SZX CARS

Documento de Especificação de Requisitos

Histórico de revisões do Documento

Versão (XX.YY)	Data (DD/MMM/YYYY)	Autor	Descrição

Índice

1. Introdução

1.1. Propósito do Documento de Requisitos

Este documento tem como propósito apresentar de forma organizada os requisitos para o desenvolvimento de um sistema de Catálogo Online de Veículos de uma concessionária. A ideia é registrar as necessidades e expectativas tanto da empresa quanto dos usuários, servindo como referência para orientar o trabalho da equipe de desenvolvimento.

O catálogo online deverá funcionar como uma vitrine digital, onde os clientes poderão visualizar os veículos disponíveis, acessar informações detalhadas de cada modelo. Além disso, o sistema também deverá oferecer ferramentas para que os administradores da concessionária possam gerenciar os veículos cadastrados.

1.2. Público Alvo

O sistema de Catálogo Online de Veículos tem como público-alvo clientes da concessionária e administradores da concessionária

1.3. Definições, Acrônimos e Abreviações.

- ADM – Administrador: Usuário responsável por gerenciar o catálogo de veículos.
- Catálogo Online: Vitrine digital com os veículos disponíveis para os clientes.
- Veículo Disponível: Item cadastrado no catálogo que ainda está à venda.
- Login Administrativo: Autenticação usada pelos administradores para acessar o sistema.

2. Descrição Geral do Produto

Esta seção do documento descreve o objetivo do sistema, suas respectivas funcionalidades, qual o público alvo do sistema, qual a necessidade de implementar o produto, o impacto do sistema e sucesso que a solução irá trazer.

2.1. Situação Atual

Atualmente, a concessionária não possui um catálogo digital estruturado para divulgação de seus veículos.
Os clientes dependem de atendimento presencial ou de contatos manuais (telefone, WhatsApp) para obter informações sobre modelos, preços e disponibilidade.
Esse processo gera dificuldades como:

- *Desatualização frequente das informações de veículos disponíveis.*
- *Falta de padronização na apresentação de dados e imagens.*
- *Dificuldade de acesso remoto por parte dos clientes.*
- *Alto custo de tempo para que administradores atualizem os dados manualmente.*

2.2. Objetivos do Produto

*O sistema de **Catálogo Online de Veículos** tem como objetivo:*

- *Criar uma vitrine digital de veículos acessível a qualquer hora, em diferentes dispositivos (computadores, tablets e smartphones).*
- *Permitir que clientes consultem informações detalhadas (modelo, ano, preço, características técnicas e imagens) de cada veículo.*
- *Fornecer ferramentas administrativas para cadastro, edição e exclusão de veículos disponíveis.*
- *Garantir segurança na autenticação dos administradores, protegendo dados internos da concessionária.*
- *Melhorar a eficiência do processo de venda, reduzindo a dependência de atendimentos presenciais e aumentando a satisfação do cliente.*

2.3. Benefícios do Projeto

- ***Maior alcance de clientes:** acesso remoto e 24/7 ao catálogo de veículos.*
- ***Atualização em tempo real:** administradores podem gerenciar informações com rapidez.*
- ***Flexibilidade:** acessível em múltiplas plataformas (web, mobile).*

- ***Padronização:*** *informações uniformes sobre todos os veículos.*
- ***Segurança:*** *autenticação criptografada e ambiente protegido com firewall e SSL.*
- ***Eficiência operacional:*** *redução da sobrecarga de atendentes e maior autonomia para o cliente.*

2.4. Escopo

*O projeto envolve o desenvolvimento de um **sistema web** hospedado em ambiente online, acessível a clientes e administradores.*
O escopo inclui as seguintes macro-funcionalidades:

Nº	Módulo	Descrição
1	Catálogo	Catálogo online para exibição de veículos.
2	Informações veiculos	Consulta detalhada de cada veículo (informações técnicas, fotos e preço).
3	Login	Sistema de autenticação para administradores.
4	Cadastro de veículos	Painel administrativo para cadastro, edição e exclusão de veículos.
5	Compatibilidade	Compatibilidade com navegadores modernos (Chrome, Firefox, Edge, Safari).

