



Faculdade de Informática e Administração Paulista

Domain Driven Design Using Java

Cauan Matos Moura Silva - 558821

Eduardo Guilherme Dias - 557886

Sérgio Henrique dos Santos Wahhab - 555901

Softesc - CAPS Bot Will

São Paulo
2024

Sumário

1 - Objetivo e Escopo.....	3
2 - Jornada do Usuário.....	3
3 - Telas do Sistema.....	4
4 - Modelo Banco de Dados.....	5
5 - Procedimentos para Rodar a Aplicação.....	6
6 - Tabela dos Endpoints.....	7
7 - Diagrama de Classes.....	8
8 - Observações Finais.....	9

1 - Objetivo e Escopo

Está em desenvolvimento uma solução para atender tanto às necessidades das pessoas que possuem carros, mas têm pouco conhecimento em mecânica para solucionar problemas quando eles surgem, quanto daquelas pessoas que possuem algum conhecimento técnico, mas enfrentam dificuldades em identificar as falhas do veículo. **E essa solução é o CAPs (Centro Automotivo Porto Seguro) Bot Will.**

O foco é diagnosticar o problema que o carro da pessoa está apresentando baseado nos relatos da mesma. Após isso, será gerado um relatório com todas as informações e valores para que essa pessoa apresente na oficina. Com isso o processo fica mais rápido e dinâmico, sem nenhuma intervenção humana, que muita das vezes causa desconfiança por parte dos clientes em relação aos mecânicos.

Tendo em vista esse cenário, o objetivo do projeto é um atendente automatizado com uma IA integrada e que será capaz de realizar diagnósticos sobre problemas de automóveis e gerar um orçamento para os usuários que o solicitarem.

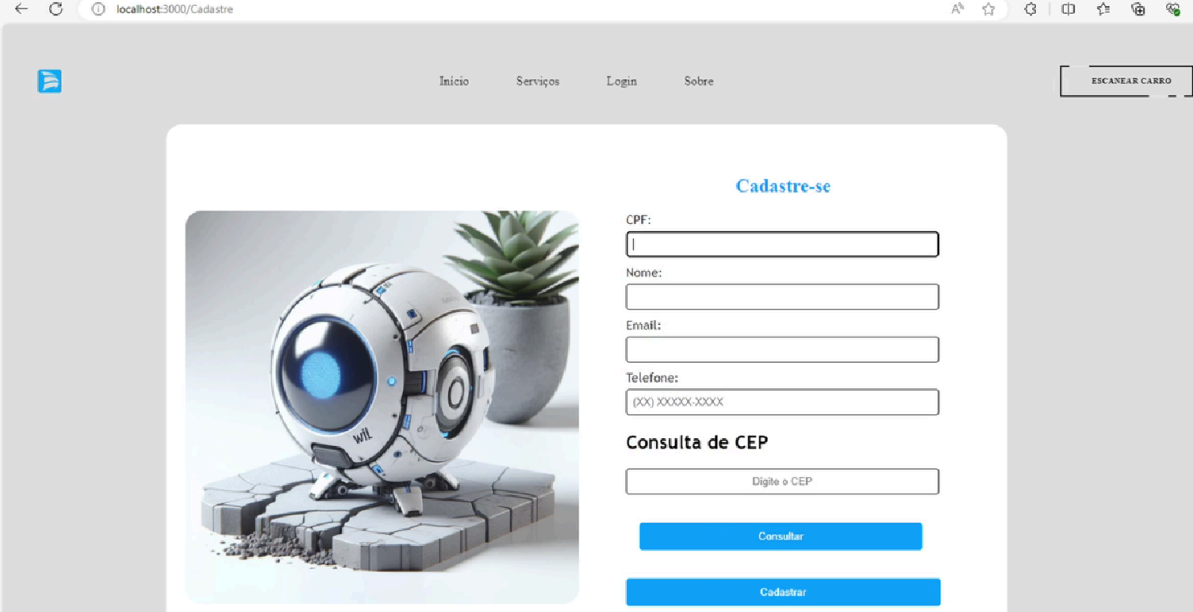
Com isso, a experiência do usuário se torna muito mais prática e segura. Uma vez que só precisa se cadastrar no site e conversar com o chatbot.

