Faculdade de Informática e Administração Paulista

Giovanna Revito Roz - RM558981

Kaian Gustavo de Oliveira Nascimento - RM558986

Lucas Kenji Kikuchi - RM554424

Domain Driven Design

PortoAutoTech

Sprint 4

INTEGRANTES

RM (SOMENTE NÚMEROS)	**NOME COMPLEMENTO (SEM ABREVIAR)**
558981	Giovanna Revito Roz
558986	Kaian Gustavo de Oliveira Nascimento
554424	Lucas Kenji Kikuchi

FIAP

SUMÁRIO

1 – Descrição

A plataforma PortoAutoTech visa a implementação de uma interface para auxiliar clientes decorrentes e novos clientes da Porto Seguro, especificamente aqueles que possuem pouco ou nenhum conhecimento sobre problemas relacionados a carros. Um sistema tecnológico e inovador, que resolve de maneira mais fácil. Com mais precisão. Mais rapidez. Mais facilidade. O objetivo é que, até o fim do projeto, o sistema incorpore uma interface que permita ao cliente ter:

- um autodiagnóstico do problema apresentado pelo veículo através de uma I.A treinada através do Machine Learning em Python, incorporando no Chatbot;
- a possibilidade de se cadastrar com suas informações pessoais (Nome, telefone, email etc.) e informações de seu veículo (modelo, marca, ano…) através de nosso aplicativo / website;
- a possibilidade de agendar um serviço através do Chatbot, informando o tipo de serviço, a oficina e a data;
- um pré orçamento, com um cálculo baseado no valor do serviço + valor das peças, e a estimativa de prazo de término do serviço definido com antecedência;
- a localização de oficinas mais próximas, informando a disponibilidade de peças e agendamento de determinados serviços com a integração com um banco de dados da oficina;
- mapeamento do carro, informando através de uma notificação automática quando uma manutenção preventiva deve ser feita, baseado na última vez que o cliente realizou um serviço no carro;
- notificação automática da disponibilidade de uma determinada peça, que será acionada quando a peça for registrada como disponível no banco de dados.
- interface que informa os pontos mais próximos de carregamento para carros elétricos;
- progresso da manutenção, visualizada através de uma barra de progresso, indicando cada mudança importante sobre o status da manutenção, conforme o serviço é feito (manualmente alterado pelo mecânico);
- reconhecimento de imagem através da I.A, que será responsável por analisar a imagem do problema enviado pelo usuário, retornando à solução mais plausível;

- comunicação através de voz (speech-to-text) com o Chatbot, permitindo descrever o problema oralmente ou enviar ruídos emitidos pelo carro, que serão analisados pela I.A;
- ligação em chamada com mecânico através do Chatbot, explicando o progresso da manutenção.

Através das funcionalidades descritas, o sistema será capaz de auxiliar pessoas que enfrentam dificuldades no entendimento dos problemas do veículo, além de fornecer diferenciais que auxiliarão em necessidades dos clientes que poucas empresas oferecem.

O sistema em Java, no caso, será responsável pela integração com o banco de dados, recebendo informações como dados de usuários, veículos, peças, agendamentos, diagnósticos e orçamentos.

2 – Procedimentos e Tabela de Endpoints para Rodar a Aplicação

Instalação e Configuração do Ambiente

- **Java JDK**: Verifique se o JDK está instalado e configurado. Você pode verificar executando java -version no terminal.
- **Oracle Database**: Certifique-se de que o Oracle Database está instalado e em execução.
- **IDE**: Abra o projeto em uma IDE compatível (Eclipse, IntelliJ IDEA, ou NetBeans).
- **JDBC Driver**: Verifique se o ojdbc11 está no arquivo pom.xml para conexão com o Oracle Database.
-

Configuração do Projeto Maven

- No arquivo pom.xml, você já incluiu as dependências essenciais, como ojdbc11 para o banco de dados Oracle, Jersey para o servidor REST, e Gson para manipulação de JSON.
- Certifique-se de que o maven-compiler-plugin está configurado para Java 17 (ou a versão que você está usando), e o maven-war-plugin para empacotar a aplicação como um arquivo WAR.

Configuração do Banco de Dados (Oracle)

- Execute os comandos SQL para criar as tabelas necessárias no Oracle Database.
- No código de configuração de banco de dados (credenciais package), adicione as credenciais do banco Oracle.

