FIAP

Faculdade de Informática e Administração Paulista

1TDSPK

Larissa de Freitas. RM: 555136

Luis Carlos Moreira Duarte. RM: 555181

Maria Eduarda Alves da Paixão. RM: 558832

DiagnosticMachine

São Paulo

2024

Sumário

1. Objetivo e escopo do projeto

O projeto visa revolucionar o setor automobilístico, otimizando o tempo dos usuários e facilitando o diagnóstico de problemas em veículos através de um chatbot intuitivo. À medida que a demanda por reparos automotivos cresce, o conhecimento técnico dos proprietários de veículos tende a diminuir, gerando incertezas e atrasos nos diagnósticos e orçamentos. Para enfrentar esses desafios, o projeto combina tecnologia de ponta e inteligência artificial para criar uma solução inovadora e acessível.

Apresentando **Carlos**, o assistente virtual que proporciona uma experiência interativa e humanizada aos usuários. Por meio de uma conversa simples e descontraída, Carlos identifica o problema no veículo e fornece um orçamento preciso, reduzindo o tempo de espera e aumentando a precisão do diagnóstico. Além disso, o sistema inclui uma seção "Ajuda e Aprenda+" que permite aos usuários adquirirem conhecimentos sobre manutenção automotiva, promovendo o engajamento contínuo com a plataforma, mesmo quando não enfrentam problemas imediatos com seus veículos.

O objetivo do projeto é não apenas resolver problemas rapidamente, mas também educar e empoderar os usuários, tornando a manutenção automotiva uma tarefa menos intimidante e mais acessível. Comprometendo-se a criar uma experiência imersiva e fácil de usar, que não só resolve problemas, mas também enriquece o conhecimento e a confiança dos usuários no cuidado com seus veículos.

1.1. Objetivo e escopo da API desenvolvida.

A API, desenvolvida com o framework Jersey e gerenciada por Maven, é um dos pilares estruturais do projeto, fornecendo a base para as operações CRUD (criação, leitura, atualização e exclusão) que possibilitam o funcionamento dinâmico e eficaz do sistema. Com essa API, todos os processos de autenticação e gerenciamento de dados dos usuários e de seus veículos são

realizados de forma robusta e segura. Cada funcionalidade foi cuidadosamente projetada para garantir uma interação fluida entre o usuário e o sistema, desde o momento em que o usuário se cadastra.
Além disso, a API desempenha um papel fundamental na comunicação com o banco de dados, garantindo que as informações estejam sempre atualizadas e acessíveis. O uso do Jersey, aliado ao gerenciamento de dependências do Maven, facilita a escalabilidade e manutenção da API, permitindo que futuras implementações e melhorias sejam feitas de maneira ágil.

2. Descrição das principais funcionalidades

- **Cadastro de Usuários**: Esta funcionalidade é essencial para identificar os usuários que estão enfrentando problemas, permitindo a criação de um histórico dinâmico sempre que o usuário acessar o sistema. Os atributos obrigatórios para o cadastro são: nome, telefone, data de nascimento, e-mail e senha. A data de nascimento será utilizada para validar se o usuário é maior de idade. O e-mail e o telefone devem ser únicos para cada usuário.

- **Cadastro de Veículos**: Após a conclusão do cadastro do usuário, a área de cadastramento de veículos será habilitada. Um usuário deve ter pelo menos um veículo cadastrado para utilizar a principal funcionalidade do sistema, que é o CHATBOT. Os atributos necessários para o cadastro de veículos incluem: modelo, ano, marca e placa. Sendo a placa um valor único e que não se repete entre os veículos dos clientes.

- **Login:** Esta funcionalidade é destinada a usuários previamente cadastrados que desejam validar seus perfis e acessar a gama de funcionalidades oferecidas pelo sistema. Os atributos necessários para o login são: e-mail e senha. É importante ressaltar que se o usuário não tiver um veículo associado a sua conta não será

possível fazer o login, então o sistema redireciona-o novamente para o cadastro de veículos.

