🔍 1. Tabela: `Cliente`

sql
CopiarEditar

```
CREATE TABLE Cliente (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    email VARCHAR(100)
);
```

✔ Por que existe:

- Representa o **usuário final** que solicita os serviços.

✔ Explicações:

- `id_cliente`: identificador único do cliente.
- `nome`: nome completo do cliente.
- `cpf`: usado para identificação e validação (único).
- `telefone` e `email`: canais de contato.

✔ Relacionamentos:

- Cada cliente pode criar **vários pedidos** → relação 1:N com `Pedido`.

🔍 2. Tabela: `Tecnico`

sql
CopiarEditar

```
CREATE TABLE Tecnico (
    id_tecnico INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(100) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL,
    senioridade ENUM('Junior', 'Pleno', 'Senior') NOT NULL,
    status ENUM('Disponivel', 'Ocupado', 'Inativo') DEFAULT
'Disponivel'
);
```

✔ Por que existe:

- Representa o **profissional** que executa os serviços.

✔ Explicações:

- `senha`: usada para login seguro.
- `senioridade`: classifica o nível técnico, o que ajuda na **alocação de pedidos por dificuldade**.
- `status`: define se está disponível para novos atendimentos.

✔ Relacionamentos:

- Pode ser atribuído a vários pedidos → 1:N com `Pedido`.
- Tem registros de geolocalização → 1:N com `Localizacao`.

🔍 3. Tabela: `Admin`

sql
CopiarEditar

```
CREATE TABLE Admin (
    id_admin INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL,
    tipo ENUM('SuperAdmin', 'Admin') NOT NULL
);
```

✔ Por que existe:

- Representa os **usuários com acesso administrativo**, como secretários e supervisores.

✔ Explicações:

- `tipo`: distingue permissões entre "SuperAdmin" (controle total) e "Admin" (operações básicas).
- Não precisa se relacionar com outras tabelas porque **atua na lógica do sistema**, e não diretamente nos dados dos pedidos.

🔍 4. Tabela: `Pedido`

sql
CopiarEditar

```
CREATE TABLE Pedido (
    id_pedido INT AUTO_INCREMENT PRIMARY KEY,
    descricao TEXT NOT NULL,
    categoria VARCHAR(100),
    dificuldade ENUM('Baixa', 'Média', 'Alta') NOT NULL,
    status ENUM('Aguardando', 'Em andamento', 'Concluído',
'Cancelado') DEFAULT 'Aguardando',
    prazo DATE,
    id_cliente INT NOT NULL,
    id_tecnico INT,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente),
    FOREIGN KEY (id_tecnico) REFERENCES Tecnico(id_tecnico)
);
```

✔ Por que existe:

- É o **núcleo da operação**. Representa cada solicitação de serviço feita no sistema.

✔ Explicações:

- `dificuldade`: ajuda na decisão de qual técnico deve pegar o serviço.

- `status`: rastreia em qual etapa o serviço está.
- `prazo`: indica a data limite para execução.

✔ Relacionamentos:

- FK `id_cliente`: identifica quem fez o pedido.
- FK `id_tecnico`: indica quem foi designado para o serviço (opcional no início).

🔍 5. Tabela: `Historico`

sql
CopiarEditar

```
CREATE TABLE Historico (
    id_historico INT AUTO_INCREMENT PRIMARY KEY,
    id_pedido INT NOT NULL,
    data_movimentacao DATETIME DEFAULT CURRENT_TIMESTAMP,
    status_anterior ENUM('Aguardando', 'Em andamento', 'Concluído',
'Cancelado'),
    status_novo ENUM('Aguardando', 'Em andamento', 'Concluído',
'Cancelado'),
    observacoes TEXT,
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido)
);
```

✔ Por que existe:

- Guarda o **rastro de todas as mudanças de status** de um pedido.

✔ Explicações:

- `status_anterior` e `status_novo`: ajudam a entender a progressão do serviço.
- `observacoes`: registra motivo de mudanças, reagendamentos etc.

✔ Relacionamentos:

- FK `id_pedido`: cada linha do histórico está ligada a um pedido → relação 1:N.

🔍 6. Tabela: `Localizacao`

sql
CopiarEditar

```
CREATE TABLE Localizacao (
    id_localizacao INT AUTO_INCREMENT PRIMARY KEY,
    id_tecnico INT NOT NULL,
    latitude DECIMAL(10,8),
    longitude DECIMAL(11,8),
    data_hora DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_tecnico) REFERENCES Tecnico(id_tecnico)
);
```

✔ Por que existe:

- Registra a **posição geográfica do técnico**, útil para o mapa em tempo real e logística.

✔ Explicações:

- Armazena coordenadas e horário da coleta da localização.
- Serve para calcular distâncias entre técnico e local de atendimento.

✔ Relacionamentos:

- FK `id_tecnico`: define a quem pertence essa localização → 1 técnico pode ter muitas localizações (1:N).

