



Faculdade de Informática e Administração Paulista

# **Domain Driving Using Java**

## **Sprint 4**

# INTEGRANTES

<b>RM (Somente Numeros)</b>	<b>Nome Completo (Sem abreviar)</b>
554874	João Gabriel Boaventura Marques e Silva
558791	Lucas de Melo Pinheiro Pinho
551124	Lucas Leal das Chagas

## Sumário

1. DESCRIÇÃO DO PROJETO.....	4
2. INTERAGINDO COM O SISTEMA.....	7
3. MODELO DO BANCO DE DADOS.....	11
4. DIAGRAMA DE CLASSES.....	16

## **1 – DESCRIÇÃO DO PROJETO**

O código funcionará para manter informações importantes do cliente para que possibilite o contato, e principalmente para que o assistente consiga ter acesso às informações tanto de automóveis quanto do plano, para assim trazer agilidade e a ajuda necessária para o devido problema do usuário. Para os planos, o código fornecerá informações sobre os diferentes tipos de carros que atendem, dependendo do estilo de vida, sobre os serviços oferecidos. O código funcionará para fornecer informações sobre as Oficinas disponíveis para o conserto de seu veículo a partir de seu plano, lhe informando sobre Avaliações, Endereço e Serviços. E também de Lojas Parceiras que lhe fornecera as peças de que necessita para a reparação de seu automóvel, disponibilizando Peças e diferentes serviços.

O Principal problema que o programa tem como função ajudar, é trazer soluções rápidas para aquelas pessoas que não possuem e nem querem possuir conhecimento mecânico, trazendo facilidade e praticabilidade para a vida do cliente.

- VO: Contém as classes que representam as entidades principais do sistema com seus atributos.

BO: Contém as classes que implementam a lógica de negócios. Elas são responsáveis por validar regras e aplicar lógica antes de interagir com o DAO.

DAO: Contém as classes que realizam operações de CRUD no banco de dados. Elas interagem diretamente com a base de dados usando ConnDAO.

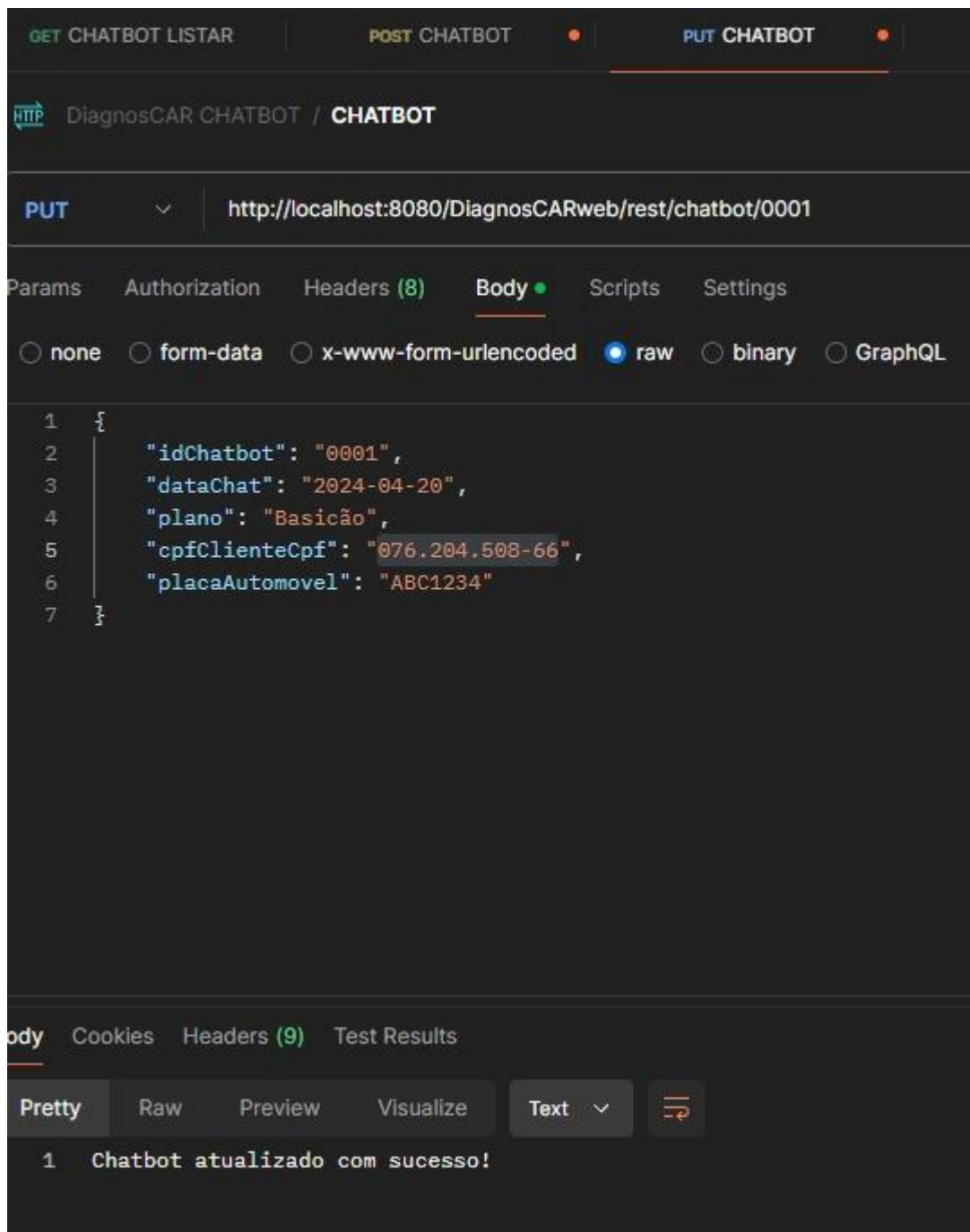
Resources: Contém arquivos de configuração e outros recursos necessários para o funcionamento do sistema