- **Atualizar dados:** Essa funcionalidade é essencial para garantir uma experiência contínua e personalizada no sistema. Os usuários têm a flexibilidade de atualizar informações importantes, como nome, telefone e data de nascimento, conforme necessário. Ao permitir que mantenham seus dados sempre atualizados, o sistema Diagnostic Machine promove uma interação mais precisa e satisfatória, ajustando-se às necessidades individuais dos usuários e fortalecendo seu engajamento com a plataforma.

- **Atualizar foto:** Para proporcionar uma experiência mais humanizada e personalizada, o sistema permite que o usuário substitua a foto padrão por uma imagem própria. Com essa opção, cada usuário pode tornar sua interação no diagnosticMachine mais pessoal, refletindo sua identidade e criando uma conexão mais próxima com a plataforma.

- **Deletar conta:** O usuário tem o direito de excluir sua conta e apagar permanentemente seus dados do nosso banco de dados. Para confirmar essa exclusão, o sistema no front-end solicita uma confirmação do usuário. Após a confirmação, a API conecta-se ao banco de dados e realiza a exclusão completa dos dados, removendo o usuário, o veículo associado e todas as relações entre essas entidades. Esse processo garante a remoção segura e total das informações, respeitando a privacidade e o controle do usuário sobre seus dados.

- **Exibir dados do usuário**: Essa funcionalidade permite que o usuário visualize todas as informações pessoais cadastradas no sistema, como nome, telefone, data de nascimento e email. Ao facilitar o acesso rápido e direto a esses dados, o Diagnostic

Machine promove uma experiência transparente e prática, permitindo que o usuário confira e gerencie suas informações sempre que necessário.

- **Exibir dados do veículo:** Com essa funcionalidade, o usuário pode visualizar todos os dados registrados sobre o veículo, como modelo, marca, ano e placa. Esse recurso é essencial para que o usuário acompanhe e verifique as informações do veículo no sistema, assegurando que estejam sempre corretas para um diagnóstico preciso e uma experiência de uso mais eficaz.

3. Tabela de Endpoints

URIs	Verbo HTTP	Status
Resources.UsuarioResource.java		
http://localhost:8080/usuarioresource/login	POST	200 (ok), 400 (Bad Request)
http://localhost:8080/usuarioresource/cadastroUsuario	POST	200 (ok), 409 (Conflict), 400 (Bad Request)
http://localhost:8080/usuarioresource/buscaIdUsuario/{email}	GET	200 (ok), 404 (Not Found)
http://localhost:8080/usuarioresource/exibirFoto/{email}	GET	200 (ok), 404 (Not Found)
http://localhost:8080/usuarioresource/exibirUsuario/{email}	GET	200 (ok), 404 (Not Found)
http://localhost:8080/usuarioresource/atualizaDados/{email}	PUT	200 (ok), 409 (Conflict)
http://localhost:8080/usuarioresource/atualizaFoto/{email}	PUT	200 (ok), 400 (Bad Request)
http://localhost:8080/usuarioresource/deletarUsuario/{idUser}	DELETE	201 (ok), 400 (Bad Request)
Resources.VeiculoResource.java		
http://localhost:8080/veiculoresource/buscarModelosMarcas	GET	200 (ok), 404 (Not Found)
http://localhost:8080/veiculoresource/buscaIdVeiculo/{placa}	GET	200 (ok), 404 (Not Found)
http://localhost:8080/veiculoresource/cadastroVeiculo	POST	200 (ok), 409 (Conflict), 400 (Bad Request)
http://localhost:8080/veiculoresource/associacaoUserVeiculo/{idUser}/{idVeiculo}	GET	200 (ok), 409 (Conflict), 400 (Bad Request)
http://localhost:8080/veiculoresource/exibirVeiculo/{idUser}	GET	200 (ok), 404 (Bad Request)

