Plano de Testes: Sistema de Gerenciamento de Veículos

Versão: 1.0 **Data:** 30/06/2025

---

1. Introdução e Visão Geral

Objetivo do Plano de Testes: Este documento detalha a abordagem e o planejamento dos testes para o projeto "Sistema de Gerenciamento de Veículos". O objetivo principal é garantir a qualidade, segurança e funcionalidade das operações de autenticação de usuário e do CRUD (Criar, Ler, Atualizar, Excluir) de veículos. O plano visa assegurar que o sistema atenda aos requisitos definidos, funcione conforme o esperado e seja seguro contra acessos não autorizados.

Escopo dos Testes:

- **Funcionalidades em Escopo:**
    - Autenticação de usuário via API (login) e geração de token JWT.
    - Proteção de rotas (endpoints) para garantir que apenas usuários autenticados possam realizar certas operações.
    - Criação de novos veículos (associados ao usuário autenticado).
    - Visualização da lista de todos os veículos.
    - Atualização de dados de veículos existentes.
    - Exclusão de veículos.
    - Validação de dados de entrada na API (campos obrigatórios, placas únicas).
- **Funcionalidades Fora de Escopo:**
    - Cadastro de novos usuários ou tela de "Esqueci minha senha".
    - Sistema de controle de entrada e saída da garagem (será abordado em outro plano).
    - Módulos de relatórios ou análises complexas.
    - Testes de desempenho, carga e stress (serão abordados em fases futuras).

---

2. Estratégia de Testes

A estratégia de testes será dividida em diferentes níveis para garantir uma cobertura abrangente, desde a lógica de negócio no backend até a experiência completa do usuário.

- **Níveis de Teste:**
    - **Testes de Unidade (com Jest):** Focados em validar as menores unidades de código de forma isolada.
        - **Aplicação:** Validar funções de serviço no backend (Node.js), como a lógica para verificar o formato de uma placa ou o ano de um veículo. Testar o middleware de autenticação (`auth`) para verificar se ele extrai o token corretamente.
    - **Testes de Integração (com Jest/Supertest):** Destinados a verificar a interação e comunicação entre diferentes módulos, principalmente entre a API e o banco de dados.
        - **Aplicação:** Assegurar que as rotas da API (endpoints) funcionem corretamente ao interagir com o banco de dados e que o middleware de `auth` proteja as rotas como esperado.
    - **Testes de Ponta a Ponta (E2E - com Cypress):** Simulam a jornada completa de um usuário final através da aplicação.
        - **Aplicação:** Validar o fluxo completo: (1) Fazer login, (2) receber um token, (3) usar o token para acessar a página de veículos, (4) criar um novo veículo, (5) editar e (6) excluir o veículo, interagindo com a interface gráfica.
- **Abordagem de Teste:** Será utilizada uma abordagem baseada em risco, priorizando o teste das funcionalidades de autenticação/autorização e dos cenários de CRUD, que são críticos para o sistema. Testes de regressão serão aplicados continuamente para garantir que novas implementações não quebrem funcionalidades existentes.

---

3. Recursos e Ambiente de Teste

- **Recursos Humanos:**
    - Equipe de Desenvolvimento do Projeto.
- **Ambiente de Teste:**
    - **Sistema Operacional:** Windows, macOS ou Linux.
    - **Software Requerido:**
        - Node.js (versão 18.x ou superior).
        - Navegador Web (Google Chrome).
        - Visual Studio Code.
        - Servidor de Banco de Dados (Ex: PostgreSQL, MySQL, ou SQLite para desenvolvimento).
        - Cliente de API (Postman ou Insomnia) para testes manuais da API.
- **Ferramentas de Teste:**
    - **Jest:** Framework de testes unitários e de integração.
    - **Supertest:** Biblioteca para testar APIs HTTP no Node.js.
    - **Cypress:** Framework de testes de ponta a ponta.
    - **Git/GitHub:** Sistema de controle de versão.