Connection: **ConnDAO**: Classe que gerencia a conexão com o banco de dados.

## Breve Descrição das Principais Funcionalidades do Projeto

- **Gerenciamento de Clientes:** Cadastro, edição, exclusão e busca de clientes, armazenando informações como CPF, CNH, e dados de contato.
- **Gerenciamento de Automóveis:** Vinculação de veículos aos clientes, incluindo informações como placa, marca, modelo e ano.
- **Gerenciamento de Lojas Parceiras e Oficinas:** Cadastro e listagem de lojas e oficinas, com informações de CNPJ, nome e especialização.
- **Gerenciamento de Peças:** Registro de peças disponíveis nas lojas parceiras, permitindo consulta e inserção de peças no sistema.

## 2- Interagindo com o sistema



GET CHATBOT LISTAR

POST CHATBOT

PUT CHATBOT

DEL CHATBOT

+

DiagnosCAR CHATBOT / CHATBOT LISTAR

GET

▼

http://localhost:8080/DiagnosCARweb/rest/chatbot

ParamsAuthorizationHeaders (6)BodyScriptsSettings

Query Params

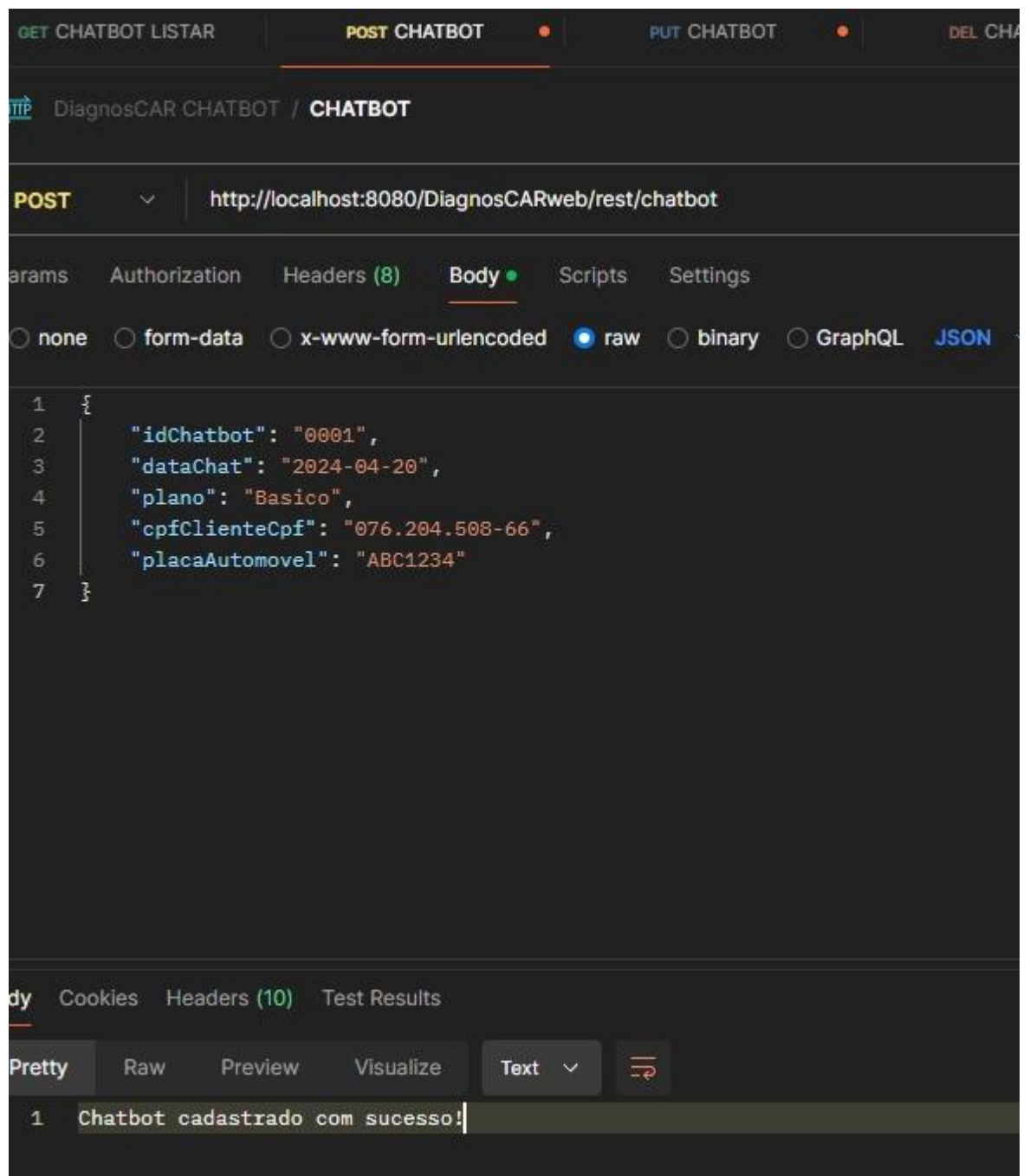
	Key	Value
	Key	Value

BodyCookiesHeaders (9)Test Results

