

Autoshop

tcm24sibdg03

Uma oficina gere os seus processos de forma manual – com registos dispersos em papel ou folhas de cálculo.

Esta abordagem dificulta:

- O controlo de agendamentos e serviços realizados;
- A rastreabilidade de ações anteriores em cada veículo;
- A gestão eficiente de clientes e da informação associada;

Isto pode levar a erros, perda de dados importantes e menor qualidade no serviço ao cliente.

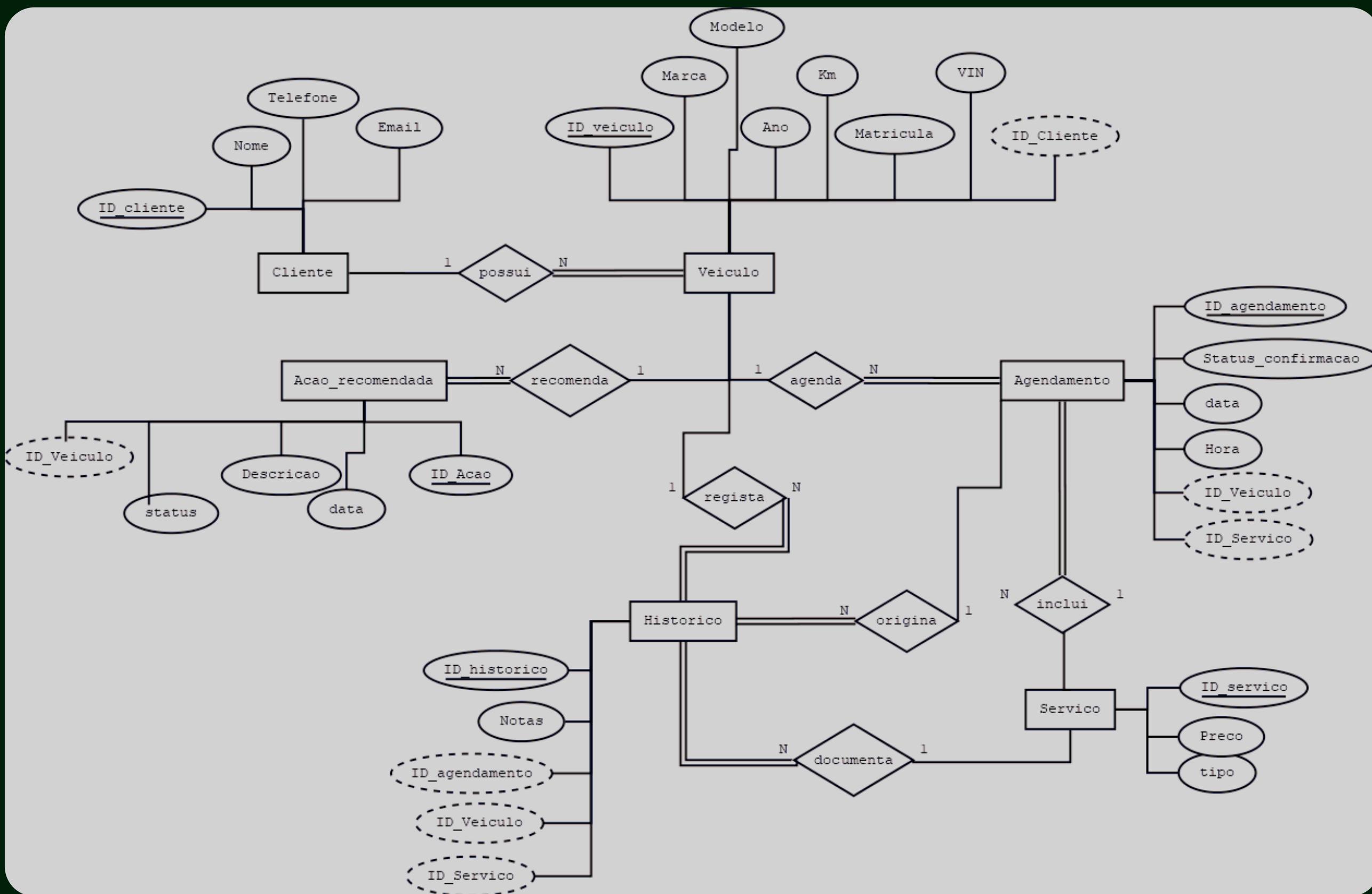
O problema

O objetivo

Desenvolver um sistema de informação para apoio à gestão de uma oficina automóvel, com foco:

- Na modelação de uma base de dados relacional normalizada;
- Na implementação de uma API RESTful para gerir clientes, veículos, agendamentos e serviços;
- Na criação de funcionalidades que permitam automatizar o registo e consulta de operações;

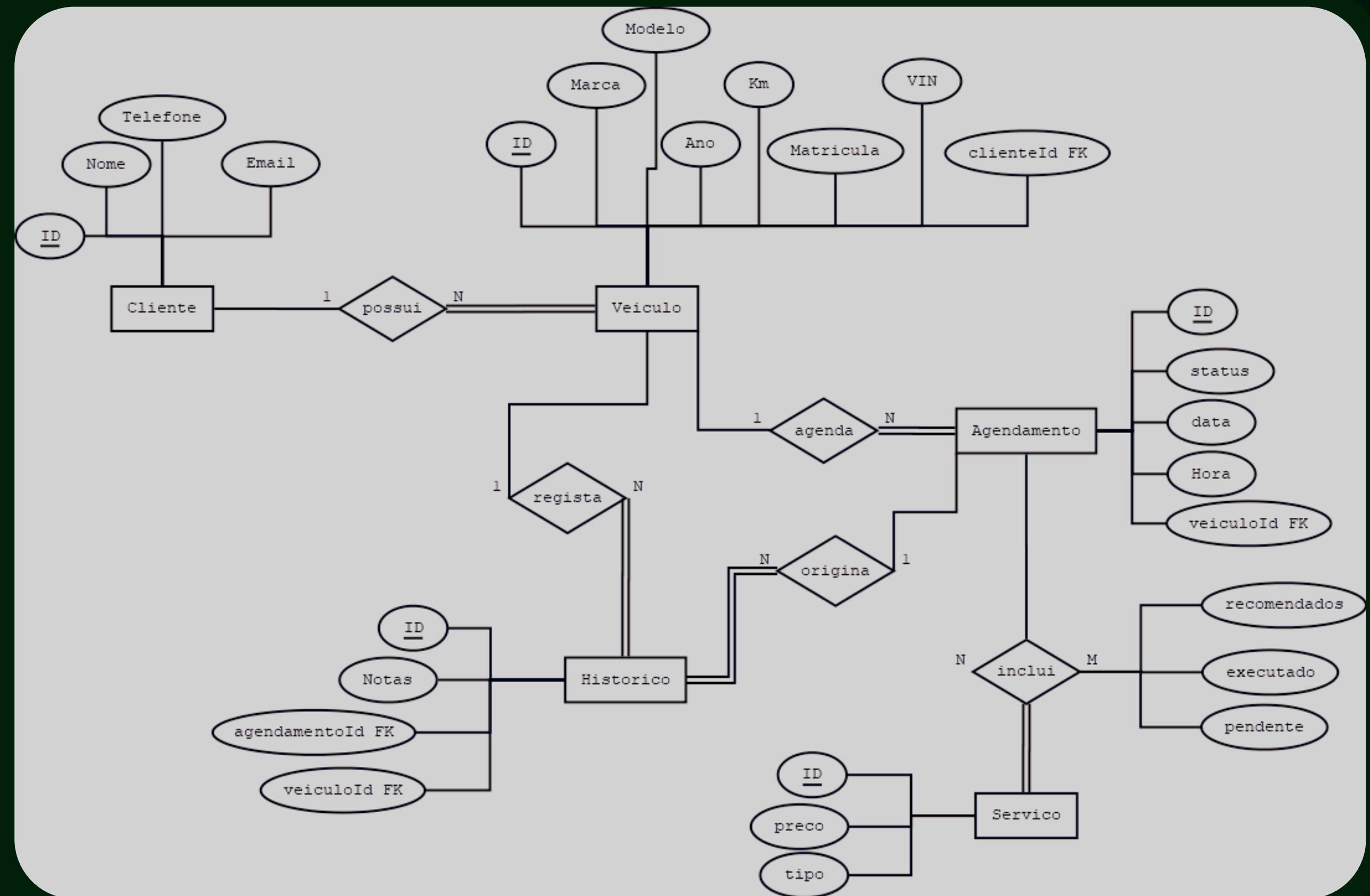
Diagrama E/A – REBD



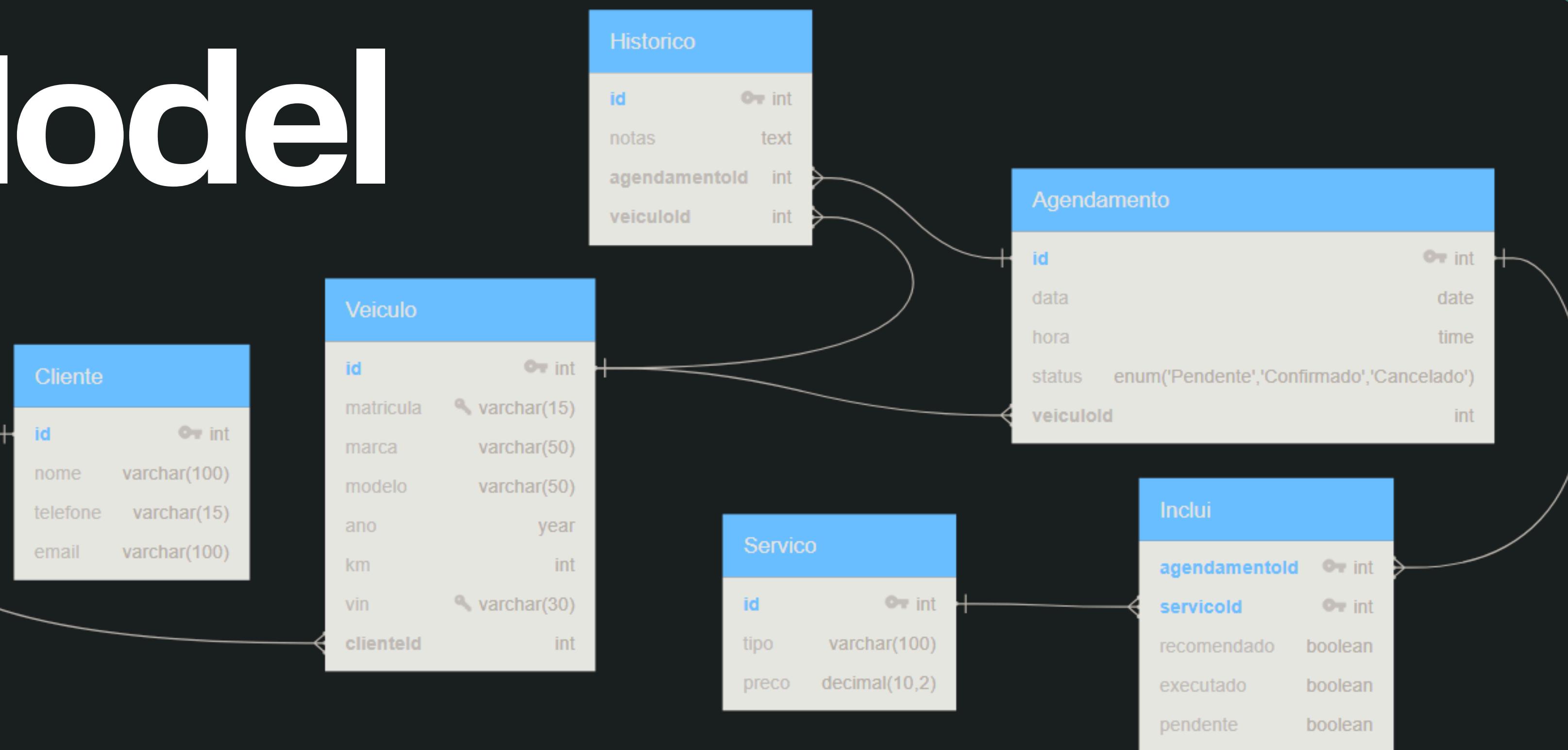
Este diagrama representa a primeira estrutura relacional completa, criada a partir da análise inicial do problema no relatório REBD.