Configuração do Servidor Tomcat

- Instale e configure o Tomcat, se ainda não o fez.
- **Configuração na IDE**:
- No Eclipse: Servers > Tomcat v9.0 Server at localhost e adicione o projeto..
- Inicie o servidor Tomcat e verifique se a aplicação é carregada com sucesso, acessando no navegador:

http://localhost:8080/sprint4-java/rest

- **Tabela de Endpoints para rodar a aplicação**

Usuário

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/usuario	200 OK	Retornado com sucesso
		400 Bad Request	Erro
			Retorna a lista de todos os usuários
POST	http://localhost:8080/sprint4-java/rest/usuario	201 OK	Criado com sucesso
		400 Bad Request	CPF já existe ou nome já existe
			Corpo da requisição: {"cpfUsuario": "11122233345", "nomeUsuario": "Lucas Kenji", "senha": "lucas099321", "email": "lucas@gmail.com", "telefone": "11982939834"}
PUT	http://localhost:8080/sprint4-java/rest/usuario/{cpfUsuario}	200 OK	Atualizado
		400 Bad Request	CPF não existe ou nome já existe
			Corpo da requisição: {"cpfUsuario": "11122233345", "nomeUsuario": "Lucas Alberto", "senha": "lucas099321", "email": "lucas@gmail.com", "telefone": "11982939834"}
DELETE	http://localhost:8080/sprint4-java/rest/usuario/{cpfUsuario}	200 OK	Deletado
		400 Bad Request	CPF não existe

Veículo

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/veiculo/usuario/{cpf}	200 OK	Retorna os veículos do usuário a partir do CPF
POST	http://localhost:8080/sprint4-java/rest/veiculo	201 OK	Criado com sucesso
		400 Bad Request	Placa já existe
			Corpo da requisição: {"marca": "Chevrolet", "modelo": "Camaro", "placa": "ABT1R23", "ano": 2023, "quilometragem": 39485.0, "usuario": {"cpfUsuario": "11122233345"}}
PUT	http://localhost:8080/sprint4-java/rest/veiculo/{placa}	200 OK	Atualizado
		400 Bad Request	Placa não existe
			Corpo da requisição: {"marca": "Chevrolet", "modelo": "Corvette", "placa": "ABT1R23", "ano": 2023, "quilometragem": 39485.0, "usuario": {"cpfUsuario": "11122233345"}}
DELETE	http://localhost:8080/sprint4-java/rest/veiculo/{placa}	200 OK	Deletado
		400 Bad Request	Placa não existe

Diagnóstico

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/diagnostico/usuario/{cpf}	200 OK	Retorna os diagnósticos a partir do CPF do usuário
		400 Bad Request	Erro
POST	http://localhost:8080/sprint4-java/rest/diagnostico	201 OK	Criado com sucesso
		400 Bad Request	Falha em alguma validação
			Corpo da requisição: {"idDiagnostico": "", "veiculo": {"marca": "Volkswagen", "modelo": "Gol", "placa": "DEF5F78", "ano": 2018, "quilometragem": 30000.0, "usuario": {"cpfUsuario": "23456789012"}}
DELETE	http://localhost:8080/sprint4-java/rest/diagnostico/{idDiagnostico}	200 OK	Deletado com sucesso
		400 Bad Request	Falha em alguma validação

Agendamento

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/agendamento	200 OK	Retorna todos os agendamentos
		400 Bad Request	Erro
GET	http://localhost:8080/sprint4-java/rest/agendamento/usuario/{cpf}	200 OK	Retorna todos os agendamentos do usuário
		400 Bad Request	Erro
POST	http://localhost:8080/sprint4-java/rest/agendamento	201 OK	Criado com sucesso
		400 Bad Request	Falha em alguma validação
			Corpo da requisição: {"idAgendamento": "", "data": "2024-10-30", "hora": "13:30", "descricao": "teste", "centro": {"idCentro": "C010"}, "servico": {"idServico": "S00031"}, "veiculo": {"marca": "Fiat"}}
PUT	http://localhost:8080/sprint4-java/rest/agendamento/{idAgendamento}	200 OK	Atualizado
		400 Bad Request	Falha em alguma validação
			Corpo da requisição: {"idAgendamento": "e399197a-155c-4d90-9f75-964757314a30", "data": "2024-11-12", "hora": "13:40", "descricao": "teste2", "centro": {"idCentro": "C001"}, "servico": {"idServico": "S00031"}, "veiculo": {"marca": "Volkswagen"}}
DELETE	http://localhost:8080/sprint4-java/rest/agendamento/{idAgendamento}	200 OK	Deletado
		400 Bad Request	Id não existe