---

4. Casos de Teste Detalhados

ID do Teste	Nível do Teste	Funcionalidade Testada	Pré-Condições	Passos para Execução	Resultado Esperado
TU-001	Unidade (Backend)	Validação do Middleware `auth` (Token Ausente)	Nenhuma	1. Simular um objeto `req` sem o cabeçalho `authorization`. 2. Chamar a função do middleware `auth` com o `req` simulado.	A função deve chamar `res.status(401)` com uma mensagem de erro indicando "token ausente ou inválido".
TU-002	Unidade (Backend)	Validação de Ano de Veículo (Inválido)	Nenhuma	1. Chamar a função de serviço de criação de veículo com o campo `year` como um valor futuro (Ex: 2050).	A função deve retornar um erro de validação indicando que o ano é inválido.
TI-001	Integração (API)	**Criação de Veículo com Token Válido**	Backend em execução; usuário autenticado com um token JWT válido.	1. Enviar uma requisição `POST` para `/api/cars` com um JSON válido para um novo veículo. 2. Incluir o cabeçalho `Authorization: Bearer <seu_token_jwt>`.	A API deve responder com status `201 Created` e o corpo da resposta deve conter o objeto do veículo criado, incluindo um `userId` correspondente ao do token.
TI-002	Integração (API)	**Tentativa de Criação de Veículo sem Token**	Backend em execução.	1. Enviar uma requisição `POST` para `/api/cars` com um JSON válido. 2. **Não** incluir o cabeçalho `Authorization`.	A API deve responder com status `401 Unauthorized` e uma mensagem de erro apropriada ("Token ausente...").
TI-003	Integração (API)	**Tentativa de Criação de Veículo com Placa Duplicada**	Backend em execução; um veículo com a placa "ABC-1234" já existe no banco de dados.	1. Enviar uma requisição `POST` para `/api/cars` tentando criar um novo veículo com a mesma placa "ABC-1234". 2. Incluir um token de autorização válido.	A API deve responder com status `400 Bad Request` (ou 500, dependendo do tratamento de erro) e uma mensagem indicando "Validation error" ou "Placa já cadastrada".
E2E-001	Ponta a Ponta	**Fluxo Completo de Criação de Veículo**	Aplicação completa (frontend e backend) em execução.	1. Abrir o navegador e navegar para a página de login. 2. Preencher as credenciais e fazer login. 3. Clicar no botão "Adicionar Novo Veículo". 4. Preencher o formulário com dados válidos (Ex: Placa: "NEW-5678"). 5. Clicar em "Salvar".	A aplicação deve redirecionar para a listagem de veículos, e o carro com a placa "NEW-5678" deve aparecer na lista.
E2E-002	Ponta a Ponta	**Acesso a Rota Protegida sem Login**	Aplicação completa em execução.	1. Limpar o armazenamento do navegador (local storage/cookies) para remover o token. 2. Tentar navegar diretamente para a URL de cadastro de veículos (Ex: `/dashboard/cars/new`).	A aplicação deve redirecionar o usuário para a página de login.
E2E-003	Ponta a Ponta	**Exclusão de Veículo**	Aplicação completa em execução; deve haver pelo menos um veículo na lista.	1. Fazer login na aplicação. 2. Na lista de veículos, localizar um carro. 3. Clicar no botão "Excluir". 4. Confirmar a exclusão.	O veículo deve ser removido da lista na interface e do banco de dados.

---

5. Cronograma e Entregáveis

- **Cronograma (Exemplo para um Ciclo de 2 Semanas):**

    - **Semana 1:** Escrita e execução dos testes de unidade para o backend (validações, middleware) e dos testes de integração da API (CRUD e rotas de autenticação).
    - **Semana 2:** Escrita e execução dos testes de ponta a ponta (E2E) para os fluxos críticos do usuário. Análise dos resultados, registro de defeitos e retestes das correções.

- **Entregáveis:**

    - **Documento de Plano de Testes:** Este próprio documento.
    - **Scripts de Teste:** Código-fonte de todos os testes (unidade, integração, E2E) versionado no GitHub.
    - **Relatórios de Execução de Testes:** Saídas geradas pelo Jest e relatórios/vídeos gerados pelo Cypress.
    - **Relatório de Defeitos:** Registro formal dos bugs encontrados no sistema de issues do projeto (Ex: GitHub Issues).

---

6. Conclusão e Aprendizados

A execução deste plano de testes será fundamental para garantir a qualidade e a segurança do "Sistema de Gerenciamento de Veículos". Através da aplicação de múltiplos níveis de teste, a equipe poderá identificar e corrigir falhas de forma precoce, desde a lógica de negócio até a experiência do usuário.

Os principais aprendizados esperados com este ciclo de testes incluem:

- **Importância da Segurança em Primeiro Lugar:** Os testes de integração focados em autenticação e autorização irão demonstrar a criticidade de proteger os dados e funcionalidades do sistema.
- **Valor da Validação de Dados:** Os testes que simulam entradas inválidas (placas duplicadas, dados ausentes) reforçarão a necessidade de validações robustas no backend para manter a integridade do banco de dados.
- **Confiança na Regressão:** Com uma suíte de testes automatizados, a equipe ganhará confiança para adicionar novas funcionalidades ou refatorar o código existente, sabendo que os testes irão alertar sobre qualquer quebra inesperada no sistema.
- **Melhora da Comunicação:** O processo de registrar e discutir os bugs encontrados irá aprimorar a comunicação e a colaboração dentro da equipe, resultando em um produto final mais estável e confiável.