PrettyRawPreviewVisualizeJSON▼

```
1  [
2    {
3      "idChatbot": "0001",
4      "dataChat": "2024-04-19",
5      "plano": "Basico",
6      "cpfClienteCpf": "076.204.508-66",
7      "placaAutomovel": "ABC1234"
8    }
9  ]
```





GET CHATBOT LISTAR

POST CHATBOT

PUT CHATBOT

DEL CHATBOT

DiagnosCAR CHATBOT / CHATBOT

DELETE

http://localhost:8080/DiagnosCARweb/rest/chatbot/0001/076.204.508-66

Params

Authorization

Headers (6)

Body

Scripts

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (9)

Test Results

Pretty

Raw

Preview

Visualize

Text

1 Chatbot deletado com sucesso!

### 3- Modelo do Banco de Dados

CREATE TABLE Cliente (

CPF\_Cliente VARCHAR2(14) PRIMARY KEY NOT NULL,

CNH\_Cliente VARCHAR2(11) UNIQUE NOT NULL,

Nome\_Cliente VARCHAR2(100) NOT NULL,

Sobrenome\_Cliente VARCHAR2(100) NOT NULL,

Email\_Cliente VARCHAR2(100) UNIQUE,

Telefone\_Cliente VARCHAR2(15) UNIQUE,

Endereco\_Cliente VARCHAR2(200) NOT NULL,

CONSTRAINT CHK\_Cliente\_CPF CHECK ( REGEXP\_LIKE(CPF\_Cliente, '^[0-9]{3}\.[0-9]{3}\.[0-9]{3}-[0-9]{2}\$')),