2.5. Atores

Nº	Ator	Definição e Privilégio de Acesso e Segurança
1	*Cliente*	*Usuário externo que acessa o catálogo online, podendo visualizar veículos disponíveis, consultar informações detalhadas (modelo, ano, preço, características) e imagens. Não possui privilégios de alteração de dados.*
2	*Administrador*	*Funcionário da concessionária responsável pelo gerenciamento do sistema. Possui privilégios de acesso ao painel administrativo, cadastro, edição e exclusão de veículos, além do controle de login.*
3	*Sistema de Catálogo*	*Sistema web que armazena e disponibiliza as informações dos veículos. Responsável por garantir a autenticação, segurança dos dados e disponibilidade 24/7.*

2.6. Premissas

- *Será utilizada uma metodologia de desenvolvimento ágil.*
- *O sistema deve estar acessível em computadores, notebooks, tablets e smartphones.*
- *O cliente (concessionária) fornecerá as informações e imagens dos veículos.*
- *O sistema será desenvolvido em ambiente seguro, com suporte a SSL e firewall.*
- *As atualizações de catálogo deverão ocorrer em tempo real.*

2.7. Itens Fora do Escopo

- *Integração com sistemas legados da concessionária.*
- *Desenvolvimento de aplicativo nativo para Android/iOS (será apenas web responsivo).*
- *Gestão financeira detalhada (controle de pagamentos e faturamento não estão incluídos).*
- *Suporte multilíngue (sistema será apenas em português).*
- *Recursos avançados de testes de desempenho, carga e estresse.*

3. Requisitos Específicos

Aqui serão descritos os requisitos funcionais e não funcionais do sistema a ser implementado. Os requisitos, em geral, refletem funções que o usuário precisa realizar para atingir o objetivo do sistema ou funções de apoio à estratégia do negócio. Registros, controle de fluxo, consultas e cadastros são requisitos típicos. Em geral, requisito é algo que o usuário solicita explicitamente (ou requisita)

3.1. Requisitos Funcionais

ID	Descrição
RF 01	*Mostrar os veículos em forma de catálogo.*
RF 02	*Oferecer filtros de busca (marca, modelo, ano, preço).*
RF 04	*Exibir informações detalhadas de cada veículo.*
RF 04	*Permitir que o administrador cadastre e edite veículos.*
RF 05	*Ser responsivo, funcionando bem em computadores e celulares.*

3.2. Requisitos Não Funcionais

ID	Descrição	Categoria
RNF 1	Login Criptografado	*Segurança*
RNF 2	Rodar bem nos navegadores mais usados.	*padrões*
RNF 3	*Responder rápido, com tempo de carregamento de até 2 segundos.*	*padrões*
RNF 4	*Ter um design acessível e fácil de usar.*	*Interface*
RNF 5	*Ser feito em HTML, CSS e JavaScript*	*Produto (software)*

3.3. Regras de Negócio

ID	Regra
RN 1	*Todo veículo que for cadastrado precisa ter as informações básicas: marca, modelo, ano, preço e pelo menos uma foto.*
RN 2	*O preço sempre deve ser um valor válido (não pode ser zero ou negativo).*
RN 3	*Somente o administrador terá permissão para cadastrar, editar ou remover veículos do catálogo.*

RN 4	*Qualquer pessoa poderá visualizar os veículos sem precisar de login, mas apenas o administrador poderá fazer alterações.*
RN 5	*Os veículos devem ser separados em categorias (como carros, motos e utilitários) para facilitar a navegação do usuário.*
RN 6	*O sistema deve oferecer uma busca e filtros que funcionem sobre os principais dados: marca, modelo, ano e preço.*
RN 7	*Não pode haver duplicidade: dois veículos exatamente iguais (mesma marca, modelo, ano e preço) não devem ser cadastrados.*

3.4. Restrições de Hardware

- O sistema deve ser acessível em computadores, notebooks, tablets e smartphones.
- Requer conexão estável com a internet. - Requisitos mínimos sugeridos:
• Processador: Dual Core ou superior
• Memória RAM: 4 GB ou mais
• Armazenamento: 500 MB de espaço livre para cache e arquivos temporários
• Resolução de tela: mínima de 1280x720

