PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Arthur Victor L. de M. Alves[1]
Brunno Enrico Machado Costa[2]
Diogo Meireles Ribeiro[3]
Vitória Rani Santos Menezes[4]

Documentação Sistema Logística

Modelo de Dados para Controle de Entregas Domiciliares

Descrição Geral

O banco de dados será responsável por gerenciar toda a operação de entregas, desde o cadastro de veículos e motoristas até a otimização de rotas e geração de relatórios. O sistema prioriza a eficiência na alocação de entregas, o controle de status em tempo real e a análise de desempenho.

Entidades Principais e Funcionalidades

1. Frota de Veículos (Tabela *veiculo*) :

- Armazena informações como tipo (moto, caminhão, van), placa, capacidade máxima (peso e volume), status (disponível, em rota, em manutenção), e histórico de manutenções.
- Permite otimizar a alocação de entregas com base na capacidade de carga do veículo.

2. Motoristas (Tabela *motorista*):

- Registra dados do motorista (codigo, nome, CNH, contato) e associa cada motorista a um veículo.
- Inclui disponibilidade e histórico de rotas.

3. Clientes e Destinatários (Tabela *Cliente*):

- Armazena informações de destinatários (codigo, nome, endereço, geolocalização, telefone).
- Facilita a busca por entregas próximas para otimização de rotas.

4. Entregas (Tabela *entrega*):

- Campos: ID, cliente_id, data_prevista, data_conclusao, status (pendente, em trânsito, entregue, atrasado), prioridade (urgente, normal).
- Calcula atrasos comparando `data_prevista` e `data_conclusao`.

5. Itens das Entregas (Tabela *Item*):

- Detalhes da mercadoria: codigo, peso, dimensões (altura, largura, profundidade), categoria (frágil, perecível), valor declarado.
- Auxilia na alocação de veículos com base no volume total dos itens.

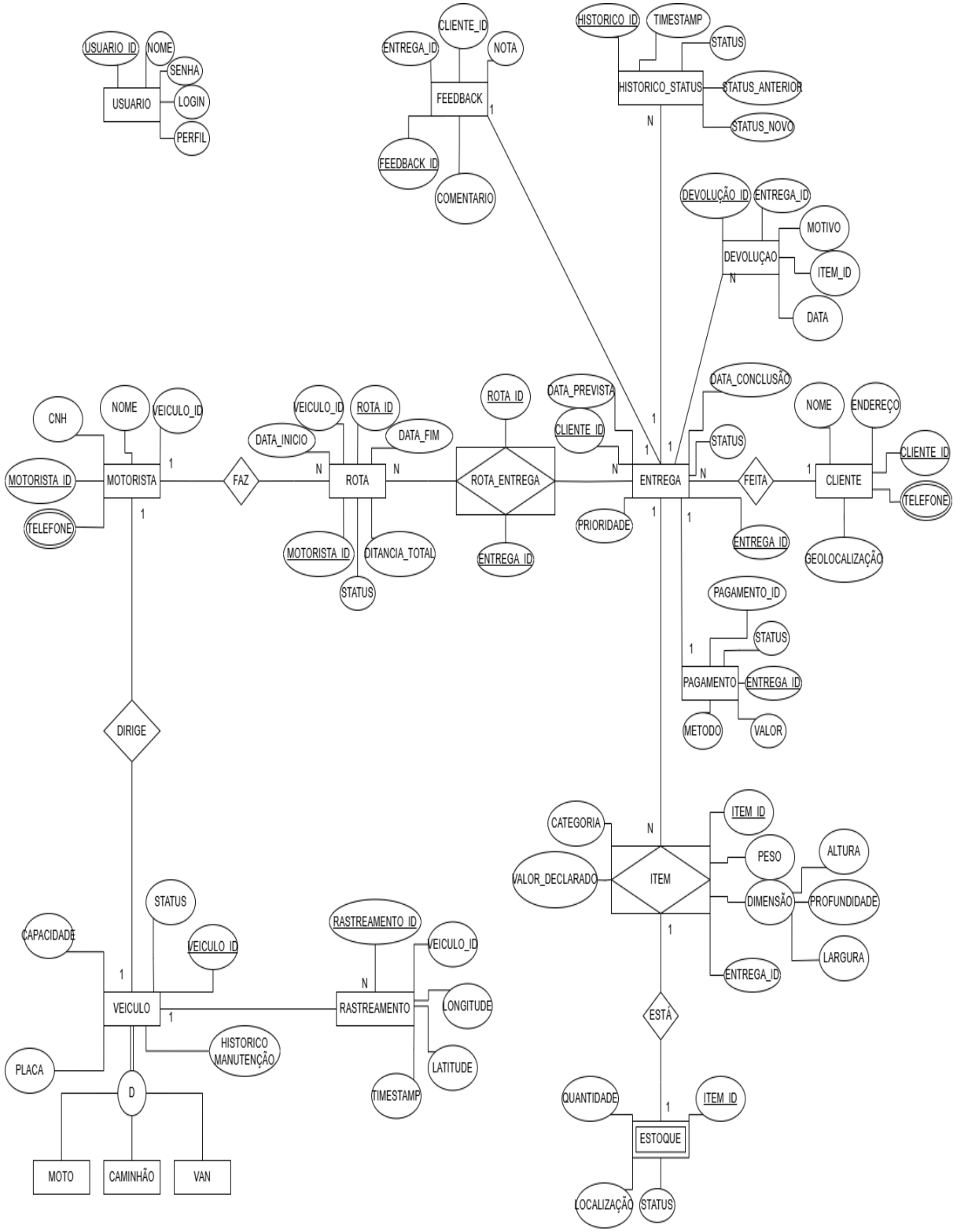
6. Rotas Otimizadas (Tabela *rota*):