2 - Jornada do Usuário

O usuário entra no site → Acessa o Chatbot clicando no botão que vai redirecionar ele pra página de conversa → Se cadastra → Solicita um diagnóstico e fornece as informações sobre o veículo para o Bot → O bot, com base nas informações, monta um panorama de problemas que o veículo tem → Gera um orçamento com peças e valores → Bot finaliza a conversa e um programa dispara o relatório para o e-mail da pessoa.

3 - Telas do Sistema

Cadastro de Clientes:



localhost:3000/Cadastre

Início Serviços Login Sobre

ESCANEAR CARRO

Cadastre-se

CPF:

Nome:

Email:

Telefone:

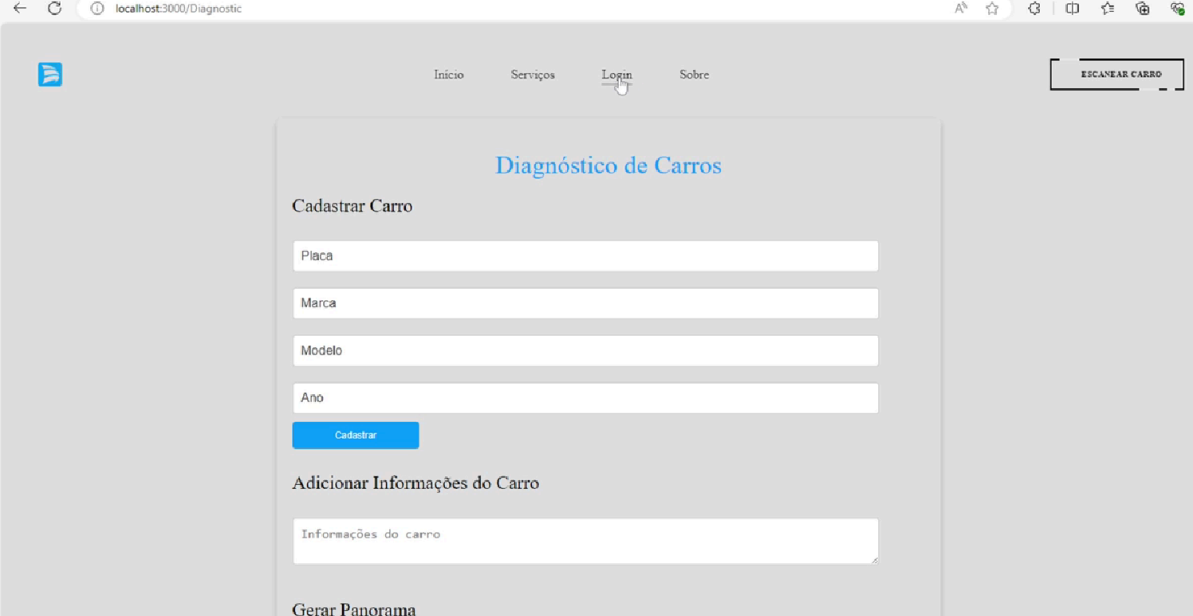
Consulta de CEP

[Consultar](#)

[Cadastrar](#)

Depois desse print, foram adicionados dois campos para o usuário preencher, sobrenome e estado, para ficar de acordo com o banco de dados que será apresentado mais adiante.

Cadastro de Veículos:



localhost:3000/Diagnostic

Início Serviços Login Sobre

ESCANEAR CARRO

Diagnóstico de Carros

Cadastrar Carro

Placa

Marca

Modelo

Ano

[Cadastrar](#)

