

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CAMPUS CORNÉLIO PROCÓPIO  
ENGENHARIA DE SOFTWARE

GABRIEL KENJI INOUE  
LUÍS AUGUSTO CASA GRANDE FONSECA  
PEDRO LUCAS VILA LANDGRAF

**PROJETO CONCESSIONÁRIA**

CORNÉLIO PROCÓPIO  
JUNHO, 2024

Gabriel Kenji Inoue  
Luís Augusto Casa Grande Fonseca  
Pedro Lucas Vila Landgraf

## **PROJETO CONCESSIONÁRIA**

Projeto elaborado na disciplina de Programação Orientada a Objetos 2 do curso de Engenharia da Computação, do Campus Cornélio Procópio da Universidade Tecnológica Federal do Paraná.

Professora: Gisele Alves Santana

CORNÉLIO PROCÓPIO

JUNHO, 2024

## RESUMO

O projeto teve como objetivos criar uma plataforma que atendesse as necessidades entre clientes e concessionárias, como registrar compras e consultas dos mais diversos tipos, e para essa meta foi decidido organizar a estrutura por meio de reuniões e diagramas, após o planejamento, foram utilizados Java jdk-21 com framework Springboot, Java Swing, e a interface Padrão do Java na IDE Netbeans para montar o sistema, o qual posteriormente foi ligado com o banco de dados MySQL por meio da conexão por Springboot. Como resultado do processo, conseguiu-se construir um sistema funcional com um banco de dados flexível e escalável, com uma interface simples de utilizar. Pode-se concluir que apesar de ser um protótipo, o sistema está bem feito e operacional de forma local.

Palavras chave: concessionária, Java, Springboot, Netbeans, MySQL.

# **1 INTRODUÇÃO**

O projeto aborda os processos básicos envolvendo o funcionamento de uma concessionária, a abordagem a esses conceitos é simplificada visto que é um protótipo, por isso não há a presença de servidor ou banco de dados online, apenas local.

Para elaborar o projeto os métodos utilizados foram reuniões informais entre os integrantes com o objetivo de elucidar o cenário, requisitos e processos envolvidos nesse contexto, além da construção de diagramas para facilitar a visualização da estrutura necessária no projeto.

A evolução do assunto abordado foi predominantemente entre a etapa teórica e prática, pois algumas coisas tiveram de ser ajustadas, principalmente a classe agenda. As aplicações no cotidiano são bem diretas, sendo o próprio objetivo do projeto: criar uma plataforma de gerenciamento de informações de uma concessionária, referente aos veículos e agendas de consultas de tipos variados.

# **2 FERRAMENTAS E TECNOLOGIAS**

Desde o começo do projeto, foi decidido a realização do software em Java jdk-21 com o Framework Springboot, que é um dos frameworks mais utilizados e famosos da linguagem Java. Foi então decidido a utilização de um programa desktop, e para isso utilizou-se o Java Swing UI, que é a interface gráfica da IDE NetBeans. Por fim, sobre o banco de dados, utilizou-se o MySQL, devido a sua simplicidade e facilitação de escalar os dados e sua conexão com o driver do Springboot

### 3 DESENVOLVIMENTO

No desenvolvimento do projeto, realizamos a divisão dos códigos em 4 pacotes, sendo Model, View, Control e outro para Exceptions.

```
> com.javagroup.maxconcessionaria.controller
> com.javagroup.maxconcessionaria.controller.exception
> com.javagroup.maxconcessionaria.model
> com.javagroup.maxconcessionaria.view
```

