**Guia Didático — Entendendo o schema.prisma**

O arquivo **schema.prisma** é o **coração** do Prisma: nele definimos como será o **mapeamento** entre nosso banco de dados e o código da aplicação (models → tabelas, campos, tipos, relacionamentos).

1. **Cabeçalho — Gerador e Fonte de Dados**

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

* **generator client**: diz ao Prisma que deve gerar o **Prisma Client** (biblioteca JS/TS para acessar o banco).
* **provider**: aqui está "prisma-client-js", que é o gerador oficial para JavaScript/TypeScript.
* **datasource db**: define de onde vêm os dados.
  + provider: tipo de banco usado (postgresql no nosso caso, mas pode ser mysql, sqlite, mongodb...).
  + url: conexão para o banco, lida do .env com env("DATABASE\_URL").

**Resumo:**  
É aqui que dizemos *qual banco usar* e *como gerar o client*.

1. **Models — Estrutura das Tabelas**

Cada model é como uma **tabela** no banco de dados.  
Os **campos** são as colunas e as **anotações** (@id, @default, @relation, etc.) definem chaves, valores padrão, relacionamentos e restrições.

**Modelo 1 — TipoDeficiencia**

model TipoDeficiencia {

id Int @id @default(autoincrement())

nome String @unique

subtipos SubtipoDeficiencia[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

* id: chave primária (@id), numérica (Int) e autoincrementada.
* nome: texto (String) único (@unique).
* subtipos: relação 1:N com SubtipoDeficiencia.
* createdAt / updatedAt: campos de auditoria.
  + @default(now()): preenche com a data/hora atual na criação.
  + @updatedAt: atualiza automaticamente na alteração.

**Modelo 2 — SubtipoDeficiencia**

model SubtipoDeficiencia {

id Int @id @default(autoincrement())

nome String

tipoId Int

tipo TipoDeficiencia @relation(fields: [tipoId], references: [id], onDelete: Cascade)

barreiras SubtipoBarreira[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

@@unique([tipoId, nome])

}

* tipoId: chave estrangeira para TipoDeficiencia.
* tipo: define a relação com o model TipoDeficiencia:
  + fields: campo local usado como FK.
  + references: campo no model pai (id) que é referenciado.
  + onDelete: Cascade: se apagar um tipo, apaga todos os subtipos ligados.
* barreiras: relação N:N com Barreira via tabela de junção SubtipoBarreira.
* @@unique([tipoId, nome]): índice único composto → evita subtipos duplicados dentro do mesmo tipo.

**Modelo 3 — Barreira**

model Barreira {

id Int @id @default(autoincrement())

descricao String @unique

subtipos SubtipoBarreira[]

acessibilidades BarreiraAcessibilidade[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

* descricao: descrição única da barreira.
* Relações:
  + subtipos: N:N com SubtipoDeficiencia via SubtipoBarreira.
  + acessibilidades: N:N com Acessibilidade via BarreiraAcessibilidade.

**Modelo 4 — Acessibilidade**

model Acessibilidade {

id Int @id @default(autoincrement())

descricao String @unique

barreiras BarreiraAcessibilidade[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

* descricao: descrição única da acessibilidade.
* barreiras: N:N com Barreira.

1. **Tabelas de Junção (Relacionamentos N:N)**

**SubtipoBarreira**

model SubtipoBarreira {

subtipoId Int

barreiraId Int

subtipo SubtipoDeficiencia @relation(fields: [subtipoId], references: [id], onDelete: Cascade)

barreira Barreira @relation(fields: [barreiraId], references: [id], onDelete: Cascade)

@@id([subtipoId, barreiraId])

@@index([barreiraId])

}

* Liga SubtipoDeficiencia ↔ Barreira.
* PK composta: (subtipoId, barreiraId).
* Índice em barreiraId para facilitar buscas.

**BarreiraAcessibilidade**

model BarreiraAcessibilidade {

barreiraId Int

acessibilidadeId Int

barreira Barreira @relation(fields: [barreiraId], references: [id], onDelete: Cascade)

acessibilidade Acessibilidade @relation(fields: [acessibilidadeId], references: [id], onDelete: Cascade)

@@id([barreiraId, acessibilidadeId])

@@index([acessibilidadeId])

}

* Liga Barreira ↔ Acessibilidade.
* PK composta: (barreiraId, acessibilidadeId).
* Índice em acessibilidadeId.

1. **Resumo das Relações**

* **TipoDeficiencia** 1:N **SubtipoDeficiencia**
* **SubtipoDeficiencia** N:N **Barreira** (via SubtipoBarreira)
* **Barreira** N:N **Acessibilidade** (via BarreiraAcessibilidade)

Representação simplificada:

TipoDeficiencia --< SubtipoDeficiencia >-- SubtipoBarreira --< Barreira >-- BarreiraAcessibilidade >-- Acessibilidade

1. **Observações Importantes**

* **Nomes no singular**: cada model representa uma entidade (tabela).
* **Campos createdAt/updatedAt**: boas práticas para rastrear mudanças.
* **Índices únicos e compostos**: garantem integridade e evitam duplicidade.
* **onDelete: Cascade**: facilita limpeza automática em relações dependentes.
* **Tabelas de junção explícitas**: permitem evoluir facilmente (adicionar campos extras no futuro).