Diagrama E/A - RPF

O diagrama final corrige erros de representação, elimina redundâncias e reflete a estrutura real usada na implementação da API.



Database Model



SQL - Exemplos DDL

```
CREATE TABLE IF NOT EXISTS `Inclui` (
    `id` INT,
    `agendamentoId` INT NOT NULL,
    `servicoId` INT NOT NULL,
    `recomendado` BOOLEAN DEFAULT FALSE,
    `executado` BOOLEAN DEFAULT FALSE,
    `pendente` BOOLEAN DEFAULT TRUE,
    PRIMARY KEY (agendamentoId, servicoId),
    FOREIGN KEY (agendamentoId) REFERENCES Agendamento(id),
    FOREIGN KEY (servicoId) REFERENCES Servico(id)
);
```

```
CREATE TABLE IF NOT EXISTS `Veiculo` (
    `id` INT AUTO_INCREMENT PRIMARY KEY,
    `matricula` VARCHAR(15) NOT NULL UNIQUE,
    `marca` VARCHAR(50) NOT NULL,
    `modelo` VARCHAR(50) NOT NULL,
    `ano` YEAR NOT NULL,
    `km` INT NOT NULL,
    `vin` VARCHAR(30) NOT NULL UNIQUE,
    `clienteId` INT NOT NULL,
    FOREIGN KEY (`clienteId`) REFERENCES `Cliente`(`id`)
);
```

```
CREATE TABLE IF NOT EXISTS `Agendamento` (
    `id` INT AUTO_INCREMENT PRIMARY KEY,
    `data` DATE NOT NULL,
    `hora` TIME NOT NULL,
    `status` ENUM('Pendente', 'Confirmado', 'Cancelado') NOT NULL,
    `veiculoId` INT NOT NULL,
    FOREIGN KEY (`veiculoId`) REFERENCES `Veiculo`(`id`)
);
```

API com LoopBack

Principais decisões técnicas:

- Criação manual da relação N:M (Inclui), usando @hasMany().through()
- Separação clara entre modelos principais (Cliente, Agendamento, etc.) e modelos de leitura (Historico)

Funcionalidades especiais:

- Lista de serviços é devolvida automaticamente nos GET /agendamentos via inclusão da relação
- Inserção e eliminação automática no Historico ao criar/apagar agendamentos
- Controladores personalizados para Inclui:
 - GET /agendamentos/{id}/servicos/{servicold}
 - PATCH e POST equivalentes

A implementação exigiu intervenção manual para ultrapassar limitações do gerador automático.

ENOTFOUND

ENOTFOUND localhost

SQL vs LoopBack Migrate

- ENUM ignorado:
 - status (em Agendamento) foi definido como `ENUM` no SQL, mas o migrate() ignora esta restrição – a API valida internamente, mas a base de dados aceita qualquer string
- Tamanhos não respeitados:
 - VARCHAR(100), VARCHAR(15) definidos no SQL → transformados em VARCHAR(512) por padrão no LoopBack

```
CREATE TABLE IF NOT EXISTS `Agendamento` (
  `id` INT AUTO_INCREMENT PRIMARY KEY,
  `data` DATE NOT NULL,
  `hora` TIME NOT NULL,
  `status` ENUM('Pendente', 'Confirmado', 'Cancelado') NOT NULL,
  `veiculoId` INT NOT NULL,
  FOREIGN KEY (`veiculoId`) REFERENCES `Veiculo`(`id`)
);
```

```
DROP TABLE IF EXISTS `agendamento`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `agendamento` (
  `id` int NOT NULL AUTO_INCREMENT,
  `data` varchar(512) NOT NULL,
  `hora` varchar(512) NOT NULL,
  `status` varchar(512) NOT NULL,
  `veiculoId` int NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

SQL vs LoopBack Migrate

- Output diferente na API
 - Antes do migrate, data era interpretada como objeto Date (mesmo estando declarado como string na API)
 - Após migrate, passa a ser interpretado como string