Centro Automotivo

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/centro	200 OK	Retorna todos os centros automotivos
		400 Bad Request	Erro

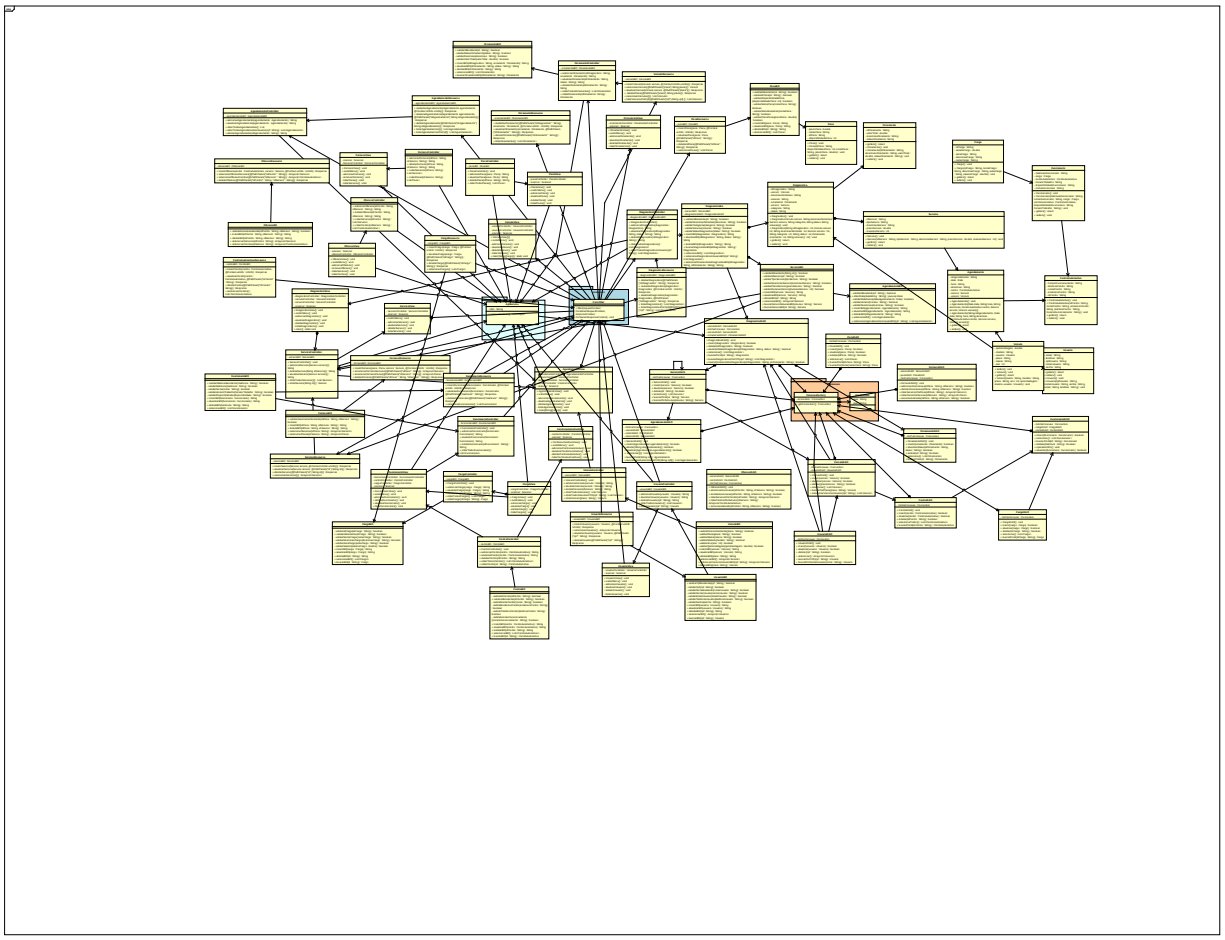
Serviço

Método	Endpoint	Código de Status	Descrição
GET	http://localhost:8080/sprint4-java/rest/servico	200 OK	Retorna todos os serviços
		400 Bad Request	Erro