3.5. Restrições de Software

- *O sistema deve ser compatível com navegadores modernos (Google Chrome, Mozilla Firefox, Microsoft Edge e Safari).*
- *Necessidade de suporte a HTML5, CSS3 e JavaScript.*
- *Requer autenticação segura (criptografia de senhas).*

3.6. Restrições de Ambiente

- *O sistema será hospedado em ambiente web, acessível 24/7.*
- *Deve suportar múltiplos acessos simultâneos de clientes e administradores.*
- *Necessidade de ambiente seguro, com firewall e certificado SSL para proteção dos dados*

3.7. Lista de Riscos

Nº	Descrição	Mitigação
1	Falta de conhecimento da equipe em tecnologias web (HTML5, CSS3, JS, PHP).	Oferecer treinamentos ou consultar documentação oficial.
2	Instabilidade na hospedagem do sistema (quedas de servidor).	Contratar serviço de hospedagem com alta disponibilidade e backup.
3	Vulnerabilidade na segurança dos dados (senhas e informações pessoais).	Implementar criptografia de senhas, SSL e firewall.

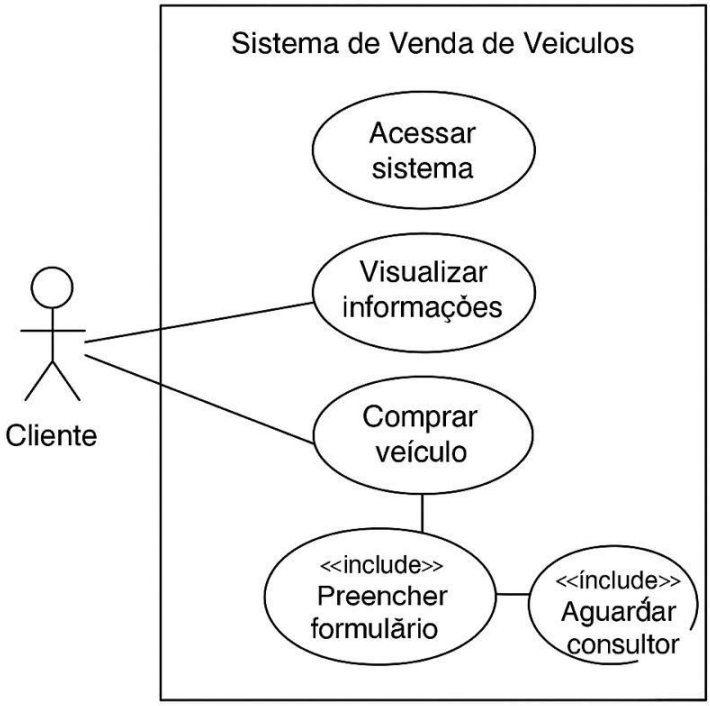
4	Problemas de expansividade em dispositivos móveis.	Testar e aplicar design responsivo compatível com múltiplos dispositivos.
5	Erros no cadastro ou atualização de veículos pelo administrador.	Criar validações de entrada e permitir edição/correção de dados.
6	Desempenho lento em múltiplos acessos simultâneos.	Otimizar banco de dados e usar balanceamento de carga.
7	Perda de dados em caso de falha no sistema.	Implementar backups automáticos e periódicos do banco de dados.