Em geral, a criação do Model ficou igual ao do nosso diagrama de classe, sendo as classes mães, classes abstratas:

```
package com.javagroup.maxconcessionaria.model;

public abstract class User {
    private Integer id;
    private Integer category;
    private String name;
    private String address;
    private String email;
    private String phone;
    private String password;

    public User(Integer id, Integer category, String name, String address, String email, String phone, String password) {
        this.id = id;
        this.category = category;
        this.name = name;
        this.address = address;
        this.email = email;
        this.phone = phone;
        this.password = password;
    }

    public User(Integer category, String name, String address, String email, String phone, String password) {
        this.category = category;
        this.name = name;
        this.address = address;
        this.email = email;
        this.phone = phone;
        this.password = password;
    }
}
```

```
public abstract class Vehicle {
    private String plate;
    private String idChassis;
    private String brand;
    private String model;
    private String color;
    private Integer yearMade;
    private Integer idUser;
    private Boolean available;
    private Boolean sold;

    public Vehicle() {
    }

    public Vehicle(String plate, String idChassis, String brand, String model, String color, Integer yearMade, Integer idUser, Boolean available) {
        this.plate = plate;
        this.idChassis = idChassis;
        this.brand = brand;
        this.model = model;
        this.color = color;
        this.yearMade = yearMade;
        this.idUser = idUser;
        this.available = available;
    }
}
```

```

@Component
public class Car extends Vehicle{
    private Integer quantAirbag;

    public Car(Integer quantAirbag, String plate, String idChassis, String brand, String model, String color, Integer yearMade, Integer idUser, Boolean available) {
        super(plate, idChassis, brand, model, color, yearMade, idUser, available);
        this.quantAirbag = quantAirbag;
    }

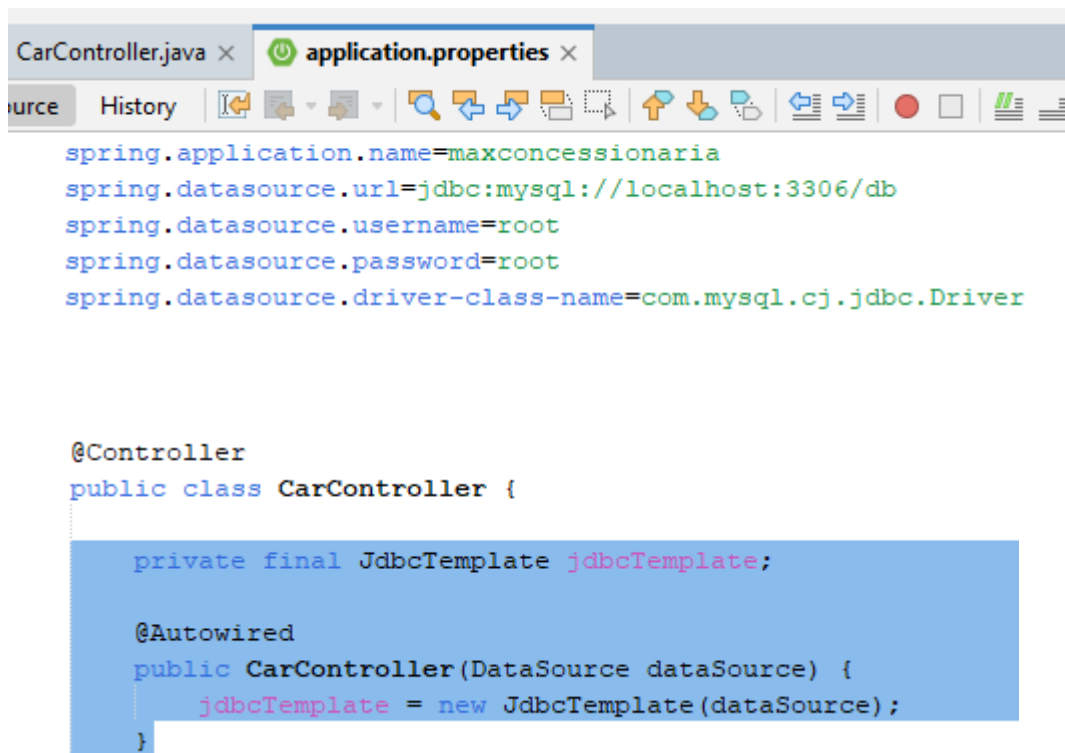
    public Car() {
    }

    public Integer getQuantAirbag() {
        return quantAirbag;
    }

    public void setQuantAirbag(Integer quantAirbag) {
        this.quantAirbag = quantAirbag;
    }
}

```

Já para o funcionamento dos Controllers, foi utilizado a conexão por jdbcTemplate, em que ele realiza a conexão automática pelas informações que inserimos para realizar conexão no application.properties na pasta de resources:



The screenshot shows an IDE with two tabs: 'CarController.java' and 'application.properties'. The 'application.properties' tab is active, displaying the following configuration:

```

spring.application.name=maxconcessionaria
spring.datasource.url=jdbc:mysql://localhost:3306/db
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

```

Below this, the 'CarController.java' file is shown with the following code:

```

@Controller
public class CarController {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public CarController(DataSource dataSource) {
        jdbcTemplate = new JdbcTemplate(dataSource);
    }
}

