**tabela Users**

CREATE TABLE Users (

user\_id SERIAL PRIMARY KEY, -- ID único com autoincremento

first\_name VARCHAR(50) NOT NULL, -- Primeiro nome do usuário

last\_name VARCHAR(50) NOT NULL, -- Sobrenome do usuário

user\_status BOOLEAN NOT NULL, -- Status do usuário (ativo/inativo, representado como true/false)

user\_admin BOOLEAN DEFAULT FALSE, -- Indica se o usuário é administrador

password VARCHAR(255) NOT NULL, -- Hash da senha do usuário

country VARCHAR(3) NOT NULL, -- Código de país (ISO Alpha-3)

state VARCHAR(2) NOT NULL, -- Código de estado (ISO Alpha-2)

city VARCHAR(100) NOT NULL, -- Nome da cidade

first\_signin TIMESTAMP, -- Data e hora do primeiro login

last\_signin TIMESTAMP, -- Data e hora do último login

first\_signup TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Data e hora do registro inicial

last\_signup TIMESTAMP, -- Data e hora da última atualização do registro

google\_signin BOOLEAN DEFAULT FALSE -- Indica se foi utilizado login pelo Google

);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**tabela Plans**

CREATE TABLE Plans (

plan\_id SERIAL PRIMARY KEY, -- ID único com autoincremento

plan\_name VARCHAR(40) NOT NULL, -- Nome do plano

plan\_description TEXT, -- Descrição do plano

plan\_status BOOLEAN NOT NULL, -- Status do plano (ativo/inativo, representado como true/false)

plan\_validity TIMESTAMP NOT NULL, -- Validade do plano (data e hora)

plan\_price NUMERIC(10, 2) NOT NULL, -- Preço do plano com até 2 casas decimais

created\_by VARCHAR(50) NOT NULL, -- Nome do usuário que criou o plano

updated\_by TIMESTAMP, -- Data e hora da última atualização do plano

num\_plan\_users INT DEFAULT 0, -- Número de usuários permitidos no plano

num\_plan\_companies INT DEFAULT 0 -- Número de empresas permitidas no plano

);

### **Detalhes:**

1. **plan\_status**:  
   * Utilizei BOOLEAN para representar o status (ativo = true, inativo = false).
2. **plan\_validity**:  
   * Define a data e hora de validade do plano utilizando o tipo TIMESTAMP.
3. **plan\_price**:  
   * Usei NUMERIC(10, 2) para suportar valores decimais, com até 10 dígitos no total e 2 após o ponto decimal.
4. **created\_by**:  
   * Utilizei VARCHAR(50) para armazenar o nome do usuário que criou o plano.
5. **updated\_by**:  
   * Armazena a data e hora da última modificação no plano.
6. **num\_plan\_users e num\_plan\_companies**:  
   * São inteiros (INT) que possuem valor padrão 0, indicando nenhum limite inicial.

Se precisar de ajustes ou quiser adicionar constraints adicionais, como CHECK, ou relacionamentos com outras tabelas, me avise!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**tabela Companies**

CREATE TABLE Companies (

company\_id SERIAL PRIMARY KEY, -- ID único da empresa com autoincremento

company\_name VARCHAR(100) NOT NULL, -- Razão Social da Empresa

company\_cnpj VARCHAR(18) UNIQUE NOT NULL, -- CNPJ da empresa (formato XX.XXX.XXX/XXXX-XX)

company\_address VARCHAR(50) NOT NULL, -- Endereço da empresa

number\_address INT NOT NULL, -- Número do endereço

country CHAR(3) NOT NULL, -- Código do país (ISO Alpha-3)

state CHAR(2) NOT NULL, -- Código do estado (ISO Alpha-2)

city VARCHAR(40) NOT NULL, -- Nome da cidade

segment INT NOT NULL REFERENCES Segment(segment\_id), -- Referência à tabela Segment

tax\_regime INT NOT NULL REFERENCES TaxRegime(tax\_regime\_id), -- Referência à tabela TaxRegime

num\_registered\_users INT DEFAULT 0, -- Número de usuários registrados

group\_id INT REFERENCES Groups(group\_id) -- Relacionamento com o grupo da empresa