CONSTRAINT CHK\_Cliente\_Telefone CHECK (REGEXP\_LIKE(Telefone\_Cliente, '^\(\d{2}\) \d{4,5}-\d{4}\$')),

CONSTRAINT CHK\_Cliente\_Email CHECK ( REGEXP\_LIKE(Email\_Cliente, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\$'))

);

CREATE TABLE Automovel (

Placa\_Automovel VARCHAR2(7) PRIMARY KEY NOT NULL,

Marca\_Automovel VARCHAR2(20) NOT NULL,

Modelo\_Automovel VARCHAR2(100) NOT NULL,

Cor\_Automovel VARCHAR2(15) NOT NULL,

Ano\_Automovel NUMBER(4) NOT NULL,

Cliente\_CPF\_Cliente VARCHAR2(14),

CONSTRAINT FK\_Automovel\_Cliente FOREIGN KEY ( Cliente\_CPF\_Cliente )  
REFERENCES Cliente(CPF\_Cliente)  
);

```

CREATE TABLE Chatbot(
  ID_Chatbot VARCHAR2(1000) NOT NULL,

  Horario_Chat DATE NOT NULL,

  Plano VARCHAR2(10) NOT NULL,

  Tipo_Chat VARCHAR2(1) NOT NULL,

  Cliente_CPF_Cliente VARCHAR2(14),

  CONSTRAINT PK_Chatbot PRIMARY KEY (ID_Chatbot, Cliente_CPF_Cliente),

  CONSTRAINT FK_Chatbot_Cliente FOREIGN KEY ( Cliente_CPF_Cliente )
  REFERENCES Cliente(CPF_Cliente)

);

```

```

CREATE TABLE Pre_Diagnostico (

  ID_PreDiagnostico VARCHAR2(1000) PRIMARY KEY NOT NULL,

  Nivel_Diagnostico NUMBER(3) CHECK (Nivel_Diagnostico >= 1 AND
  Nivel_Diagnostico <= 100) ,

  Diagnostico VARCHAR2(500) NOT NULL,

  Assistente_ID_Chatbot VARCHAR2(1000) NOT NULL,

  Cliente_CPF_Cliente VARCHAR2(14) NOT NULL,

  CONSTRAINT FK_PreDiagnostico_Chatbot FOREIGN KEY
  (Assistente_ID_Chatbot, Cliente_CPF_Cliente)

  REFERENCES Chatbot(ID_Chatbot, Cliente_CPF_Cliente)

);

```

```

CREATE TABLE Loja_Parceira (

    Endereco_Loja VARCHAR2(200) PRIMARY KEY NOT NULL,
    Cnpj_Loja VARCHAR2(18) NOT NULL UNIQUE,

    Nome_Loja VARCHAR2(120) NOT NULL,

    Avaliacao_Loja NUMBER(3, 2) ,

    Especializacao_Loja VARCHAR2(50) NOT NULL,
    CONSTRAINT CHK_Loja_Cnpj CHECK ( REGEXP_LIKE(Cnpj_Loja,
'^[0-9]{2}\.[0-9]{3}\.[0-9]{3}/[0-9]{4}-[0-9]{2}$')),

    CONSTRAINT CHK_Loja_Avaliacao CHECK (Avaliacao_Loja BETWEEN 0 AND
10)

);

```

```

CREATE TABLE Oficina (

    Endereco_Oficina VARCHAR2(200) PRIMARY KEY NOT NULL,

    Cnpj_Oficina VARCHAR2(18) NOT NULL UNIQUE,

    Nome_Oficina VARCHAR2(120) NOT NULL,

    Avaliacao_Oficina NUMBER(3, 2) ,

    Especializacao_Oficina VARCHAR2(50) NOT NULL,

    Chatbot_ID_Chatbot VARCHAR2(1000),

    Chatbot_Cliente_CPF_Cliente VARCHAR2(14),

    CONSTRAINT FK_Oficina_Chatbot FOREIGN KEY ( Chatbot_ID_Chatbot,
Chatbot_Cliente_CPF_Cliente) REFERENCES Chatbot(ID_Chatbot,
Cliente_CPF_Cliente),

    CONSTRAINT CHK_Oficina_Cnpj CHECK ( REGEXP_LIKE(Cnpj_Oficina,
'^[0-9]{2}\.[0-9]{3}\.[0-9]{3}/[0-9]{4}-[0-9]{2}$')),

    CONSTRAINT CHK_Oficina_Avaliacao CHECK (Avaliacao_Oficina BETWEEN 0
AND 10)

);