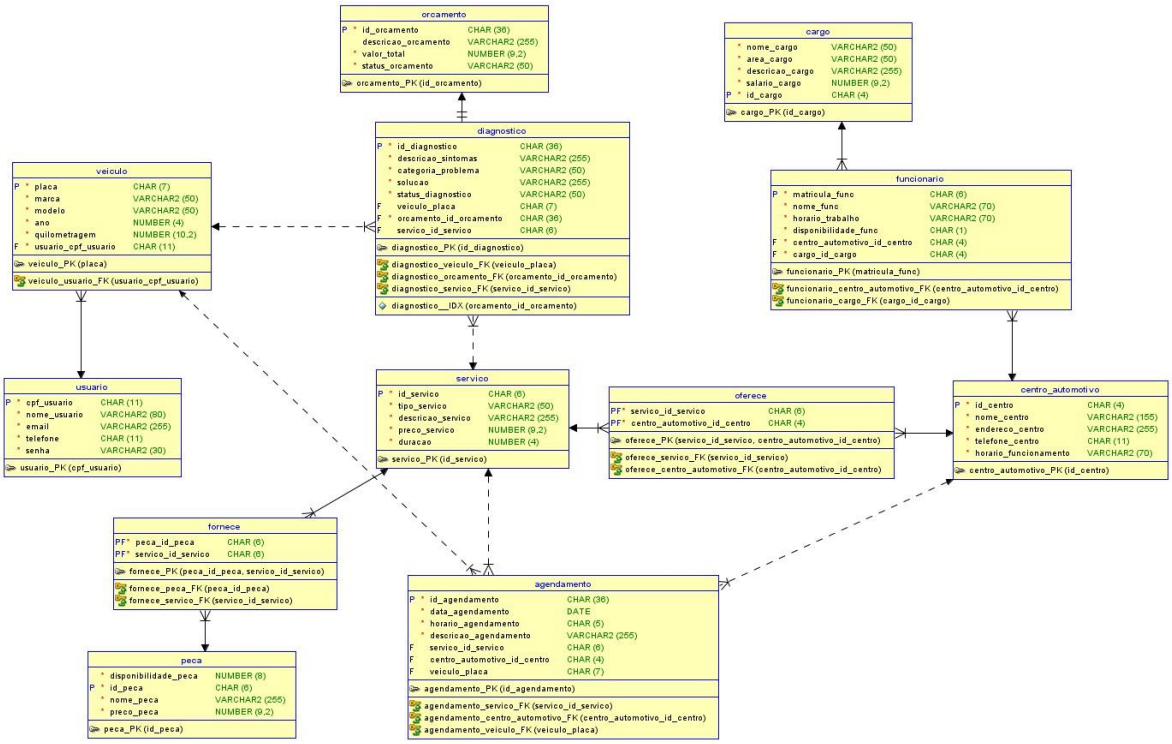
Orçamento

Método	Endpoint	Código de Status	Descrição
POST	http://localhost:8080/sprint4-java/rest/orcamento/{idDiagnostico}	201 OK	Criado com sucesso
		400 Bad Request	Falha em alguma validação
			Corpo da requisição: {"idOrcamento": "", "descricaoOrcamento": "", "valorTotal": null, "statusOrcamento": ""}
DELETE			