4.1. Identificação dos Casos de Uso

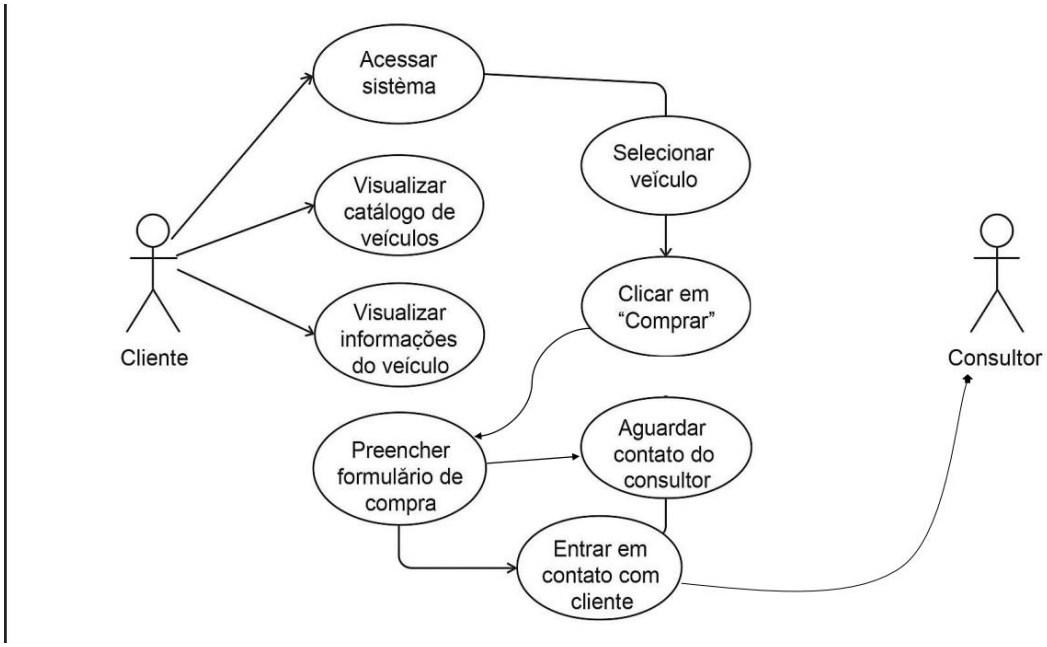
ID	Nome do Caso de Uso	Atores Envolvidos	Descrição
UC01	Acessar sistema	Cliente	Cliente acessa o sistema para iniciar o processo.
UC02	Visualizar catálogo de veículos	Cliente	Cliente navega pelo catálogo de veículos disponíveis.
UC03	Visualizar informações do veículo	Cliente	Cliente acessa detalhes específicos de um veículo selecionado.

UC04	Selecionar veículo	Cliente	Cliente escolhe um veículo de interesse.
UC05	Clicar em “Comprar”	Cliente	Cliente manifesta interesse em comprar o veículo.
UC06	Preencher formulário de compra	Cliente	Cliente preenche os dados necessários para iniciar a compra.
UC07	Aguardar contato do consultor	Cliente	Cliente aguarda que um consultor entre em contato após a solicitação.
UC08	Entrar em contato com cliente	Consultor	Consultor realiza o contato com o cliente interessado.

4.2. Diagrama de Casos de Uso



4.3. Diagrama de Atividades



5. Projeto

SITE https://concessionaria-automax.onrender.com/

REPOSITORIO https://github.com/Isaaclucaspfr/concessionaria-automax

ACESSANDO A AREA ADMINISTRATIVA

PASSO 1: Acessar a pagina de login

- No site, clique no botão "Login" no canto superior direito

PASSO 2: Fazer login com as credenciais

- Usuário: Isaaclucaspfr
- Senha: teste123

PASSO 3: Clicar em "Entrar"

- Você será redirecionado para o painel administrativo

Arquitetura do Software — Concessionária AutoMax

Última atualização: 2025-11-17

Este documento descreve a arquitetura do sistema desenvolvido no repositório `concessionaria-automax` (frontend React + backend Node.js simples).

Sumário

Visão Geral

O sistema é uma aplicação web para uma concessionária, com interface React no `frontend/` e um backend Node.js mínimo em `backend/`. O frontend consome dados do backend (mock de carros) e apresenta páginas como `Homepage`, `Carros`, `Admin` e `Login`.

Objetivo do documento: ser referência técnica para desenvolvedores e permitir conversão direta para PDF.

Contrato do Sistema (resumo)

- Inputs:
  - Requisições HTTP do frontend para o backend (GET/POST conforme necessidade).
  - Ações do usuário via UI (login, cadastro de carros, navegação).
- Outputs:
  - Páginas HTML servidas pelo frontend (via dev server / build estático).
  - Respostas JSON do backend contendo listas de carros, detalhes e mensagens de sucesso/erro.
- Critério de sucesso:
  - UI apresenta corretamente os dados do backend e permite operações administrativas simples.

- Modos de erro:
  - Backend indisponível → frontend deve tratar erros e mostrar mensagem amigável.
  - Dados inválidos no frontend → validação básica de campos antes de enviar.