```

```

CREATE TABLE Peca (
    ID_Peca NUMBER PRIMARY KEY NOT NULL,

    Tipo_Peca VARCHAR2(30) NOT NULL UNIQUE,

    Nome_Peca VARCHAR2(40) NOT NULL,

    Endereco_Peca VARCHAR2(200) NOT NULL,

    Loja_Parceira_Endereco_Loja VARCHAR2(200),
    CONSTRAINT FK_Peca_Loja FOREIGN KEY ( Loja_Parceira_Endereco_Loja )
    REFERENCES Loja_Parceira(Endereco_Loja)

);

```

```

CREATE TABLE Entrega (

    ID_Entrega VARCHAR2(100) PRIMARY KEY NOT NULL,

    Data_Entrega DATE NOT NULL,

    Destino_Entrega VARCHAR2(200) NOT NULL,

    Item_Entrega VARCHAR2(100) NOT NULL,

    Endereco_Loja VARCHAR2(200) NOT NULL,

    CONSTRAINT FK_Entrega_Loja FOREIGN KEY (Endereco_Loja) REFERENCES
    Loja_Parceira(Endereco_Loja)

);

```

CREATE TABLE Vincula (

CPF\_Cliente VARCHAR2(14) NOT NULL,

ID\_Chatbot VARCHAR2(1000) NOT NULL,

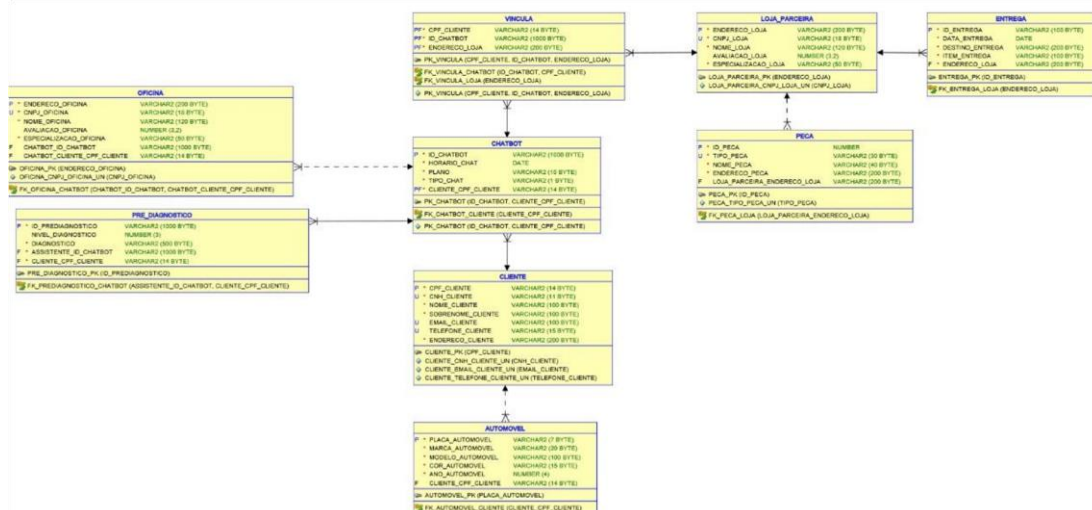
Endereco\_Loja VARCHAR2(200),

CONSTRAINT PK\_Vincula PRIMARY KEY (CPF\_Cliente, ID\_Chatbot,  
Endereco\_Loja),

