Documentação do Sistema de Transporte Escolar

Visão Geral

O Sistema de Transporte Escolar é uma API REST desenvolvida em PHP utilizando o framework Laravel. O sistema permite gerenciar rotas escolares, motoristas, monitores, ônibus, alunos, paradas, horários, viagens e controle de presença dos alunos.

Arquitetura

O sistema segue uma arquitetura em camadas:

1. **Controllers**: Responsáveis por receber as requisições HTTP, validar os dados de entrada e retornar as respostas apropriadas.
2. **Services**: Contém a lógica de negócio da aplicação, fazendo a intermediação entre os controllers e os models.
3. **Models**: Representam as entidades do banco de dados e suas relações.
4. **Migrations**: Definem a estrutura do banco de dados.
5. **Seeders**: Populam o banco de dados com dados iniciais para testes.

Entidades Principais

Aluno

Representa um estudante que utiliza o transporte escolar.

Motorista

Profissional responsável por conduzir o veículo.

Monitor

Profissional responsável por acompanhar os alunos durante a viagem.

Ônibus

Veículo utilizado para o transporte dos alunos.

Rota

Trajeto definido para o transporte, com origem, destino e paradas intermediárias.

Parada

Ponto de embarque e desembarque de alunos.

Horário

Define os dias da semana e horários em que uma rota opera.

Viagem

Representa a execução de uma rota em um determinado dia e horário, com um ônibus, motorista e monitor específicos.

Presença

Registra a presença ou ausência de um aluno em uma viagem específica.

Fluxos Principais

1. Cadastro de Alunos

1. O usuário acessa o endpoint `POST /api/alunos` enviando os dados do aluno
2. O sistema valida os dados recebidos
3. Se válidos, o aluno é cadastrado no banco de dados
4. Sistema retorna os dados do aluno cadastrado

2. Cadastro de Rotas e Paradas

1. O usuário cadastra paradas via endpoint `POST /api/paradas`
2. O usuário cadastra uma rota via endpoint `POST /api/rotas`
3. O usuário associa paradas à rota definindo a ordem em que serão percorridas

3. Cadastro de Horários

1. O usuário cadastra horários para uma rota específica via endpoint `POST /api/horarios`
2. O sistema registra em quais dias da semana e horários a rota irá operar

4. Programação de Viagens

1. O usuário programa uma viagem via endpoint `POST /api/viagens`
2. O sistema associa uma rota, data, horário, ônibus, motorista e monitor à viagem
3. A viagem é registrada com status ativo e horários previstos de saída e chegada

5. Registro de Presença

1. Durante a execução da viagem, o monitor registra a presença dos alunos via endpoint `POST /api/presencas`
2. O sistema associa o registro de presença à viagem e ao aluno específico
3. O monitor pode registrar observações sobre cada aluno

6. Finalização de Viagem

1. Ao término da viagem, o usuário atualiza os dados via endpoint `PUT /api/viagens/{id}`
2. O sistema registra os horários reais de saída e chegada
3. O sistema pode atualizar o status da viagem para concluída

Detalhamento das APIs

Alunos API

Listar Alunos

- **Endpoint**: `GET /api/alunos`
- **Descrição**: Retorna a lista de todos os alunos cadastrados
- **Parâmetros de Query**:
  - `page`: Número da página (padrão: 1)
  - `per_page`: Itens por página (padrão: 10)

Obter Aluno

- **Endpoint**: `GET /api/alunos/{id}`
- **Descrição**: Retorna os detalhes de um aluno específico
- **Parâmetros de Path**:
  - `id`: ID do aluno

Criar Aluno

- **Endpoint**: `POST /api/alunos`
- **Descrição**: Cadastra um novo aluno

Corpo da Requisição:
{ "nome": "Nome do Aluno", "descricao": "Descrição opcional", "data_nascimento": "2015-03-15", "responsavel": "Nome do Responsável", "telefone_responsavel": "+5511999999999", "endereco": "Endereço completo", "ponto_referencia": "Referência opcional", "status": true}

-

Atualizar Aluno

- **Endpoint**: `PUT /api/alunos/{id}`

- **Descrição**: Atualiza os dados de um aluno existente
- **Parâmetros de Path**:
    - `id`: ID do aluno
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Aluno

- **Endpoint**: `DELETE /api/alunos/{id}`
- **Descrição**: Remove um aluno do sistema
- **Parâmetros de Path**:
    - `id`: ID do aluno

Listar Presenças do Aluno

- **Endpoint**: `GET /api/alunos/{id}/presencas`
- **Descrição**: Retorna o histórico de presenças de um aluno
- **Parâmetros de Path**:
    - `id`: ID do aluno