- Define agrupamentos de entregas para um veículo em uma data específica.
- Campos: ID, veiculo_id, motorista_id, data_inicio, data_fim, distancia_total, status (planejada, em andamento, concluída).
- Tabela `rota_entrega`: Relaciona rotas a múltiplas entregas (relação muitos-para-muitos).

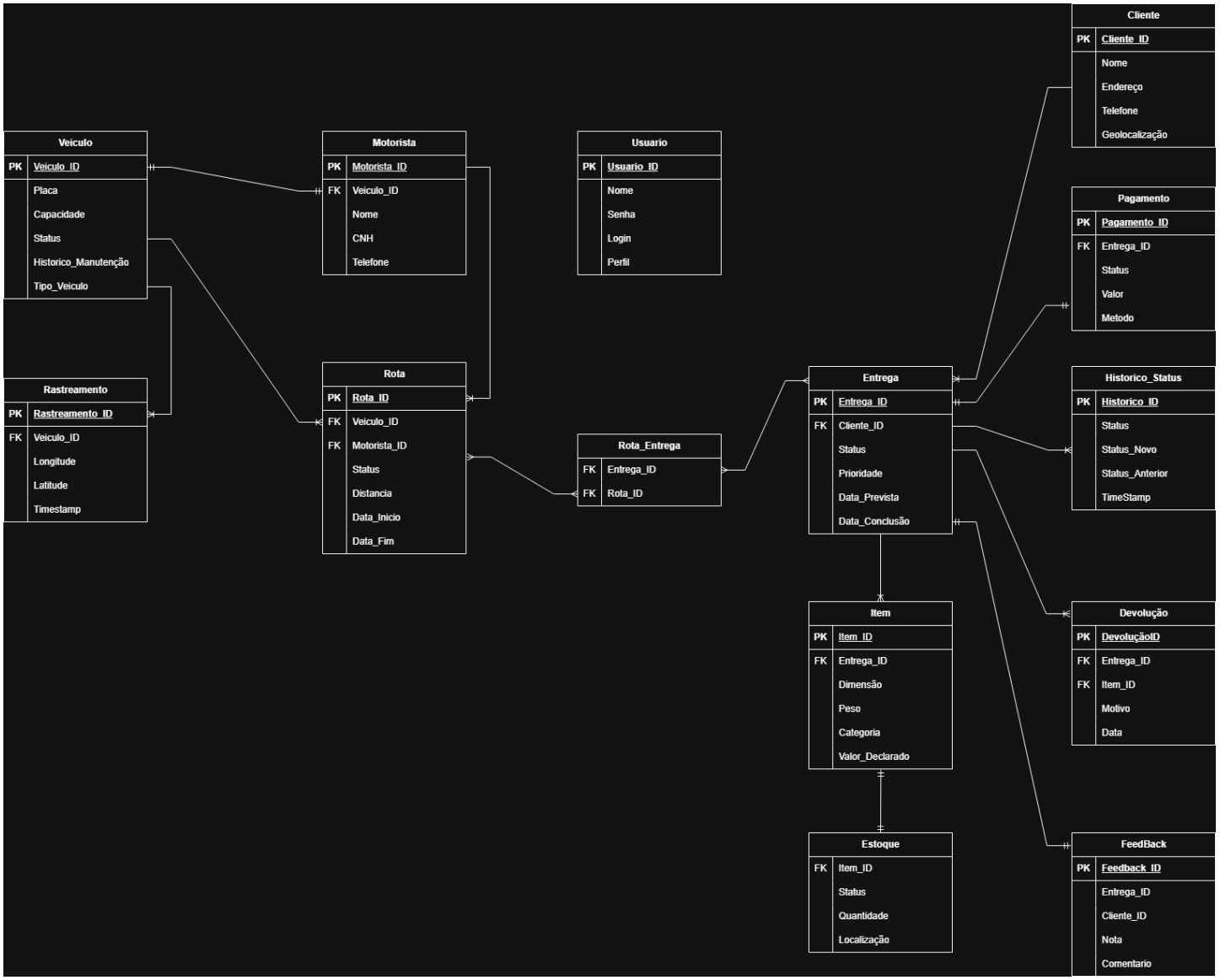
7. Usuários e Permissões (Tabela *usuario*):