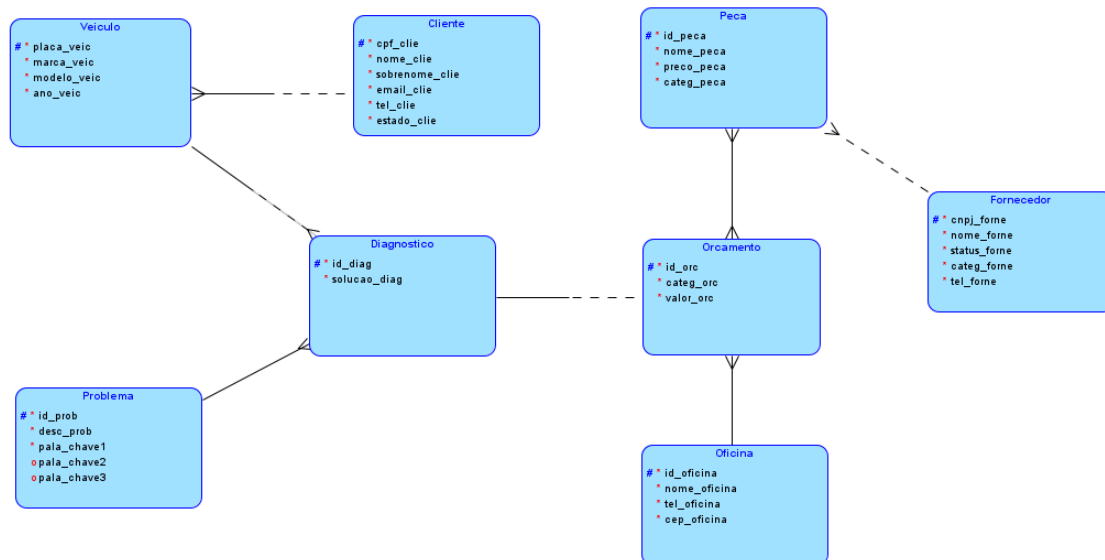
Adicionar Informações do Carro

Informações do carro

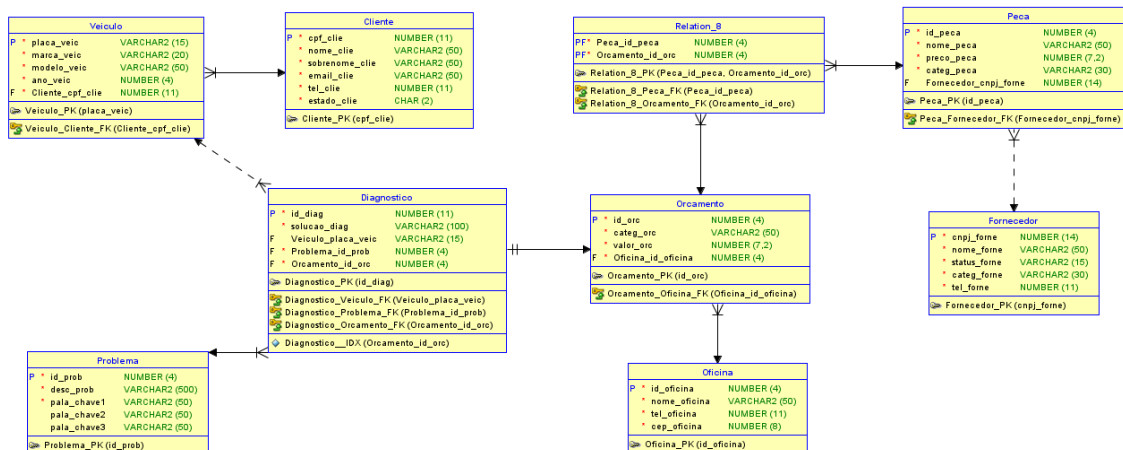
[Gerar Panorama](#)

4 - Modelo Banco de Dados

Modelo Lógico



Modelo Físico



5 - Procedimentos para Rodar a Aplicação

Para rodar a aplicação é necessário ter as credenciais do SQL Developer e as tabelas criadas.

As credenciais estão numa classe específica organizada no MVC e os comandos para a criação das tabelas estão logo abaixo.

Foram desenvolvidas as duas classes mais relevantes que o usuário terá contato ao interagir com as telas do front-end (cadastro de “Cliente” e “Veículo”).

Código SQL completo da criação das tabelas: [codigoSql](#)

```
drop table cliente cascade constraints;
```

```
drop table veiculo cascade constraints;
```

```
create table cliente (
```

```
    cpf_clie number(11) constraint clie_cpf_pk primary key,
```

```
    nome_clie varchar(50) constraint clie_nome_nn not null,
```

```
    sobrenome_clie varchar(50) constraint clie_sobrenome_nn not null,
```

```
    email_clie varchar(50) constraint clie_email_nn not null constraint clie_email_un  
unique,
```

```
    tel_clie number(11) constraint clie_tel_nn not null,
```

```
    estado_clie char(2) constraint clie_estado_nn not null);
```

```
create table veiculo (
```

```
    placa_veic varchar(15) constraint veic_placa_pk primary key,
```

```
    marca_veic varchar(20) constraint veic_marcaa_nn not null,
```

```
    modelo_veic varchar(50) constraint veic_modelo_nn not null,
```

```
    ano_veic number(4) constraint veic_ano_nn not null constraint veic_ano_ck check  
(ano_veic >= 1950 and ano_veic <= 2024));
```