Motoristas API

Listar Motoristas

- **Endpoint**: `GET /api/motoristas`
- **Descrição**: Retorna a lista de todos os motoristas cadastrados

Obter Motorista

- **Endpoint**: `GET /api/motoristas/{id}`
- **Descrição**: Retorna os detalhes de um motorista específico
- **Parâmetros de Path**:
    - `id`: ID do motorista

Criar Motorista

- **Endpoint**: `POST /api/motoristas`
- **Descrição**: Cadastra um novo motorista

Corpo da Requisição:
{ "nome": "Nome do Motorista", "cpf": "123.456.789-00", "cnh": "12345678900", "categoria_cnh": "D", "validade_cnh": "2026-05-20", "telefone": "+5511977777777", "endereco": "Endereço completo", "data_contratacao": "2022-01-15", "status": true}

-

Atualizar Motorista

- **Endpoint**: `PUT /api/motoristas/{id}`
- **Descrição**: Atualiza os dados de um motorista existente

- **Parâmetros de Path**:
    - `id`: ID do motorista
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Motorista

- **Endpoint**: `DELETE /api/motoristas/{id}`
- **Descrição**: Remove um motorista do sistema
- **Parâmetros de Path**:
    - `id`: ID do motorista

Listar Viagens do Motorista

- **Endpoint**: `GET /api/motoristas/{id}/viagens`
- **Descrição**: Retorna as viagens associadas a um motorista
- **Parâmetros de Path**:
    - `id`: ID do motorista

Monitores API

Listar Monitores

- **Endpoint**: `GET /api/monitores`
- **Descrição**: Retorna a lista de todos os monitores cadastrados

Obter Monitor

- **Endpoint**: `GET /api/monitores/{id}`
- **Descrição**: Retorna os detalhes de um monitor específico
- **Parâmetros de Path**:
    - `id`: ID do monitor

Criar Monitor

- **Endpoint**: `POST /api/monitores`
- **Descrição**: Cadastra um novo monitor

Corpo da Requisição:
{ "nome": "Nome do Monitor", "cpf": "987.654.321-00", "telefone": "+5511955555555", "endereco": "Endereço completo", "data_contratacao": "2022-03-15", "status": "Ativo"}

-

Atualizar Monitor

- **Endpoint**: `PUT /api/monitores/{id}`
- **Descrição**: Atualiza os dados de um monitor existente
- **Parâmetros de Path**:

    - `id`: ID do monitor
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Monitor

- **Endpoint**: `DELETE /api/monitores/{id}`
- **Descrição**: Remove um monitor do sistema
- **Parâmetros de Path**:
    - `id`: ID do monitor

Listar Viagens do Monitor

- **Endpoint**: `GET /api/monitores/{id}/viagens`
- **Descrição**: Retorna as viagens associadas a um monitor
- **Parâmetros de Path**:
    - `id`: ID do monitor

Ônibus API

Listar Ônibus

- **Endpoint**: `GET /api/onibus`
- **Descrição**: Retorna a lista de todos os ônibus cadastrados

Obter Ônibus

- **Endpoint**: `GET /api/onibus/{id}`
- **Descrição**: Retorna os detalhes de um ônibus específico
- **Parâmetros de Path**:
    - `id`: ID do ônibus

Criar Ônibus

- **Endpoint**: `POST /api/onibus`
- **Descrição**: Cadastra um novo ônibus

Corpo da Requisição:
{ "placa": "ABC1D234", "modelo": "Mercedes Benz O500U", "capacidade": 40, "ano_fabricacao": 2020, "status": "Ativo"}

-

Atualizar Ônibus

- **Endpoint**: `PUT /api/onibus/{id}`
- **Descrição**: Atualiza os dados de um ônibus existente
- **Parâmetros de Path**:
    - `id`: ID do ônibus

- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Ônibus

- **Endpoint**: `DELETE /api/onibus/{id}`
- **Descrição**: Remove um ônibus do sistema
- **Parâmetros de Path**:
  - `id`: ID do ônibus

Listar Viagens do Ônibus

- **Endpoint**: `GET /api/onibus/{id}/viagens`
- **Descrição**: Retorna as viagens associadas a um ônibus
- **Parâmetros de Path**:
  - `id`: ID do ônibus

Paradas API

Listar Paradas

- **Endpoint**: `GET /api/paradas`
- **Descrição**: Retorna a lista de todas as paradas cadastradas

Obter Parada

- **Endpoint**: `GET /api/paradas/{id}`
- **Descrição**: Retorna os detalhes de uma parada específica
- **Parâmetros de Path**:
  - `id`: ID da parada