- Gerencia perfis (administrador, motorista, gerente) com diferentes níveis de acesso.

Modelo Entidade-Relacionamento (ER)



Modelo Relacional



Script SQL

```sql
-- Script SQL para criar as tabelas no Azure SQL Database
-- Este script implementa as sugestões de normalização e melhorias.
-- Abordagem robusta para dropar tabelas com FKs, sem usar
sp_MSforeachtable.

-- ***** IMPORTANTE: Certifique-se de que você está conectado ao banco
de dados correto no seu cliente SQL (ex: VS Code) *****
-- USE [TrabalhoDeBD-SRL]; -- Este comando não é suportado para mudar de
DB em alguns clientes como o VS Code MSSQL.

-- PASSO 1: Desabilitar e Dropar TODAS as chaves estrangeiras existentes
-- Isso é feito de forma dinâmica para garantir que todas as FKs sejam
removidas,
-- permitindo que as tabelas sejam dropadas sem erros de dependência.
DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += N'ALTER TABLE '
    + QUOTENAME(OBJECT_SCHEMA_NAME(parent_object_id)) + '.'
    + QUOTENAME(OBJECT_NAME(parent_object_id))
    + ' DROP CONSTRAINT ' + QUOTENAME(name) + ';' + CHAR(13) + CHAR(10)
FROM sys.foreign_keys;

EXEC sp_executesql @sql;
PRINT 'Todas as chaves estrangeiras existentes foram removidas.';

-- PASSO 2: Remover tabelas existentes (todos os dados serão apagados!)
-- A ordem aqui se torna menos crítica após a remoção das FKs.
DROP TABLE IF EXISTS Carregamento;
DROP TABLE IF EXISTS Produto_A_Ser_Entregue;
DROP TABLE IF EXISTS Usuario;
DROP TABLE IF EXISTS Funcionario;
DROP TABLE IF EXISTS Cliente;
DROP TABLE IF EXISTS Dados_Rastreamento;
DROP TABLE IF EXISTS Veiculo;
DROP TABLE IF EXISTS Sede;
DROP TABLE IF EXISTS Pessoa;
DROP TABLE IF EXISTS Endereco;
PRINT 'Todas as tabelas existentes foram removidas (se existiam).';

-- PASSO 3: Criar as tabelas com a nova estrutura (na ordem correta de
dependência para criação)

-- Tabela Endereco (primeiro, pois muitas tabelas dependem dela)
CREATE TABLE Endereco (
```

```
    ID_Endereco INT IDENTITY(1,1) PRIMARY KEY, -- Auto-incremental
    CEP VARCHAR(10) NOT NULL,
    Estado VARCHAR(50) NOT NULL,
    Cidade VARCHAR(100) NOT NULL,
    Bairro VARCHAR(100) NOT NULL,
    Rua VARCHAR(200) NOT NULL,
    Numero VARCHAR(20) NOT NULL,
    Complemento VARCHAR(200)
);
PRINT 'Tabela Endereco criada.';

-- Tabela Sede (depende de Endereco)
CREATE TABLE Sede (
    ID_Sede INT IDENTITY(1,1) PRIMARY KEY,
    Tipo INT NOT NULL CHECK (Tipo IN (1, 2, 3)), -- 1 - Distribuição, 2
- Loja, 3 - Ambos
    ID_Endereco INT UNIQUE NOT NULL, -- Uma sede tem um endereço único
    Telefone VARCHAR(20),
    FOREIGN KEY (ID_Endereco) REFERENCES Endereco(ID_Endereco)
);
PRINT 'Tabela Sede criada.';

-- Tabela Pessoa (nova tabela para normalização de Cliente e
Funcionario)
CREATE TABLE Pessoa (
    Codigo_Pessoa INT IDENTITY(1,1) PRIMARY KEY,
    Nome VARCHAR(255) NOT NULL,
    RG VARCHAR(20),
    Telefone VARCHAR(20),
    Email VARCHAR(255),