6 - Tabela dos Endpoints

Página “Cliente”:

Protocolo	Endpoint	Código de Status
GET	http://localhost:8080/projetoTeste/rest/cliente	200OK
POST	http://localhost:8080/projetoTeste/rest/cliente	201 Created
PUT	http://localhost:8080/projetoTeste/rest/cliente/{cpf}	200OK
DELETE	http://localhost:8080/projetoTeste/rest/cliente/{cpf}	200OK

Página “Veiculo”:

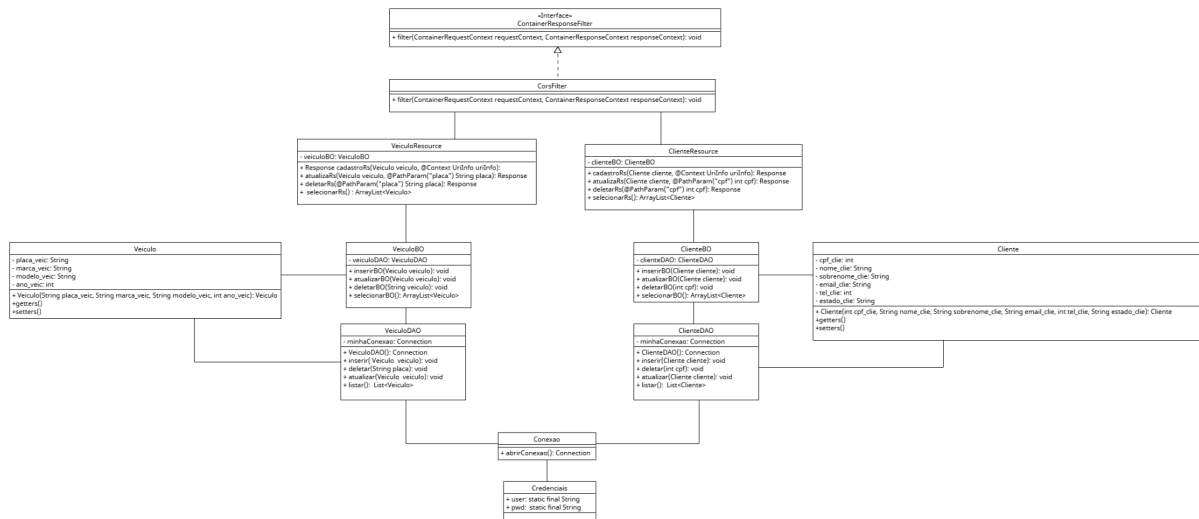
Protocolo	Endpoint	Código de Status
GET	http://localhost:8080/projetoTeste/rest/veiculo	200OK
POST	http://localhost:8080/projetoTeste/rest/veiculo	201 Created
PUT	http://localhost:8080/projetoTeste/rest/veiculo/{placa}	200OK
DELETE	http://localhost:8080/projetoTeste/rest/veiculo/{placa}	200OK

```

  ✓ Cliente
    GET http://localhost:8080/projetoTeste/rest/cliente
    POST http://localhost:8080/projetoTeste/rest/cliente
    PUT http://localhost:8080/projetoTeste/rest/cliente/123
    DEL http://localhost:8080/projetoTeste/rest/cliente/123
  ✓ Veiculo
    GET http://localhost:8080/projetoTeste/rest/veiculo
    POST http://localhost:8080/projetoTeste/rest/veiculo
    PUT http://localhost:8080/projetoTeste/rest/veiculo/abc123
    DEL http://localhost:8080/projetoTeste/rest/veiculo/abc123

```

7 - Diagrama de Classes



Veja melhor em: [DiagramaDeClasses](#)

8 - Observações Finais

Ao fazer os testes, atente-se às constraints do banco de dados, como tipo de dados, valores não nulos e primary keys diferentes.

Há duas validações no momento de inserir os dados via POST nas classes ClienteBO e VeiculoBO.

Para inserir um cliente, você precisa passar a sigla do estado e essa classe vai fazer a validação para ver se a sigla digitada está presente no vetor de Estados.

Já em VeiculoBO, você só pode inserir veículos de 1950 até 2024, caso contrário o veículo não será inserido no banco e não estará disponível para uma consulta via GET.