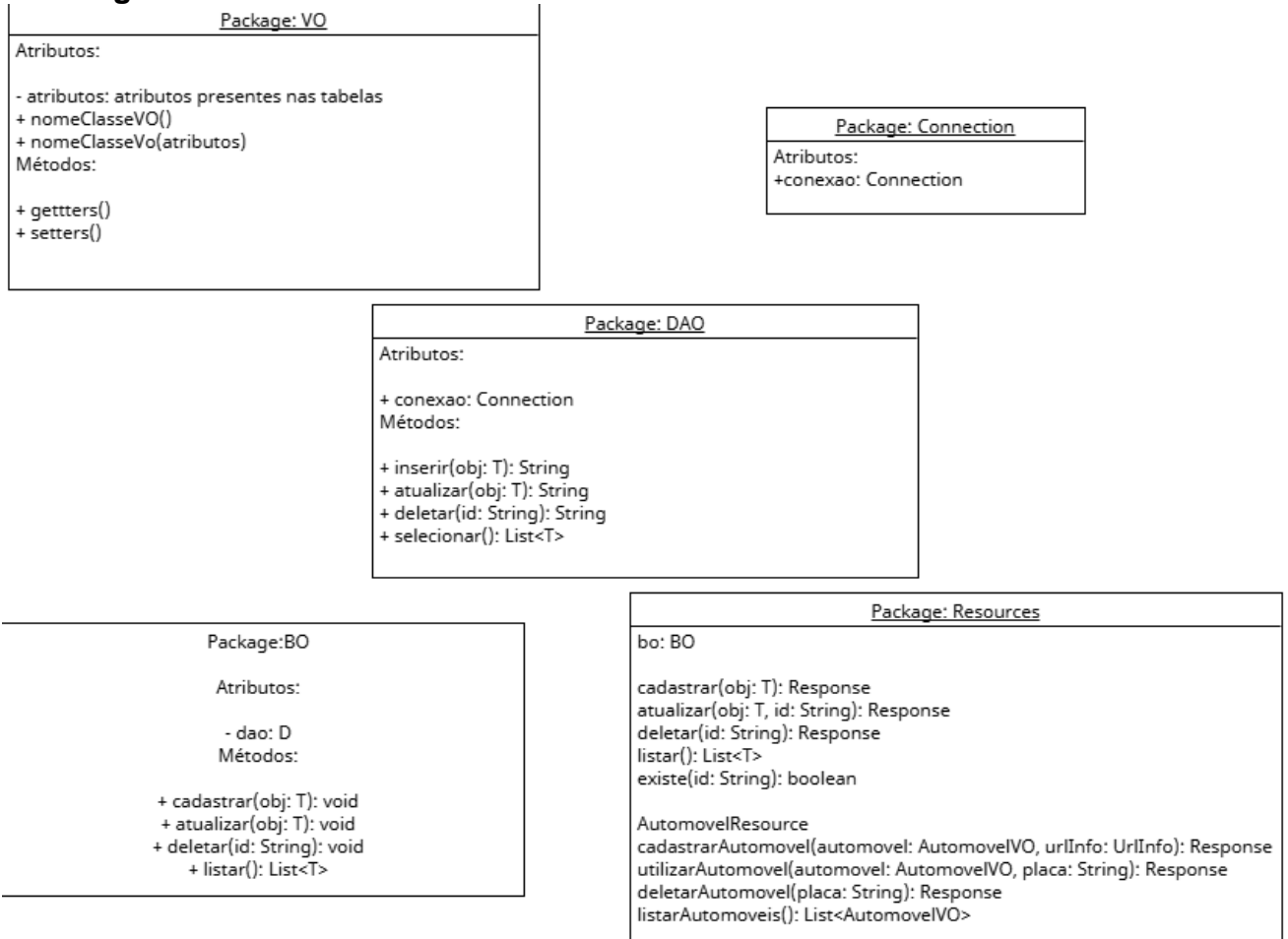
CONSTRAINT FK\_Vincula\_Chatbot FOREIGN KEY (CPF\_Cliente, ID\_Chatbot)  
REFERENCES Chatbot(Cliente\_CPF\_Cliente, ID\_Chatbot),

CONSTRAINT FK\_Vincula\_Loja FOREIGN KEY (Endereco\_Loja) REFERENCES  
Loja\_Parceira(Endereco\_Loja)

);



## 4 – Diagrama de classes



**\*Feito no UMLetino**