3.1. O que os endpoints esperam receber:

- http://localhost:8080/usuarioresource/login **Exemplo JSON:** {
  "email": "usuario@example.com",
  "senha": "minhasenha123",
  }
- http://localhost:8080/usuarioresource/cadastroUsuario
  Exemplo JSON: {
  "email": "usuario@example.com",
  "senha": "minhasenha123",
  "nome_completo": "Maria Eduarda",
  "data_nasc": "2000-05-15",
  "telefone": "123456789"
  }

- http://localhost:8080/usuarioresource/atualizaDados/{email}
  PathParam(na url): e-mail do usuário desejado para edição
  Exemplo JSON: {
  "nome_completo": "Maria Eduarda",
  "data_nasc": "2000-05-15",
  "telefone": "123456789"
  }
- http://localhost:8080/usuarioresource/atualizaFoto/{email}
  PathParam(na url): e-mail do usuário desejado para edição
  Exemplo JSON: {
  “imgURL”: “linkdafotofirebase.com.br”
  }

- http://localhost:8080/veiculoresource/cadastroVeiculo
  Exemplo JSON: {
  "ano": 2020,
  "placa": "ABC1234",
  "id_modelo_marca": 1
  }

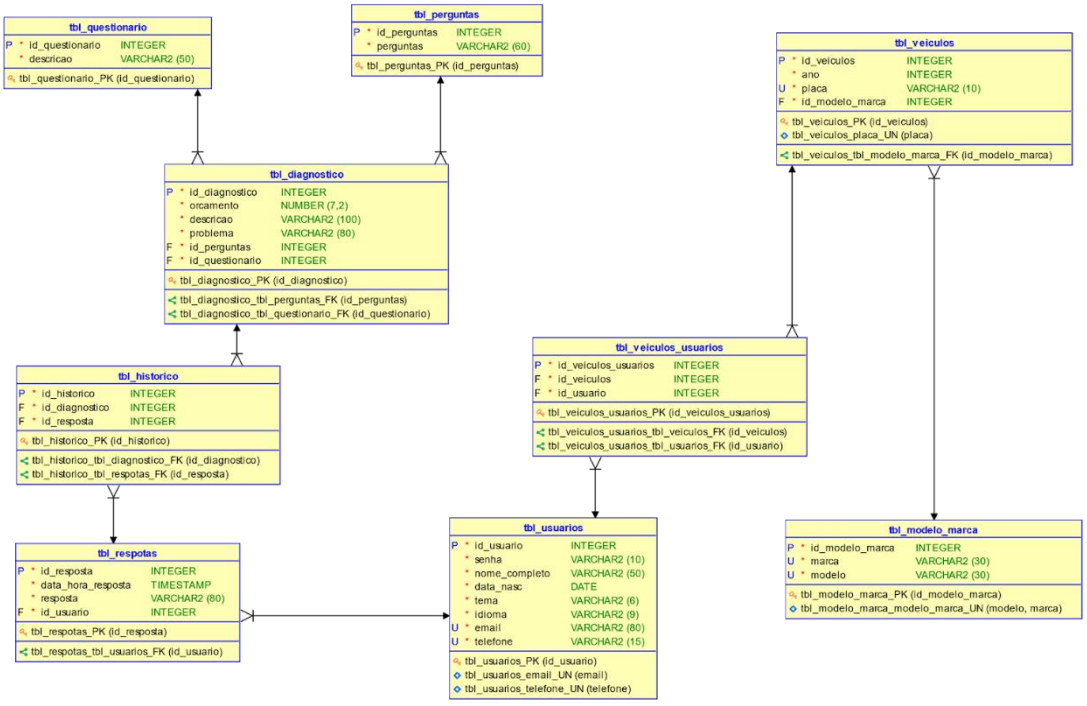
4. Protótipo

- Link do site: https://diagnostic-machine.vercel.app/

-Link do vídeo apresentando o site utilizando a API: https://drive.google.com/file/d/1iBPLB44SSM2-f6PXcg5jqYGG13XGGGRB/view?usp=sharing

-Link do repositório Java: https://github.com/MariaEdPaixao/API-BD---DiagnosticMachine

5. Modelo do banco de dados



6. Diagrama de Classes