudante, que será alimentado pelo próprio Estudante (portal) ou pelo Operacional (secretaria), deve contemplar as seguintes informações, armazenadas na tabela estudantes e suas relacionadas:

Seção	Campo (DB)	Tipo de Dado	Obrigatório	Propósito Logístico/Social
Dados Pessoais	nome, cpf, rg, data_nascimento, genero	VARCHAR/ DATE	Sim	Identificação e validação única.
Contato & Endereço	telefone_principal, email, cep, logradouro, numero, bairro, cidade_residencia	VARCHAR	Sim	Comunicação e mapeamento da linha de transporte.
Situação Familiar e Habitacional	status_habacional (Própria, Alugada, Cedida), qtd_membros_familia, renda_familia_r_mensal	ENUM/INT/ DECIMAL	Sim	Análise socioeconômica para programas de auxílio.
Dados Acadêmicos	instituicao_id (FK), curso, matricula_universitaria, turno (Manhã/Noite/Integral), previsao_conclusao	FK/VARCHAR/ENUM/DATE	Sim	Definição da linha de transporte e elegibilidade.

Bolsa de Estudos	possui_bolsa (S/N), tipo_bolsa (Prouni, Fies, Institucional), percentual_bolsa	BOOLEAN/E NUM/DECIMAL	Não	Informação complementar para elegibilidade/prioridade.
Termo de Responsabilidade	termo_aceite	BOOLEAN	Sim	Requisito legal (deve ser marcado/assinado digitalmente no cadastro).

2. Módulo de Gestão Administrativa (RF-Gestão)

Este módulo é de uso exclusivo dos perfis **Administrador** e **Operacional**.

2.1.

Painel do Gestor (RF-G.1)

- **RF-G.1.1:** Apresentar um dashboard com indicadores chave (KPIs):
 - % de Ocupação Média das linhas.
 - Total de Estudantes Ativos/Inativos/Pendentes.
 - Taxa de Faltas (média semanal/mensal).
 - Status do Controle Documental (Pendente de Análise).
- **RF-G.1.2:** Exibir alertas importantes (e.g., Linha com alta taxa de lotação, Cadastro expirando).

2.2.

Gestão de Linhas (RF-G.2)

- **RF-G.2.1:** CRUD completo para linhas (nome, veículo, capacidade, rota, instituições atendidas).
- **RF-G.2.2:** Funcionalidade para alocação/relocação automática de estudantes em linhas, considerando turno e instituição, dentro do limite de capacidade_veiculo.

2.3.

Controle Documental (RF-G.3)

- **RF-G.3.1:** Interface centralizada para análise de documentos pendentes, com filtros por tipo de documento/status.
- **RF-G.3.2:** Geração automática da lista de estudantes aptos a embarcar após a aprovação de todos os documentos.

3. Módulo de Controle de Assiduidade (RF-Assiduidade)

Este módulo é de uso principal do **Fiscal de Transporte** e deve ser otimizado para **uso mobile**.

3.1.

Sistema de Presença (RF-A.1)

- **RF-A.1.1:** O sistema deve permitir o registro rápido da presença (embarque) via leitura de código (**QR Code, Cartão de Identificação, ou ID do aluno**).
- **RF-A.1.2:** Ao registrar, o sistema deve verificar instantaneamente se o estudante está **Ativo e Alocado** na linha atual.
- **RF-A.1.3:** O registro deve salvar a `linha_id`, `estudante_id`, `data_registro` e o tipo de evento (Embarque/Desembarque).

3.2.

Controle de Faltas (RF-A.2)

- **RF-A.2.1:** O sistema deve automaticamente consolidar as ausências registradas em um período.
- **RF-A.2.2:** Implementar uma regra para suspensão automática de estudantes que excederem um limite de faltas.

3.3.

Relatório de Frequência (RF-A.3)

- **RF-A.3.1:** Geração de relatórios detalhados de frequência por estudante, por linha e por período, indicando o total de presenças e faltas.
- **RF-A.3.2:** Exportação dos relatórios em **PDF/CSV** para prestação de contas.

4. Requisitos de Arquitetura e Tecnologia (RNF) - Foco JavaScript/Node.js

Requisito	Detalhamento Técnico	Diretriz JavaScript/Node.js
Compatibilidade Web/Mobile	Design Responsivo (RNF3.1) utilizando framework CSS (Bootstrap) e desenvolvimento Mobile-First.	Utilizar Frameworks Front-end (Ex: React, Vue, Angular) para construção de Views e Componentes Dinâmicos. O JavaScript gerencia a interface responsiva.
Banco de Dados	Relacional (MySQL/MariaDB). O sistema deve utilizar Prepared Statements para todas as consultas (proteção contra SQL Injection).	Utilizar um ORM/Query Builder (Ex: Sequelize, Knex.js) no Node.js para interagir com o DB. Usar <i>Parameterized Queries</i> por padrão para mitigar SQL Injection.

Backup Integrado	Definir um script de backup automático (e.g., cron job no servidor) que exporte o banco de dados e arquivos de documentos para um local seguro.	Utilizar um módulo Node.js (e.g., <code>node-cron</code>) ou um script Bash/Shell invocado pelo Node.js para agendamento de tarefas (cron).
Interface Administrativa	Interface exclusiva e segura para os perfis Gestor/Operacional, separada do portal do Estudante.	Uso de Middlewares (Ex: Express/Koa) para controle de acesso baseado em <i>tokens</i> (JWT) e verificação de perfil (<i>role-based access control</i> - RNF2.3).
API de Integração	Implementar uma API RESTful básica (em JSON) para permitir que outros sistemas municipais (ex: matrícula, assistência social) consultem o status do estudante.	Construir <i>Endpoints</i> com Node.js e Express/NestJS que retornem dados em formato JSON , protegidos por chaves de API (RNF2.2).