☑ Considerações Técnicas Gerais

Decisão	Justificativa

`INT AUTO_INCREMENT`	Simples de indexar, garante unicidade
`ENUM`	Restringe valores válidos e previne erros
`FOREIGN KEY`	Garante integridade relacional entre as tabelas
Separação de `Admin`, `Cliente`, `Tecnico`	Clareza na lógica de permissão e papéis
Tabela de `Histórico` separada	Garante rastreabilidade e auditabilidade
Localização separada de `Tecnico`	Registro contínuo ao longo do tempo, sem sobrescrever

---

Banco Completo:

```
CREATE DATABASE IF NOT EXISTS sistema_servicos_tecnicos;
USE sistema_servicos_tecnicos;

-- Tabela de Clientes
CREATE TABLE Cliente (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    email VARCHAR(100)
);

-- Tabela de Técnicos
CREATE TABLE Tecnico (
    id_tecnico INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL,
    senioridade ENUM('Junior', 'Pleno', 'Senior') NOT NULL,
    status ENUM('Disponivel', 'Ocupado', 'Inativo') DEFAULT 'Disponivel'
);

-- Tabela de Admins
CREATE TABLE Admin (
    id_admin INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
```

```
    senha VARCHAR(255) NOT NULL,
    tipo ENUM('SuperAdmin', 'Admin') NOT NULL
);

-- Tabela de Pedidos
CREATE TABLE Pedido (
    id_pedido INT AUTO_INCREMENT PRIMARY KEY,
    descricao TEXT NOT NULL,
    categoria VARCHAR(100),
    dificuldade ENUM('Baixa', 'Média', 'Alta') NOT NULL,
    status ENUM('Aguardando', 'Em andamento', 'Concluído', 'Cancelado') DEFAULT
'Aguardando',
    prazo DATE,
    id_cliente INT NOT NULL,
    id_tecnico INT,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente),
    FOREIGN KEY (id_tecnico) REFERENCES Tecnico(id_tecnico)
);

-- Tabela de Histórico de Pedidos
CREATE TABLE Historico (
    id_historico INT AUTO_INCREMENT PRIMARY KEY,
    id_pedido INT NOT NULL,
    data_movimentacao DATETIME DEFAULT CURRENT_TIMESTAMP,
    status_anterior ENUM('Aguardando', 'Em andamento', 'Concluído', 'Cancelado'),
    status_novo ENUM('Aguardando', 'Em andamento', 'Concluído', 'Cancelado'),
    observacoes TEXT,
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido)
);

-- Tabela de Localizações de Técnicos
CREATE TABLE Localizacao (
    id_localizacao INT AUTO_INCREMENT PRIMARY KEY,
    id_tecnico INT NOT NULL,
    latitude DECIMAL(10,8),
    longitude DECIMAL(11,8),
    data_hora DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_tecnico) REFERENCES Tecnico(id_tecnico)
);
```

📘 Tipos de Relacionamento:

Tipo	Significado

1:1 (Um para Um)	Um registro em A está ligado a **um único** registro em B, e vice-versa.
1:N (Um para Muitos)	Um registro em A pode estar ligado a **muitos** registros em B.
N:M (Muitos para Muitos)	Precisa de uma tabela intermediária (não há no seu DER por enquanto).

📊 Relações no seu DER:

Vamos verificar uma a uma:

◇ `Cliente` → `Pedido`

Tipo: 1:N
Por quê? Um cliente pode fazer vários pedidos. Mas cada pedido pertence a **um único cliente**.

◇ `Técnico` → `Pedido`

Tipo: 1:N (opcional)
Por quê? Um técnico pode realizar vários pedidos ao longo do tempo. Um pedido só pode estar com **um técnico por vez** (mesmo que seja trocado depois).
📌 Observação: o `id_tecnico` pode ser `NULL` até o pedido ser alocado.

◇ `Pedido` → `Histórico`

Tipo: 1:N
Por quê? Um pedido pode ter **várias mudanças de status**, que ficam registradas na tabela `historico`.

◇ `Técnico` → `Localizacao`

Tipo: 1:N
Por quê? Um técnico pode ter **vários registros de localização** ao longo do tempo. Cada registro é ligado a **um único técnico**.

◇ `Admin`

Sem relacionamento direto
Por quê? A tabela `admin` atua como controle de acesso e gestão, e não precisa estar ligada diretamente às outras por FK. É uma boa separação.

☑ E onde tem 1:1 (Um para Um)?

👉 No seu DER **não há nenhum relacionamento 1:1 explícito**, o que **é bom** nesse caso! 1:1 normalmente é usado em situações bem específicas, como:

- Separar informações sensíveis (ex: dados bancários) em outra tabela.
- Dividir responsabilidade entre tipos de usuários (ex: `usuario - cliente` e `usuario - técnico` com mesma PK).

Mas no seu sistema atual, não há necessidade de um relacionamento 1:1, pois as entidades são bem distintas.