```

Na foto acima, ao chamarmos a função CarController, passamos como parâmetro o DataSource, que resgata a informação do application.properties e realiza a conexão do SQL. Em geral, as funções dos controllers são bem parecidas, já que são um CRUD:

```

private void validateCar(Car car) {
    if (isEmpty(car.getPlate()) || isEmpty(car.getIdChassis()) || isEmpty(car.getBrand()) || isEmpty(car.getModel()) ||
        car.getYearMade() == null || car.getQuantAirbag() == null || car.getAvailable() == null) {
        throw new NumberFormatException();
    }
}

public void save(Car car) throws IdDuplicatedException, NumberFormatException {
    try{
        String checkSql = "SELECT COUNT(*) FROM Vehicle WHERE plate = ?";
        int count = jdbcTemplate.queryForObject(checkSql, new Object[]{car.getPlate()}, Integer.class);

        if (count > 0) {
            throw new IdDuplicatedException();
        }

        validateCar(car);

        String sql = "INSERT INTO Vehicle VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
        jdbcTemplate.update(sql, car.getPlate(), car.getIdUser(), car.getIdChassis(), car.getBrand(), car.getModel(), car.getYearMade(), car.getColor(), car.getAvailable(), car.getSold());

        String sql2 = "INSERT INTO Car VALUES (?, ?)";
        jdbcTemplate.update(sql2, car.getPlate(), car.getQuantAirbag());

        JOptionPane.showMessageDialog(null, "CARRO CADASTRADO COM SUCESSO!", "CADASTRO DE CARRO", 1);

    } catch (DataAccessException ex) {
        JOptionPane.showMessageDialog(null, "ERRO NA AÇÃO DO BANCO DE DADOS!", "ERRO NO CADASTRO DE CARRO", 0);
    }
}

```

```

public void update(Car car) throws IdDuplicatedException, NumberFormatException {
    try{
        String checkSql = "SELECT COUNT(*) FROM Vehicle WHERE plate = ?";
        int count = jdbcTemplate.queryForObject(checkSql, new Object[]{car.getPlate()}, Integer.class);

        if (count == 0) {
            throw new IdDoesNotExistException();
        }

        String sql = "UPDATE Vehicle SET idChassis = ?, brand = ?, model = ?, color = ?, yearMade = ?, available = ?, sold = ? WHERE plate = ?";
        jdbcTemplate.update(sql, car.getIdChassis(), car.getBrand(), car.getModel(), car.getColor(), car.getYearMade(), car.getAvailable(), car.getSold(), car.getPlate());

        String sql2 = "UPDATE Car SET quantAirbag = ? WHERE idPlate = ?";
        jdbcTemplate.update(sql2, car.getQuantAirbag(), car.getPlate());

        JOptionPane.showMessageDialog(null, "CARRO ATUALIZADO COM SUCESSO!", "ATUALIZAÇÃO DE CARRO", 1);
    }
}

```

```

public void delete(String plate){
    try{
        if(!"".equals(plate) || plate == null){
            throw new NumberFormatException();
        }

        String sql3 = "DELETE FROM Car WHERE idPlate = ?";
        jdbcTemplate.update(sql3, plate);

        String sql4 = "DELETE FROM Vehicle WHERE plate = ?";
        jdbcTemplate.update(sql4, plate);

        JOptionPane.showMessageDialog(null, "CARRO DELETADO COM SUCESSO!", "DELETE DE CARRO", 1);
    }
}

```

```

public void consultAll(DefaultTableModel tableModel) {
    try {
        String sql5 = "SELECT v.plate, v.idChassis, v.brand, v.model, v.color, v.yearMade, c.quantAirbag " +
            "FROM Vehicle v " +
            "INNER JOIN Car c ON (v.plate = c.idPlate)";

        List<Car> listCar = jdbcTemplate.query(sql5, (ResultSet rs, int rowNum) -> {
            Car car = new Car();
            car.setPlate(rs.getString("plate"));
            car.setIdChassis(rs.getString("idChassis"));
            car.setBrand(rs.getString("brand"));
            car.setModel(rs.getString("model"));
            car.setColor(rs.getString("color"));
            car.setYearMade(rs.getInt("yearMade"));
            car.setQuantAirbag(rs.getInt("quantAirbag"));
            return car;
        });
    }
}

```

Ao chamarmos um View, primeiramente chamamos o controller para podermos pegar os valores que estão na interface para as funções:

```

public class ScreenCarCreate extends javax.swing.JFrame {

    private final CarController carController;

    @Autowired
    public ScreenCarCreate(CarController carroController) {
        this.carController = carroController;
        initComponents();
    }
}