Criar Parada

- **Endpoint**: `POST /api/paradas`
- **Descrição**: Cadastra uma nova parada

Corpo da Requisição:
{ "nome": "Escola Municipal", "endereco": "Rua da Educação, 100", "ponto_referencia": "Em frente à praça", "latitude": -23.5505, "longitude": -46.6333, "tipo": "Inicio", "status": true}

-

Atualizar Parada

- **Endpoint**: `PUT /api/paradas/{id}`
- **Descrição**: Atualiza os dados de uma parada existente
- **Parâmetros de Path**:
  - `id`: ID da parada
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Parada

- **Endpoint**: `DELETE /api/paradas/{id}`
- **Descrição**: Remove uma parada do sistema
- **Parâmetros de Path**:
    - `id`: ID da parada

Rotas API

Listar Rotas

- **Endpoint**: `GET /api/rotas`
- **Descrição**: Retorna a lista de todas as rotas cadastradas

Obter Rota

- **Endpoint**: `GET /api/rotas/{id}`
- **Descrição**: Retorna os detalhes de uma rota específica
- **Parâmetros de Path**:
    - `id`: ID da rota

Criar Rota

- **Endpoint**: `POST /api/rotas`
- **Descrição**: Cadastra uma nova rota

Corpo da Requisição:
{ "nome": "Rota Centro-Bairro", "descricao": "Rota que liga o centro da cidade ao bairro residencial", "origem": "Centro", "destino": "Bairro", "horario_inicio": "08:00", "horario_fim": "18:00", "tipo": "Escolar", "distancia_km": 15.5, "tempo_estimado_minutos": 45, "status": true}

-

Atualizar Rota

- **Endpoint**: `PUT /api/rotas/{id}`
- **Descrição**: Atualiza os dados de uma rota existente
- **Parâmetros de Path**:
    - `id`: ID da rota
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Rota

- **Endpoint**: `DELETE /api/rotas/{id}`
- **Descrição**: Remove uma rota do sistema
- **Parâmetros de Path**:
    - `id`: ID da rota

Listar Paradas da Rota

- **Endpoint**: `GET /api/rotas/{id}/paradas`
- **Descrição**: Retorna as paradas associadas a uma rota
- **Parâmetros de Path**:
    - `id`: ID da rota

Listar Viagens da Rota

- **Endpoint**: `GET /api/rotas/{id}/viagens`
- **Descrição**: Retorna as viagens associadas a uma rota
- **Parâmetros de Path**:
    - `id`: ID da rota

Horários API

Listar Horários

- **Endpoint**: `GET /api/horarios`
- **Descrição**: Retorna a lista de todos os horários cadastrados

Obter Horário

- **Endpoint**: `GET /api/horarios/{id}`
- **Descrição**: Retorna os detalhes de um horário específico
- **Parâmetros de Path**:
    - `id`: ID do horário

Criar Horário

- **Endpoint**: `POST /api/horarios`
- **Descrição**: Cadastra um novo horário

Corpo da Requisição:
{ "rota_id": 1, "dias_semana": [1, 3, 5], "hora_inicio": "07:00", "hora_fim": "07:45", "status": true}

-

Atualizar Horário

- **Endpoint**: `PUT /api/horarios/{id}`
- **Descrição**: Atualiza os dados de um horário existente
- **Parâmetros de Path**:
    - `id`: ID do horário
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Horário

- **Endpoint**: `DELETE /api/horarios/{id}`
- **Descrição**: Remove um horário do sistema
- **Parâmetros de Path**:
    - `id`: ID do horário

Listar Viagens do Horário

- **Endpoint**: `GET /api/horarios/{id}/viagens`
- **Descrição**: Retorna as viagens associadas a um horário
- **Parâmetros de Path**:
    - `id`: ID do horário

Viagens API

Listar Viagens

- **Endpoint**: `GET /api/viagens`
- **Descrição**: Retorna a lista de todas as viagens cadastradas

Obter Viagem

- **Endpoint**: `GET /api/viagens/{id}`
- **Descrição**: Retorna os detalhes de uma viagem específica
- **Parâmetros de Path**:
    - `id`: ID da viagem

Criar Viagem

- **Endpoint**: `POST /api/viagens`
- **Descrição**: Cadastra uma nova viagem

Corpo da Requisição:
{ "data_viagem": "2024-03-20", "rota_id": 1, "onibus_id": 1, "motorista_id": 1, "monitor_id": 1, "horario_id": 1, "hora_saida_prevista": "07:00", "hora_chegada_prevista": "07:45", "status": true}

-

Atualizar Viagem

- **Endpoint**: `PUT /api/viagens/{id}`
- **Descrição**: Atualiza os dados de uma viagem existente
- **Parâmetros de Path**:
    - `id`: ID da viagem