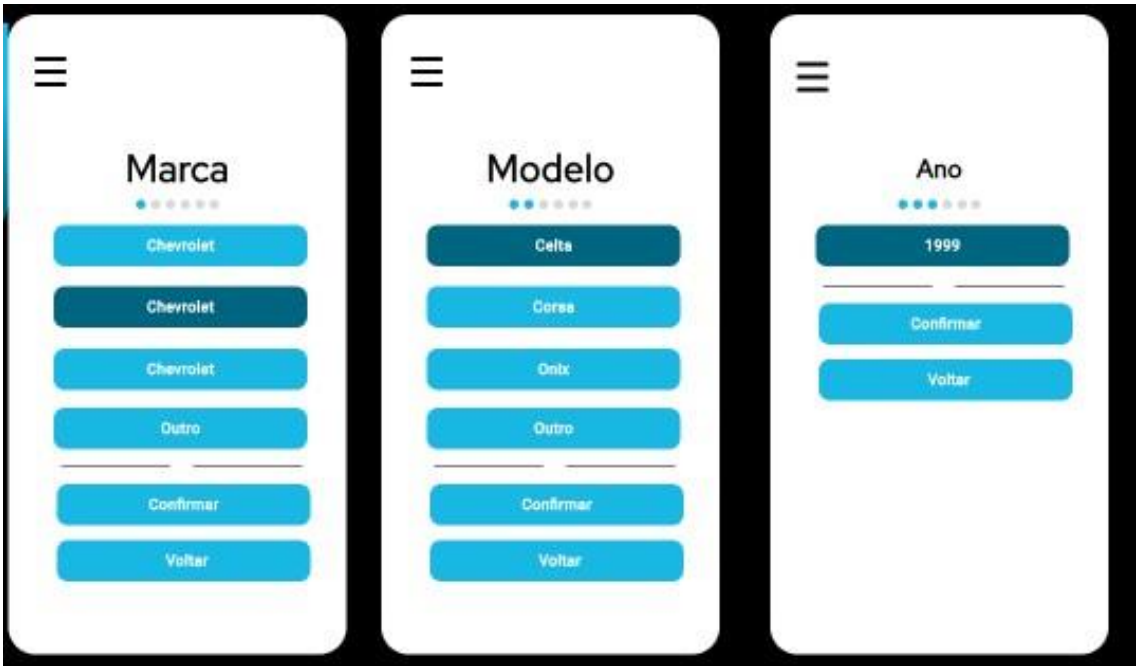
3 – Diagrama de Classes



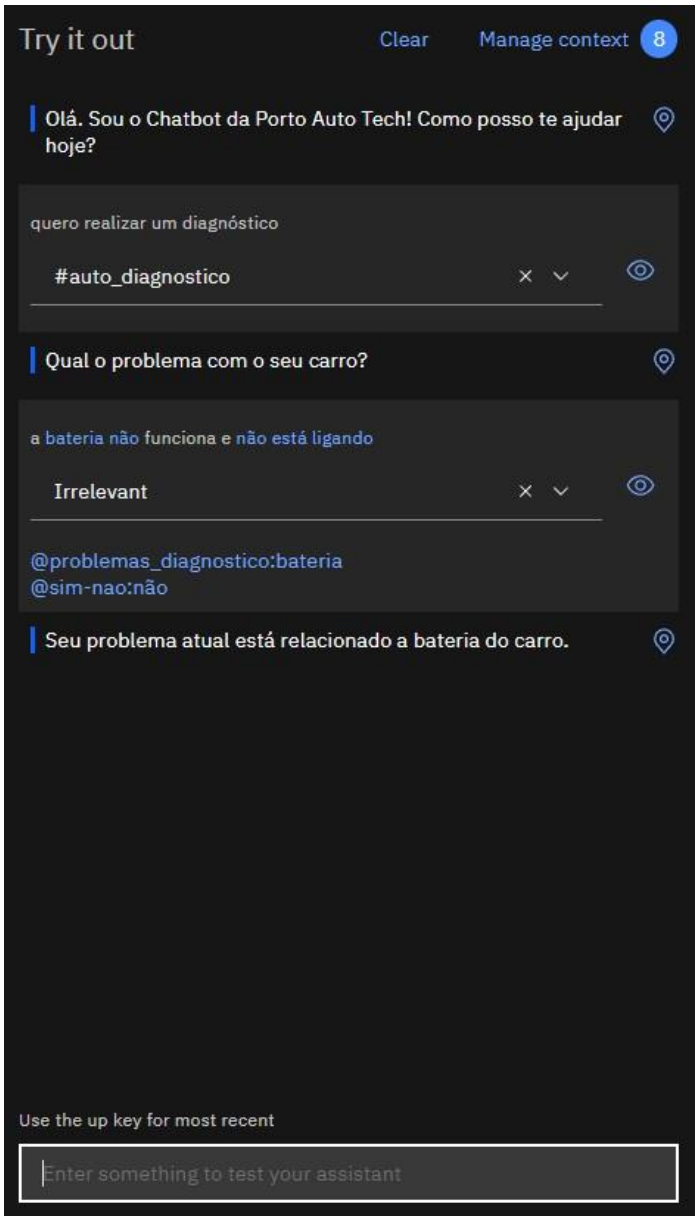
4 – MER



5 – Exemplo de protótipo



O usuário seleciona cada opção de cadastro do carro (marca, modelo, ano, quilometragem e placa), até que possa cadastrá-lo. Com isso, o veículo será inserido no banco de dados.



Através do Chatbot, podemos perguntar ao usuário qual o problema que ele está tendo com o carro. Ao obter o problema identificado, podemos gerar o diagnóstico e inserilo no banco de dados.