```

```

private void ButtonRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Boolean available = boxAvailable.getSelectedItem().equals("Sim");

        if("".equals(fieldAirbag.getText()) || "".equals(fieldYear.getText())){
            JOptionPane.showMessageDialog(null, "TIPO DE DADO CADASTRADO ERRADO!", "ERRO NO CADASTRO DE CARRO", 0);
        }

        Car car = new Car(Integer.valueOf(fieldAirbag.getText()), fieldPlate.getText(), fieldChassis.getText(), fieldBrand.getText(), fieldModel.getText());
        carController.save(car);
    }
    catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(null, "TIPO DE DADO CADASTRADO ERRADO!", "ERRO NO CADASTRO DO CARRO", 0);
    }
}

```

Por fim, para implementarmos a boa prática, como visto acima, temos algumas características, como:

- Escrita está em apenas 1 língua (inglês)
- Nomeação concisa com o que a função realiza e o que é as variáveis
- Nomeação uppercase e lowercase de funções e variáveis
- Indentação do código
- Importação apenas de bibliotecas necessárias

Para o funcionamento do banco de dados, o usuário precisa ter o MySQL instalado na máquina, modificar o application.properties pela informação que está na sua máquina e então rodar o script que está no código de criação de tabelas, por fim, executar/compilar na IDE do Netbeans.

### 3.1 Levantamento dos Requisitos

Os requisitos determinam coisas necessárias para o projeto, e quanto aos levantados, são compostos em Funcionais (determinam funcionalidades do sistema) e não funcionais (Determinam características que envolvem o processo do sistema).

Requisitos Funcionais (RF):



Cadastro de Clientes (RF01): Permitir que os clientes se cadastrem no aplicativo fornecendo informações pessoais básicas como nome, endereço, número de telefone e e-mail.

Catálogo de Veículos (RF02): Mostrar uma lista atualizada de veículos disponíveis na concessionária, incluindo detalhes como marca, modelo, ano, preço e disponibilidade.

Pesquisa Avançada (RF03): Oferecer aos clientes a capacidade de pesquisar veículos com base em critérios específicos, como marca, modelo, ano, faixa de preço e características.

Agendamento de Test Drive (RF04): Permitir que os clientes agendem test drives para os veículos de interesse diretamente pelo aplicativo, escolhendo data e horário disponíveis.

Gerenciamento de Vendas (RF05): Permitir que os funcionários registrem vendas realizadas diretamente pelo aplicativo, incluindo detalhes do cliente, veículo vendido e forma de pagamento.

#### Requisitos Não Funcionais (RNF):

Segurança (RNF01): Garantir a segurança dos dados dos clientes, utilizando medidas como criptografia de dados, autenticação de dois fatores e padrões de segurança de aplicativos móveis.

Desempenho (RNF02): Assegurar que o aplicativo seja responsivo e rápido, mesmo durante períodos de alta demanda de tráfego.

Usabilidade (RNF03): Projetar uma interface de usuário intuitiva e amigável para facilitar a interação dos usuários com o aplicativo.

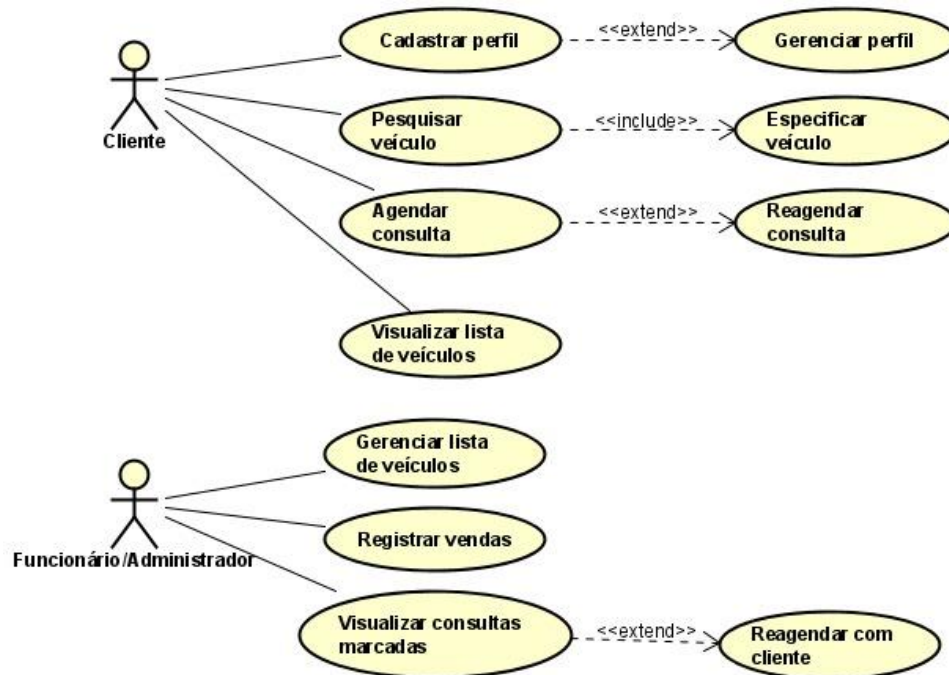
Confiabilidade (RNF04): Garantir que o aplicativo seja robusto e confiável, minimizando falhas e tempo de inatividade.