```
{  
  "id": 1,  
  "data": "Wed May 28 2025 01:00:00 GMT+0100 (Western European Summer Time)",  
  "hora": "09:00:00",  
  "status": "Cancelado",  
  "veiculoId": 2,  
  "servicos": [  
    {  
      "id": 2,  
      "tipo": "Troca de Óleo",  
      "preco": 70  
    }  
  ]  
}
```

```
{  
  "id": 1,  
  "data": "2025-05-28",  
  "hora": "09:00:00",  
  "status": "Cancelado",  
  "veiculoId": 2,  
  "servicos": [  
    {  
      "id": 2,  
      "tipo": "Troca de Óleo",  
      "preco": 70  
    }  
  ]  
}
```

Organização Endpoints Postman

- A coleção foi organizada por entidade, com os endpoints mais relevantes para a administração e funcionamento interno da oficina.
- Todos os métodos não utilizados foram omitidos deliberadamente, com foco na lógica do domínio e não na cobertura total CRUD.

The screenshot shows a Postman collection named 'autoshop-api / VEICULOS / GET VEHICLES BY CLIENT ID'. The request method is 'GET' and the URL is 'http://localhost:3000/veiculos?filter[where][clientId]=:id'. The 'Params' tab is selected, showing two query parameters: 'filter[where][clientId]' with value ':id' and an empty row for another parameter. Other tabs include 'Authorization', 'Headers (6)', 'Body', 'Scripts', and 'Settings'.

Endpoints Personalizados:

- Criados manualmente para responder a necessidades específicas que o LoopBack não gera automaticamente

Organização Endpoints Postman

ENTIDADE

ENDPOINTS

CLIENTES

GET POST PUT DELETE
GET /clientes/:ID

VEICULOS

GET POST PUT DELETE
GET /veiculo/:ID
GET /clientes/:id/veiculo

AGENDAMENTOS

GET POST PUT DELETE
GET /agendamentos/:id

SERVICOS

GET POST PUT DELETE
GET /servicos/:id

INCLUI

GET POST PATCH

HISTORICO

GET
GET /historicos/veiculo/:id

Organização Endpoints Postman

```
@get('/agendamentos/{agendamentoId}/servicos', {
  responses: {
    '200': {
      description: 'Lista de serviços incluídos num agendamento',
      content: {'application/json': {schema: {type: 'array', items: {'x-ts-type': Inclui}}}},
    },
  },
})
async findServicosPorAgendamento(
  @param.path.number('agendamentoId') agendamentoId: number,
): Promise<Inclui[]> {
  return this.incluiRepository.find({where: {agendamentoId}});
}
```

```
@post('/agendamentos')
@response(200, {
  description: 'Agendamento model instance',
  content: {'application/json': {schema: getModelSchemaRef(Agendamento)}},
})
async create(
  @requestBody({
    content: {
      'application/json': {
        schema: getModelSchemaRef(Agendamento, {
          title: 'NewAgendamento',
          exclude: ['id'],
        }),
      },
    },
  })
  agendamento: Omit<Agendamento, 'id'>,
): Promise<Agendamento> {
  const novo = await this.agendamentoRepository.create(agendamento);

  await this.historicoRepository.create({
    agendamentoId: novo.id,
    veiculoId: novo.veiculoId,
    notas: `Status: ${novo.status}`,
  });
  return novo;
}
```

```
@del('/agendamentos/{id}')
@response(204, {
  description: 'Agendamento DELETE success',
})
async deleteById(@param.path.number('id') id: number): Promise<void> {
  await this.historicoRepository.deleteAll({agendamentoId: id});
  await this.agendamentoRepository.deleteById(id);
}
```

video mockup interface

Base de Dados & API REST – AUTOSHOP

Projeto SIBD – 2024/2025
tcm24sibdg03

**Jonathan
Claro
A045207**