    ID_Endereco INT NOT NULL, -- Endereço principal da pessoa
    FOREIGN KEY (ID_Endereco) REFERENCES Endereco(ID_Endereco)
);
PRINT 'Tabela Pessoa criada.';

-- Tabela Cliente (agora com FK para Pessoa, e campos de PF/PJ
opcionais)
CREATE TABLE Cliente (
    Codigo_Pessoa INT PRIMARY KEY, -- PK e FK para Pessoa
    Tipo_Cliente VARCHAR(2) NOT NULL CHECK (Tipo_Cliente IN ('PF',
'PJ')), -- Pessoa Física ou Pessoa Jurídica
    CPF VARCHAR(14), -- Para Pessoa Física
    Data_Nascimento DATE, -- Para Pessoa Física
    CNPJ VARCHAR(18), -- Para Pessoa Jurídica
    Nome_Empresa VARCHAR(255), -- Para Pessoa Jurídica
    -- Restrições para garantir que os campos corretos sejam preenchidos
```

```
    CONSTRAINT CHK_Cliente_PF_PJ CHECK (
        (Tipo_Cliente = 'PF' AND CPF IS NOT NULL AND Data_Nascimento IS
NOT NULL AND CNPJ IS NULL AND Nome_Empresa IS NULL) OR
        (Tipo_Cliente = 'PJ' AND CNPJ IS NOT NULL AND Nome_Empresa IS
NOT NULL AND CPF IS NULL AND Data_Nascimento IS NULL)
    ),
    FOREIGN KEY (Codigo_Pessoa) REFERENCES Pessoa(Codigo_Pessoa)
);
PRINT 'Tabela Cliente criada.';

-- Tabela Veiculo
CREATE TABLE Veiculo (
    Placa_Veiculo VARCHAR(10) PRIMARY KEY,
    Carga_Suportada DECIMAL(10, 2) NOT NULL, -- em kg
    Tipo VARCHAR(50) NOT NULL CHECK (Tipo IN ('Carro', 'Moto', 'Van',
'Caminhão')),
    Status VARCHAR(20) NOT NULL CHECK (Status IN ('Disponivel',
'Indisponivel'))
);
PRINT 'Tabela Veiculo criada.';

-- Tabela Funcionario (agora com FK para Pessoa, e campos de
especialização opcionais)
CREATE TABLE Funcionario (
    Codigo_Funcionario INT PRIMARY KEY, -- PK e FK para Pessoa
    CPF VARCHAR(14) UNIQUE NOT NULL, -- CPF é único para funcionário
    Departamento VARCHAR(50) NOT NULL, -- Ex: 'Entregas', 'Atendimento'
    Cargo VARCHAR(50) NOT NULL CHECK (Cargo IN ('Motorista', 'Auxiliar
de Logistica', 'Atendente', 'Gerente', 'Admin')), -- Adicionado 'Admin'
para tipo de usuário
    Placa_Veiculo VARCHAR(10), -- Para Motorista (FK para Veiculo)
    ID_Sede INT, -- Para Auxiliar de Logística, Atendente, Gerente (FK
para Sede)
    FOREIGN KEY (Codigo_Funcionario) REFERENCES Pessoa(Codigo_Pessoa),
    FOREIGN KEY (Placa_Veiculo) REFERENCES Veiculo(Placa_Veiculo), --
Adiciona FK para Veiculo
    FOREIGN KEY (ID_Sede) REFERENCES Sede(ID_Sede), -- Adiciona FK para
Sede
    -- Restrições para garantir que os campos corretos sejam preenchidos
    CONSTRAINT CHK_Funcionario_Cargo CHECK (
        (Cargo = 'Motorista' AND Placa_Veiculo IS NOT NULL AND ID_Sede
IS NULL) OR
        (Cargo IN ('Auxiliar de Logistica', 'Atendente', 'Gerente') AND
ID_Sede IS NOT NULL AND Placa_Veiculo IS NULL) OR
        (Cargo = 'Admin' AND Placa_Veiculo IS NULL) OR -- Ajuste: Admin
pode ter ID_Sede NULL
```

```
        (Cargo NOT IN ('Motorista', 'Auxiliar de Logistica',
'Atendente', 'Gerente', 'Admin') AND Placa_Veiculo IS NULL AND ID_Sede
IS NULL)
    )
);