Acessibilidade (RNF05): Garantir que o aplicativo seja acessível para pessoas com deficiências, seguindo diretrizes de acessibilidade móvel e oferecendo opções de acessibilidade, como aumento de tamanho de fonte e suporte a leitores de tela.

### 3.2 Diagramas de Casos de Uso

Os casos de uso demonstram as funções que os usuários exercem no sistema.

De forma resumida, o sistema é utilizado pelo cliente, o qual tem ações ligadas ao seu perfil, a pesquisa de veículos e utilizar sua agenda. Outro usuário é o funcionário/administrador, o qual pode gerenciar os veículos, as vendas e a agenda da concessionária.



O diagrama de Casos de Uso não passou por alterações pois por não ser específico não houveram conflitos com a parte prática do projeto.

### 3.3 Especificação dos Casos de Uso

Na especificação de casos de uso há 4 exemplos para se visualizar, todos possuindo caminhos alternativos em casos específicos que impossibilitem o andamento do caso:

ID do caso de uso:	1	ID do caso de uso:	2
ID do cenário:	1	ID do cenário:	2
Nome do cenário:	Cadastrar perfil	Nome do cenário:	Pesquisar veículo
Sequência de eventos:	<p>1) O usuário clica em “Criar perfil”.</p> <p>2) O sistema disponibiliza campos para o usuário preencher, como nome, endereço, telefone e e-mail.</p> <p>3) O usuário insere os dados e clica em criar conta.</p> <p>4) O sistema verifica se o login já existe.</p> <p>5) O sistema valida e salva os dados do usuário.</p> <p>6) O sistema gera um ID para o usuário.</p> <p>7) O sistema emite uma mensagem de sucesso.</p> <p>ALTERNATIVO:</p> <p>4.1) Caso o usuário insira dados que já existam em outro perfil, o sistema notifica a ocorrência e indica quais campos devem ser alterados, impossibilitando a criação do perfil.</p> <p>5.1) Caso o sistema não consiga validar e/ou salvar os dados do usuário, o sistema notifica o usuário do problema no serviço e não permite a criação do perfil.</p>	Sequência de eventos:	<p>1) O usuário clica na ferramenta de busca.</p> <p>2) O sistema apresenta as opções de filtros de pesquisa e permite o usuário digitar o que deseja.</p> <p>3) O usuário insere as informações necessárias sobre o veículo, como nome ou características do mesmo.</p> <p>4) O sistema recebe as informações e filtra a lista de veículos de acordo com as especificações.</p> <p>5) O sistema apresenta os veículos de acordo com os filtros em uma lista.</p> <p>ALTERNATIVO:</p> <p>3.1) Caso o nome do veículo não seja compatível com algum da lista, o sistema notifica ao usuário que o veículo não está na lista.</p> <p>5.1) Caso o sistema não encontre algum veículo com a filtragem atual, o mesmo exibe uma mensagem de erro e indica o usuário alterar a filtragem.</p>

A primeira é sobre o cadastro de perfil, no qual o usuário segue um caminho de inserção de informações para criar seu cadastro no sistema. Os caminhos alternativos são para o caso de inserção incorreta de id ou de o sistema não consiga salvar as informações.

A segunda é sobre a pesquisa de veículo, no qual o usuário segue um caminho de inserção de informações chave que serão utilizadas para filtrar a busca de veículos e posteriormente apresentar uma lista para o usuário. Os caminhos alternativos são para o caso de o nome do veículo não existir na lista ou de não encontrar algum veículo utilizando a filtragem atual.