🔁 Recapitulando:

Entidade A → Entidade B	Tipo	Exemplo
Cliente → Pedido	1:N	João tem 5 pedidos
Técnico → Pedido	1:N	Maria realizou 3 serviços
Pedido → Histórico	1:N	Pedido #12 teve 4 mudanças de status
Técnico → Localização	1:N	Carlos teve 10 posições geográficas registradas
Admin → *	(indep.)	Atua sobre o sistema, mas sem FK direta

Guia para fazer o figma

GUIA DE TELAS PARA CRIAÇÃO NO FIGMA

1. Tela de Login (Cliente e Técnico/Admin)
Nome no Figma: Login

Componentes:
Logotipo da empresa

Botão "Entrar com Google" (cliente)

Campo de e-mail e senha (técnicos e administradores)

Botão "Entrar"

Link “Esqueceu a senha?”

💡 Dica: use autenticação via Google apenas para o cliente. Técnicos e admins usam login tradicional.

2. Tela de Solicitação de Orçamento (Cliente)
Nome no Figma: Novo Pedido - Cliente

Componentes:
Campo de descrição do problema

Campo de categoria (dropdown)

Campo de dificuldade (dropdown: Básico, Intermediário, Avançado)

Upload de imagem (opcional)

Campos finais:

Nome completo

CPF

Telefone

Botão "Enviar pedido"

💡 Dica: Design clean, mobile-friendly. Ideal para um card central com campos verticais.

3. Dashboard do Admin (Gestor)
Nome no Figma: Dashboard - Admin

Estrutura:

Drawer lateral com seções:

Pedidos

Técnicos

Histórico

Mapa

Contas

Cards de pedidos em andamento (visão geral):

Número do pedido

Cliente

Técnico atribuído

Dificuldade (com cor)

Status

Prazo

Botões "Detalhes", "Reatribuir", "Finalizar"

Filtro no topo por: status, dificuldade, técnico

💡 Dica: Use cores para destacar status (ex: verde = concluído, amarelo = em andamento, vermelho = atrasado).

4. Tela de Detalhe do Pedido (Admin)
Nome no Figma: Detalhe do Pedido

Componentes:
Dados completos do cliente

Descrição do serviço

Dificuldade

Status atual

Lista de técnicos disponíveis

Botão "Associar técnico"

Botão "Aprovar solicitação"

Histórico de movimentações do pedido

💡 Dica: pode ser um modal ou uma nova tela. Coloque ações em destaque com botões coloridos.

5. Tela de Lista de Técnicos (Admin)
Nome no Figma: Técnicos

Componentes:
Tabela ou cards com:

Nome do técnico

Senioridade (ícone ou cor)

Status (ocupado, disponível)

Histórico de pedidos

Botão "Adicionar técnico"

Filtro por status/senioridade

6. Tela de Mapa com Técnicos (Admin)
Nome no Figma: Mapa - Técnicos

Componentes:
Mapa central (Google Maps style)

Pins com localização dos técnicos

Tooltip ao clicar no pin com:

Nome do técnico

Status

Botão "Ver detalhes"

💡 Dica: use pins de cores diferentes para status (disponível/ocupado).

7. Tela do Técnico (Painel próprio)
Nome no Figma: Dashboard - Técnico

Componentes:
Lista de pedidos atribuídos

Com botão para alterar status: "Em andamento", "Concluído", "Reagendar"

Campo de observações rápidas

Seção “Pedidos disponíveis”

Com botão "Solicitar execução"

Histórico de serviços realizados

Notificações recentes

Mini mapa com localização atual (caso habilitado)

💡 Dica: Faça com layout otimizado para celular!

8. Tela de Histórico (Admin)
Nome no Figma: Histórico

Componentes:
Filtros por:

Data

Técnico

Cliente

Status

Tabela com:

Pedido

Data de abertura e conclusão

Técnico

Status final

Botões:

"Exportar PDF"

"Exportar CSV"

9. Tela de Gerenciamento de Contas
Nome no Figma: Contas

Componentes:
Lista de usuários com nome, tipo de acesso

Botão "Adicionar usuário"

Ações: editar, remover

Separação por tipo:

Super Admin

Admin

Técnico

🎨 Estilo Visual (Sugestão)

Elemento	Estilo
Fonte	Inter ou Roboto (14px base)
Cor primária	Azul (#1976D2 ou similar)
Cor secundária	Cinza claro
Status	Verde (concluído), Amarelo (em andamento), Vermelho (atrasado)
Ícones	Google Material Icons
Botões	border-radius: 8px, hover com leve sombra

📌 Sugestões finais
Crie 1 página por perfil no Figma: Cliente, Admin, Técnico.

Use Auto Layout para facilitar responsividade.

Use Components para reutilizar cards, botões e campos.

Crie Prototyping Links entre telas para simular a navegação.

Se quiser, posso te entregar isso tudo também em PDF ou com imagens das telas (wireframes). Posso até gerar o layout de algumas dessas interfaces pra você colar no Figma direto.

Quer que eu comece por alguma tela em específico com visual?

sistema-servicos/

```
├── index.html                (Login)
├── cliente/
│   └── novo-pedido.html
├── admin/
│   ├── dashboard.html
│   ├── detalhe-pedido.html
│   ├── tecnicos.html
│   ├── mapa.html
│   ├── historico.html
│   └── contas.html
├── tecnico/
│   └── dashboard.html
├── assets/
│   ├── css/
│   │   └── style.css
│   ├── js/
│   │   └── main.js
│   └── img/
│       └── logo.png
```