PRINT 'Tabela Funcionario criada.';

-- Tabela Dados_Rastreamento
CREATE TABLE Dados_Rastreamento (
    ID_Rastreamento INT IDENTITY(1,1) PRIMARY KEY,
    Codigo_Rastreamento VARCHAR(50) UNIQUE NOT NULL,
    Nome_Destinatario VARCHAR(255) NOT NULL,
    CPF_Destinatario VARCHAR(14),
    ID_Endereco INT NOT NULL, -- Endereço de entrega
    Cidade VARCHAR(100) NOT NULL,
    Estado VARCHAR(50) NOT NULL,
    Telefone_Destinatario VARCHAR(20),
    FOREIGN KEY (ID_Endereco) REFERENCES Endereco(ID_Endereco)
);
PRINT 'Tabela Dados_Rastreamento criada.';

-- Tabela Produto_A_Ser_Entregue (com ID_Remetente e ID_Destinatario)
CREATE TABLE Produto_A_Ser_Entregue (
    ID_Produto INT IDENTITY(1,1) PRIMARY KEY,
    Peso DECIMAL(10, 2) NOT NULL, -- em kg
    Status_Entrega VARCHAR(50) NOT NULL, -- Ex: 'Em Processamento', 'Em
Transito', 'Entregue', 'Cancelado'
    Data_Chegada_CD DATE NOT NULL,
    Data_Prevista_Entrega DATE,
    Tipo_Produto VARCHAR(50) NOT NULL CHECK (Tipo_Produto IN ('Fragil',
'Perecivel', 'Comum')),
    ID_Remetente INT NOT NULL, -- FK para Pessoa (quem enviou)
    ID_Destinatario INT NOT NULL, -- FK para Pessoa (quem vai receber)
    Codigo_Funcionario_Motorista INT, -- FK para Funcionario (Motorista)
    ID_Rastreamento INT UNIQUE NOT NULL, -- FK para Dados_Rastreamento
    FOREIGN KEY (ID_Remetente) REFERENCES Pessoa(Codigo_Pessoa),
    FOREIGN KEY (ID_Destinatario) REFERENCES Pessoa(Codigo_Pessoa),
    FOREIGN KEY (Codigo_Funcionario_Motorista) REFERENCES
Funcionario(Codigo_Funcionario),
    FOREIGN KEY (ID_Rastreamento) REFERENCES
Dados_Rastreamento(ID_Rastreamento)
);
PRINT 'Tabela Produto_A_Ser_Entregue criada.';

-- Tabela Carregamento
CREATE TABLE Carregamento (
```

```
    ID_Carregamento INT IDENTITY(1,1) PRIMARY KEY,
    Placa_Veiculo VARCHAR(10) NOT NULL, -- FK para Veiculo
    ID_Produto INT NOT NULL, -- FK para Produto_A_Ser_Entregue
    Data_Carregamento DATETIME NOT NULL DEFAULT GETDATE(),
    FOREIGN KEY (Placa_Veiculo) REFERENCES Veiculo(Placa_Veiculo),
    FOREIGN KEY (ID_Produto) REFERENCES
Produto_A_Ser_Entregue(ID_Produto),
    CONSTRAINT UQ_Carregamento UNIQUE (Placa_Veiculo, ID_Produto,
Data_Carregamento) -- Evita carregamentos duplicados exatos
);
PRINT 'Tabela Carregamento criada.';

-- Tabela Usuario (agora com FK para Pessoa e campo Tipo_Usuario, e
senha hashed)
CREATE TABLE Usuario (
    Login VARCHAR(100) PRIMARY KEY,
    Senha_Hash VARCHAR(255) NOT NULL, -- Para armazenar o hash da senha
    Codigo_Pessoa INT UNIQUE NOT NULL, -- FK para Pessoa (um usuário
está associado a uma pessoa)
    Tipo_Usuario VARCHAR(50) NOT NULL CHECK (Tipo_Usuario IN ('Cliente',
'Motorista', 'Auxiliar de Logistica', 'Atendente', 'Gerente', 'Admin')),
-- Define o tipo de acesso
    FOREIGN KEY (Codigo_Pessoa) REFERENCES Pessoa(Codigo_Pessoa)
);
PRINT 'Tabela Usuario criada.';

PRINT 'Script de criação de tabelas concluído com sucesso.';
```