ID do caso de uso:	3
ID do cenário:	3
Nome do cenário:	Agendar consulta
Sequência de eventos:	<p>1) O usuário clica em "Consultas".</p> <p>2) O sistema apresenta a interface de consultas.</p> <p>3) O usuário clica em "Agendar nova consulta".</p> <p>4) O sistema abre as opções de escolha tanto dos horários disponíveis quanto do assunto da consulta.</p> <p>5) O usuário escolhe ambos detalhes sobre a consulta e clica em confirmar.</p> <p>6) O <u>sistema salva</u> a consulta na lista de horários da empresa e emite uma mensagem de sucesso.</p> <p>ALTERNATIVO:</p> <p>3.1) Caso o sistema de consultas não esteja disponível no momento, o sistema notifica ao usuário o problema com o serviço.</p> <p>4.1) Caso a consulta seja sobre algo que não esteja nas opções de predefinição do sistema, o usuário poderá clicar na opção outro e digitar qual é o motivo da consulta.</p>

ID do caso de uso:	4
ID do cenário:	3
Nome do cenário:	Reagendar com cliente
Sequência de eventos:	<p>1) O funcionário clica em "Consultas".</p> <p>2) O sistema apresenta a interface com as consultas dos clientes.</p> <p>3) O funcionário escolhe a consulta desejada.</p> <p>4) O sistema abre uma interface com as informações sobre a consulta.</p> <p>5) O funcionário clica em "Remarcar".</p> <p>6) O sistema disponibiliza o telefone do usuário e uma interface com os horários disponíveis.</p> <p>7) O funcionário liga para o cliente e marca com ele um novo horário.</p> <p>8) O funcionário preenche o novo horário e clica em "Marcar".</p> <p>9) O sistema emite uma mensagem de sucesso.</p> <p>ALTERNATIVO:</p> <p>2.1) Caso o sistema esteja fora do ar ou não possa acessar as consultas, o sistema informa o funcionário sobre o problema no serviço e cancela a abertura da interface de consultas.</p> <p>7.1) Caso o cliente queira cancelar, ou o funcionário não consiga contatar o cliente, o sistema envia um e-mail para o cliente informando o ocorrido e dá a opção para o funcionário de cancelar a consulta.</p>

A terceira é sobre agendar uma consulta, no qual o usuário segue um caminho de escolhas para determinar uma data para uma consulta e o tema abordado na consulta. Os caminhos alternativos são para o caso de o sistema não estar disponível ou de o usuário querer identificar um tipo não predefinido de consulta.

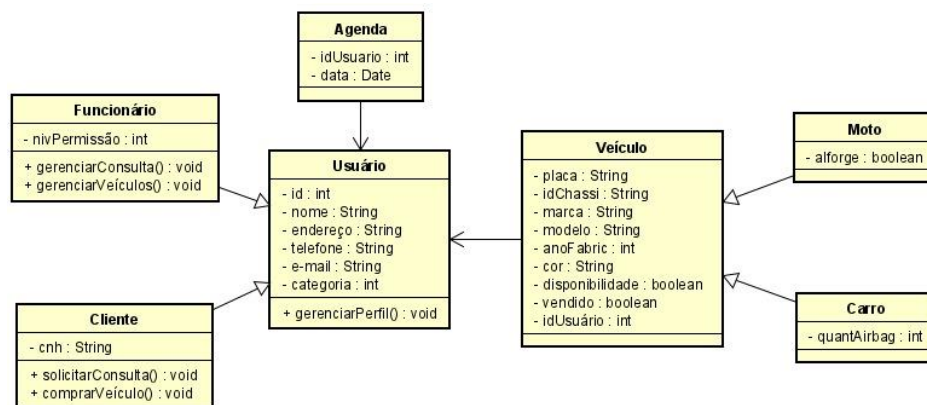
A quarta é sobre reagendar uma consulta com o cliente, no qual o funcionário segue um caminho de escolhas e tenta contatar o cliente para trocar a data da sua consulta. Os caminhos são para o caso de o sistema não poder ser acessado ou de o cliente querer cancelar devido a troca de data ou caso não haja retorno do cliente.

### 3.4 Diagrama de Classes

O diagrama de classes demonstra como é a estrutura de informações e funções do sistema.

O diagrama possui as classes mães Usuário (que é a generalização de Funcionário e Cliente) e a Veículo (generalização de Moto e Carro) além da classe Agenda (a qual podem ser várias para apenas um usuário).

Pode-se perceber que as funções estão concentradas nos usuários, visto que eles praticam as ações, enquanto as outras classes sofrem as ações.