Tecnologias e Estrutura do Repositório

- Frontend: React (create-react-app), código em `frontend/src/`.
  - Principais arquivos: `App.js`, `index.js`, `index.css`.
  - Páginas: `frontend/src/pages/*` (Homepage, Carros, Admin, Login ...)
  - Componentes: `frontend/src/components/*` (Header, Footer, CarImage...)
- Backend: Node.js simples, arquivo `backend/server.js`.
  - Dados mock: `backend/data/carros-mock.js`.
- Gerenciamento de versão: Git (repositório remoto no GitHub em `origin/main`).
- Imagens e ativos: `frontend/public/images/`.

Componentes Principais

Frontend

- `Header` (`frontend/src/components/Header.js`): navegação e botão de login.
- `Homepage` (`frontend/src/pages/Homepage.js`): hero e botão WhatsApp.
- `Carros` (`frontend/src/pages/Carros.js`): listagem de carros e exibição de preços.
- `CarrosNovo` (`frontend/src/pages/CarrosNovo.js`): formulário para adicionar novo carro (Admin).
- `Login` (`frontend/src/pages/Login.js`): tela de autenticação (simples).

Estilo: centralizado em `frontend/src/index.css` e estilos locais por componente.

Backend

- `server.js`: servidor express minimal que serve endpoints para obter a lista de carros e operações simples (se implementadas).
- `data/carros-mock.js`: array com objetos de carros (modelo, preço, imagem, etc.).

Fluxos de Dados e Comunicação

1. Usuário abre a `Homepage` no navegador.
2. O frontend consome dados do backend via fetch/axios (GET `/carros` ou endpoint correspondente no `server.js`).
3. Dados retornados em JSON são mapeados para componentes React e renderizados.
4. Operações administrativas (ex.: adicionar carro) chamam endpoints POST/PUT no backend ou atualizam um mock local conforme implementação.

Modelos de Dados (exemplo simplificado)

Carro (exemplo):

- id: string | number
- marca: string
- modelo: string
- ano: number
- preco: number
- cor: string
- imagem: string (path relativo em `public/images/cars/`)

Endpoints HTTP (resumo esperado)

- GET /api/carros — retorna lista de carros
- GET /api/carros/:id — retorna detalhes de um carro
- POST /api/carros — cria novo carro (Admin)
- PUT /api/carros/:id — atualiza carro
- DELETE /api/carros/:id — remove carro

Observação: adapte as rotas conforme implementação real em `backend/server.js`. Verificar o arquivo para confirmar nomes exatos.

Requisitos Não-Funcionais

- Performance: páginas estáticas servidas rápido; carregar imagens otimizadas.
- Escalabilidade: arquitetura monolítica simples; para alto tráfego, migrar backend para serviços e adicionar cache (Redis).
- Portabilidade: frontend padrão React, pode ser hospedado como SPA em qualquer host estático.
- Testabilidade: testes unitários em `frontend/src/App.test.js` (já existe scaffold do CRA).

Segurança

- Autenticação: atualmente básica (verificar `Login.js`); para produção, adicionar JWT/HTTPS e backend com persistência segura.
- Validação de entrada: validar no frontend e no backend.
- Armazenamento de segredos: usar variáveis de ambiente (`frontend/.env` existe localmente). Nunca commitar segredos reais.

Deploy e Execução Local

Execução local (frontend):

1. Abrir terminal na pasta `frontend/`.
2. Instalar dependências: `npm install` (se necessário).
3. Rodar em dev: `npm start`.

Backend:

1. Abrir terminal na pasta `backend/`.
2. Instalar dependências: `npm install` (se houver `package.json`).
3. Rodar: `node server.js` ou `npm start`.

Pontos de melhoria / Próximos passos

- Documentar contratos de API com OpenAPI/Swagger.
- Substituir mocks por um banco de dados (ex.: SQLite ou MongoDB) para persistência.
- Adicionar testes de integração para endpoints do backend.
- Introduzir CI (GitHub Actions) para lint, build e testes automáticos.
- Planejar implantação: Dockerfile para backend e build do frontend para host estático.

Referências (no repositório)

- Frontend: `frontend/src/`
- Backend: `backend/server.js`, `backend/data/carros-mock.js`
- Estilos globais: `frontend/src/index.css`