# 1. Introdução

• Nome do aluno(ª): Jarom Teodoro Medeiros Nascimento

• Matrícula: RA: 00285267

• Nome do projeto: AutoList - Sistema de Gestão de Clientes

• Objetivo: O sistema tem como objetivo principal gerenciar o cadastro de clientes para uma loja de veículos. Ele resolve o problema da organização e consulta de informações de clientes e dos veículos de seu interesse, utilizando uma API externa para obter dados atualizados dos automóveis.

• Público-alvo: Administradores e vendedores da loja de veículos, que precisam de uma ferramenta para registrar e consultar informações de clientes e propostas de venda.

# 2. Requisitos do Sistema

## 2.1 Funcionais

• [X] O sistema deve permitir autenticação (login e senha).

• [X] O sistema deve realizar CRUD das entidades principais (clientes).

• [X] O sistema deve se conectar a uma API externa para obter informações (API FIPE de veículos).

• [X] O sistema deve armazenar dados em um banco de dados.

## 2.2 Não Funcionais

• Desenvolvido em: PHP

• Banco de dados utilizado: MySQL

• Frameworks/Bibliotecas: Bootstrap 5 (para o front-end) e Font Awesome (para ícones).

• [X] O sistema deve ser simples, intuitivo e de fácil uso.

# 3. Funcionalidades do Sistema

• Cadastro de usuários: O sistema possui um sistema de autenticação com usuários pré-definidos (admin, vendedor), mas não uma tela para criar novos usuários dinamicamente.

• Login: Acesso ao sistema somente com credenciais válidas (usuário e senha). O sistema diferencia os níveis de acesso.

• Gerenciamento de clientes: CRUD completo (cadastrar, listar, editar e excluir) para os registros de clientes.

• Consulta à API externa: O sistema consulta a API FIPE para buscar marcas, modelos e valores de veículos durante o cadastro e edição de clientes, exibindo as informações na tela.

# 4. Tecnologias Utilizadas

• Linguagem de programação: PHP 8

• Banco de dados: MySQL

• APIs utilizadas: API FIPE da Parallelum (https://parallelum.com.br/fipe/api/v1/carros/)

• Outras ferramentas: Visual Studio Code (Editor de código), XAMPP/MAMP (Ambiente de desenvolvimento local com Apache e MySQL).

# 5. Instruções de Execução

1. Fazer o download/clonar o repositório do projeto.

2. Criar o banco de dados: Em seu ambiente MySQL, crie um novo banco de dados com o nome exato de autolist.

3. Configurar a conexão com o banco de dados: Abra o arquivo conexao.php. As variáveis de conexão já estão definidas, mas você deve verificar se elas correspondem ao seu ambiente local. O arquivo está configurado da seguinte forma:

$localhost = "localhost";  
$username = "root";  
$password = "root"; // (MAMP usa "root", XAMPP geralmente usa a senha em branco "")  
$port = 8889; // Porta MySQL (MAMP padrão)  
$database = "autolist";

Observação: conforme a configuração exibida no MAMP, as portas utilizadas são:  
• Apache Port: 8888  
• MySQL Port: 8889

4. Executar o projeto: Este projeto não utiliza um comando de inicialização. Simplesmente coloque a pasta do projeto no diretório raiz do seu servidor web (por exemplo, a pasta htdocs do XAMPP ou MAMP).

5. Acessar no navegador:  
• Se estiver usando MAMP com as portas configuradas da forma padrão exibida (Apache 8888 e MySQL 8889), acesse: http://localhost:8888/sua-pasta/  
• Se estiver usando XAMPP (Apache 80 e MySQL 3306), acesse: http://localhost/sua-pasta/

# 6. Conclusão

• Breve resumo do que foi aprendido com o desenvolvimento do sistema: O desenvolvimento deste projeto permitiu aplicar conceitos essenciais de programação web, como a criação de um sistema CRUD (Create, Read, Update, Delete) com PHP, a interação com um banco de dados MySQL utilizando queries SQL, a implementação de um sistema de autenticação e gerenciamento de sessões, e a integração com uma API REST de terceiros para consumir dados externos.

• Possíveis melhorias futuras que poderiam ser implementadas:  
o CRUD de Usuários: Criar uma área administrativa para gerenciar os usuários do sistema (adicionar, editar, remover).  
o Segurança Aprimorada: Implementar validação de dados mais robusta e proteção contra ataques comuns, como CSRF.  
o Relatórios: Desenvolver um módulo para gerar relatórios de vendas e clientes.  
o Interface mais dinâmica: Utilizar JavaScript ou um framework como Vue.js para criar uma experiência de usuário mais rica e interativa, sem a necessidade de recarregar a página a cada ação.