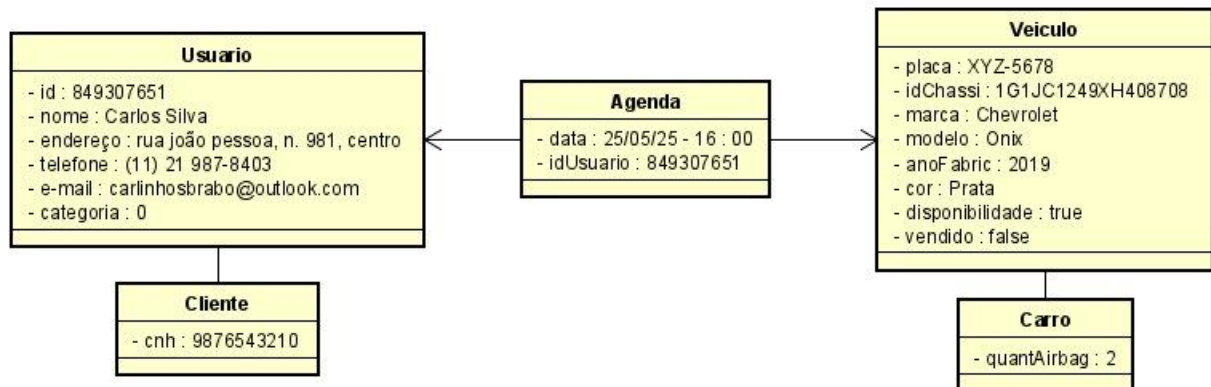


O diagrama de Classes acabou passando por algumas mudanças devido a alteração da percepção da equipe durante o desenvolvimento do projeto.

### 3.5 Diagrama de Objetos

O diagrama de objetos é uma demonstração prática das informações organizadas na estrutura.

Nesse caso foram criadas informações de exemplo (sample) que permitem visualizar o tipo da informação de forma prática, a escolha foi utilizar cliente e carro nesse caso.



### 3.6 Banco de Dados

```

CREATE TABLE IF NOT EXISTS User(
    id INT AUTO_INCREMENT PRIMARY KEY,
    category INT NOT NULL,
    name VARCHAR(50) UNIQUE NOT NULL,
    address VARCHAR(50),
    email VARCHAR(70),
    phone VARCHAR(50),
    password VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS Customer(
    idUser INT PRIMARY KEY,
    cnh VARCHAR(50) NOT NULL,
    FOREIGN KEY (idUser) REFERENCES User(id)
);

CREATE TABLE IF NOT EXISTS Employee(
    idUser INT PRIMARY KEY,
    lvlPermission INT NOT NULL,
    FOREIGN KEY (idUser) REFERENCES User(id)
);

```

```

CREATE TABLE IF NOT EXISTS Schedule(
    idUser INT,
    date DATE,
    FOREIGN KEY (idUser) REFERENCES User(id)
);

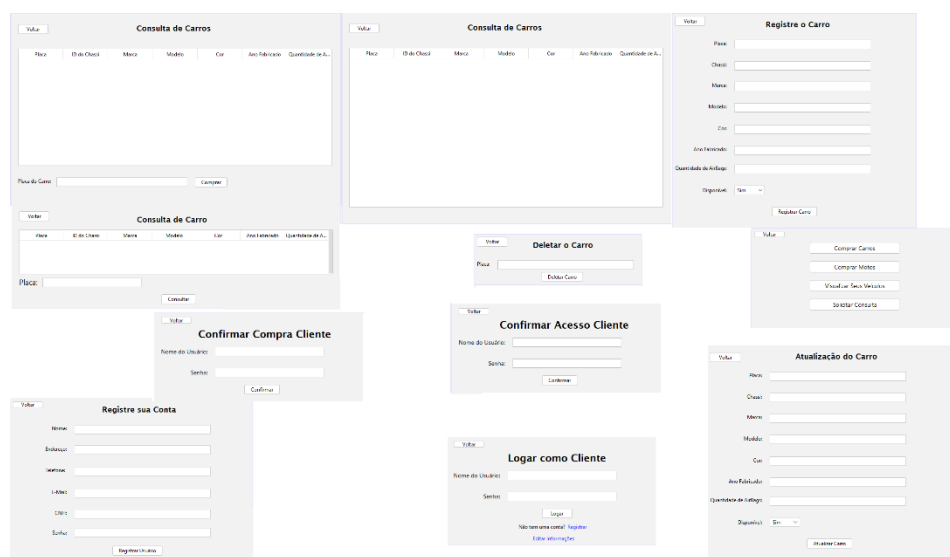
CREATE TABLE IF NOT EXISTS Vehicle(
    plate VARCHAR(50) PRIMARY KEY,
    idUser INT,
    idChassis VARCHAR(50) NOT NULL,
    brand VARCHAR(50),
    model VARCHAR(50),
    yearMade INT,
    color VARCHAR(50),
    available BOOLEAN,
    sold BOOLEAN,
    FOREIGN KEY (idUser) REFERENCES User(id)
);

CREATE TABLE IF NOT EXISTS Motorcycle(
    idPlate VARCHAR(50) PRIMARY KEY,
    saddlebag BOOLEAN,
    FOREIGN KEY (idPlate) REFERENCES Vehicle(plate)
);

CREATE TABLE IF NOT EXISTS Car(
    idPlate VARCHAR(50) PRIMARY KEY,
    quantAirbag INT,
    FOREIGN KEY (idPlate) REFERENCES Vehicle(plate)
);