Corpo da Requisição: Campos a serem atualizados (todos opcionais)
{ "status": false, "hora_saida_real": "07:05", "hora_chegada_real": "07:50", "observacoes": "Viagem concluída sem incidentes"}

-

Excluir Viagem

- **Endpoint**: `DELETE /api/viagens/{id}`
- **Descrição**: Remove uma viagem do sistema
- **Parâmetros de Path**:
    - `id`: ID da viagem

Presenças API

Listar Presenças

- **Endpoint**: `GET /api/presencas`
- **Descrição**: Retorna a lista de todas as presenças registradas

Obter Presença

- **Endpoint**: `GET /api/presencas/{id}`
- **Descrição**: Retorna os detalhes de uma presença específica
- **Parâmetros de Path**:
    - `id`: ID da presença

Criar Presença

- **Endpoint**: `POST /api/presencas`
- **Descrição**: Registra uma nova presença

Corpo da Requisição:
{ "viagem_id": 1, "aluno_id": 1, "hora_embarque": "07:10", "presente": true, "observacoes": "Aluno embarcou no ponto normal"}

-

Atualizar Presença

- **Endpoint**: `PUT /api/presencas/{id}`
- **Descrição**: Atualiza os dados de uma presença existente
- **Parâmetros de Path**:
    - `id`: ID da presença
- **Corpo da Requisição**: Campos a serem atualizados (todos opcionais)

Excluir Presença

- **Endpoint**: `DELETE /api/presencas/{id}`
- **Descrição**: Remove um registro de presença do sistema
- **Parâmetros de Path**:
    - `id`: ID da presença

Listar Presenças por Viagem

- **Endpoint**: `GET /api/presencas/viagem/{viagemId}`
- **Descrição**: Retorna as presenças associadas a uma viagem
- **Parâmetros de Path**:
    - `viagemId`: ID da viagem

Listar Presenças por Aluno

- **Endpoint**: `GET /api/presencas/aluno/{alunoId}`
- **Descrição**: Retorna as presenças associadas a um aluno
- **Parâmetros de Path**:
    - `alunoId`: ID do aluno

Regras de Negócio

1. **Validação de Dados**:
    - CPF de motoristas e monitores devem ser únicos
    - Placa de ônibus deve ser única
    - Datas de nascimento de alunos devem ser válidas
    - Horários de início e fim devem ser válidos e o horário de fim deve ser posterior ao de início
2. **Estados e Status**:
    - Monitores e Motoristas podem ter status: Ativo, Férias, Licença, Inativo
    - Ônibus podem ter status: Ativo, Manutenção, Inativo
    - Viagens possuem status booleano (ativo/inativo)
    - Paradas podem ser do tipo: Início, Intermediária, Final
3. **Relações**:
    - Uma viagem deve estar associada a um horário
    - Uma viagem deve estar associada a uma rota
    - Uma viagem deve estar associada a um ônibus
    - Uma viagem deve estar associada a um motorista
    - Uma viagem pode ter um monitor (opcional)
    - Uma presença deve estar associada a uma viagem e a um aluno
4. **Horários**:
    - Horários definem em quais dias da semana (0-6, onde 0 é domingo) a rota opera
    - Horários definem a hora de início e fim da operação
5. **Viagens**:
    - Viagens têm horários previstos de saída e chegada
    - Viagens podem ter horários reais de saída e chegada registrados posteriormente

    - Viagens são cadastradas para uma data específica
6. **Presença**:
    - A presença de um aluno é registrada para uma viagem específica
    - O sistema registra a hora de embarque do aluno
    - Cada registro de presença indica se o aluno esteve presente ou ausente
    - Observações podem ser adicionadas a cada registro de presença

Considerações para Implantação

1. **Permissões**: O sistema atual não possui uma camada de autenticação e autorização. Seria recomendável implementar controle de acesso para garantir que apenas usuários autorizados possam realizar operações específicas.

2. **Desempenho**: Para grandes volumes de dados, considere implementar cache e otimizar consultas ao banco de dados.

3. **Monitoramento**: Implemente um sistema de monitoramento para acompanhar o desempenho da API e detectar problemas.

4. **Backup**: Configure uma estratégia de backup para os dados do sistema.

5. **Segurança**: Implemente práticas de segurança como HTTPS, proteção contra CSRF, validação rigorosa de entradas, etc.

Conclusão

O Sistema de Transporte Escolar oferece uma solução completa para gerenciar o transporte de alunos, com funcionalidades para cadastro de rotas, paradas, horários, viagens e controle de presença. A API REST permite que diferentes tipos de clientes (web, mobile, etc.) possam integrar-se ao sistema.

A arquitetura em camadas facilita a manutenção e evolução do sistema, permitindo adicionar novas funcionalidades ou modificar as existentes com impacto mínimo nas outras partes do sistema.