```

### 3.7 Telas do sistema



Voltar

Consulta de Agenda

Nome do Usuário

Data

Voltar

Registre a Agenda

Data

Nome do Usuário

Registrar Agenda

Voltar

Deletar a Agenda

Data

Nome do Usuário

Deletar Agenda

Voltar

Registrar Agenda

Deletar Agenda

Consultar Agenda

Voltar

Atualização do Cliente

Nome

Senha

Informações para Modar:

Senha

Endereço

E-mail

Telefone

Atualizar informações

Voltar

Consulta de Veículos

Placa

ID do Item

Marca

Cor

Modelo

Ano Fabricado

Voltar

Registrar Carro

Atualizar Carro

Deletar Carro

Consultar Carro

Consultar Todos os Carros

Voltar

Registre sua Conta

Nome

Endereço

Telefone

E-Mail

Senha

Permissão: Funcionário

Registrar Usuário

Voltar

Logar como Funcionário

Nome do Usuário

Senha

Login

Não tem uma conta? [Registrar](#)

[Editar Informações](#)

Voltar

Consultar Carro

Consultar Moto

Consultar Agenda

Voltar

Atualização do Funcionário

Nome

Senha

Informações para Modar:

Senha

Endereço

E-mail

Telefone

Atualizar informações

Voltar

Registrar Moto

Atualizar Moto

Deletar Moto

Consultar Moto

Consultar Todos os Motos

Voltar

Registrar Agenda

Deletar Agenda

Consultar Agenda

Voltar

Consulta de Moto

Placa

ID do Item

Marca

Modelo

Ano Fabricado

Ano Modelo

Placa:

Consultar

Voltar

Registre a Moto

Placa

Marca

Modelo

Ano Fabricado

Ano Modelo

Verificação

Senha

Registrar Moto

Voltar

Consulta de Moto

Placa

ID do Item

Marca

Modelo

Cor

Ano Fabricado

Ano Modelo

Placa/Modelo:

Consultar

Voltar

Consulta de Motos

Placa

ID do Item

Marca

Modelo

Cor

Ano Fabricado

Ano Modelo

Voltar

Deletar Moto

Placa

Deletar Moto

Voltar

Atualização de Moto

Placa

Marca

Modelo

Ano Fabricado

Cor

Ano Modelo

Verificação

Senha

Atualizar Moto

Voltar

Registre a Agenda

Data

Nome do Usuário

Registrar Agenda

Voltar

Consulta de Agenda

Nome do Usuário

Data

Voltar

Deletar a Agenda

Data

Nome do Usuário

Deletar Agenda

Selecionar o Tipo de Usuário

Cliente

Funcionário



## 4 CONCLUSÕES

Em comparação entre os resultados esperados e obtidos, o projeto foi um sucesso, pois ele proporciona um sistema que possibilita gerenciamento de veículos e agenda de uma concessionária, para uso de cliente e funcionário, que é local por natureza de ser um protótipo.

As dificuldades foram as adaptações necessárias entre a parte teórica e prática devido a mudança de perspectiva sobre alguns pontos do cenário de uso do projeto, mas são mudanças pontuais e simples. Quanto as melhorias, recomenda-se trocar o Java Swing por outra opção como o React, devido a suas limitações e necessidades de importações.

O projeto contribuiu para o aprendizado da equipe no sentido de gerenciar mudanças da documentação por causa da implementação e como implementar a