);

### **Detalhes:**

1. **Chave Primária**:  
   * company\_id foi definido como SERIAL para criar IDs únicos automaticamente.
2. **company\_cnpj**:  
   * Usei VARCHAR(18) com a constraint UNIQUE para garantir que o CNPJ não se repita.
3. **Relacionais**:  
   * **segment**: Relacionado à tabela Segment através da coluna segment\_id. Você precisará criar a tabela Segment com os valores possíveis.
   * **tax\_regime**: Relacionado à tabela TaxRegime através da coluna tax\_regime\_id. Crie a tabela TaxRegime para armazenar os regimes fiscais.
   * **group\_id**: Relaciona a empresa a um grupo, com a tabela Groups.
4. **Campos de endereço**:  
   * country e state usam CHAR para armazenar códigos padronizados (ISO Alpha-3 e Alpha-2, respectivamente).
   * number\_address foi definido como INT.
5. **Outros**:  
   * num\_registered\_users tem um valor padrão 0, indicando que inicialmente nenhuma pessoa está associada à empresa.

Se precisar de auxílio para criar as tabelas relacionadas (Segment, TaxRegime, Groups) ou ajustar algo, é só avisar!

—------------------

**tabela Groups**

CREATE TABLE Groups (

group\_id SERIAL PRIMARY KEY, -- ID único do grupo com autoincremento

plan\_id INT NOT NULL REFERENCES Plans(plan\_id), -- Relacionamento com a tabela Plans

company\_id INT NOT NULL REFERENCES Companies(company\_id) -- Relacionamento com a tabela Companies

);

### **Detalhes:**

1. **group\_id**:  
   * Definido como SERIAL para gerar IDs únicos automaticamente.
2. **plan\_id**:  
   * Faz referência à tabela Plans através de plan\_id. Certifique-se de que a tabela Plans já exista com a coluna correspondente.
3. **company\_id**:  
   * Faz referência à tabela Companies através de company\_id. Certifique-se de que a tabela Companies já exista.
4. **Relação Many-to-Many**:  
   * Essa estrutura permite que cada grupo tenha um plano associado e várias empresas inseridas no grupo. Se você precisa modelar uma relação muitos-para-muitos (várias empresas em vários grupos), o design pode precisar de uma tabela intermediária.

### **Alternativa para relação Many-to-Many:**

Se um grupo pode ter várias empresas, e uma empresa pode pertencer a vários grupos, você pode criar uma tabela intermediária como GroupCompanies:

CREATE TABLE Groups (

group\_id SERIAL PRIMARY KEY, -- ID único do grupo com autoincremento

plan\_id INT NOT NULL REFERENCES Plans(plan\_id) -- Relacionamento com a tabela Plans

);

CREATE TABLE GroupCompanies (

group\_id INT NOT NULL REFERENCES Groups(group\_id), -- Relacionamento com Groups

company\_id INT NOT NULL REFERENCES Companies(company\_id), -- Relacionamento com Companies

PRIMARY KEY (group\_id, company\_id) -- Combinação única de grupo e empresa

);

Essa abordagem adiciona flexibilidade para gerenciar grupos e suas empresas. Se precisar de mais ajustes ou explicações, me avise!

---------------------

**tabela TaxRegime**

CREATE TABLE TaxRegime (

tax\_regime\_id SERIAL PRIMARY KEY, -- ID único do regime tributário com autoincremento

tax\_regime\_name VARCHAR(50) NOT NULL, -- Nome do regime tributário (escolhido entre 5 opções)

num\_companies INT GENERATED ALWAYS AS (

(SELECT COUNT(\*) FROM Companies WHERE tax\_regime = tax\_regime\_id)

) STORED -- Número de empresas no regime, calculado automaticamente

);

### **Detalhes:**

1. **tax\_regime\_id**:  
   * Utilizei SERIAL para criar IDs únicos automaticamente.
2. **tax\_regime\_name**:  
   * Campo de texto para armazenar o nome do regime tributário.
3. **num\_companies**:  
   * Utilizei uma coluna calculada (GENERATED ALWAYS AS ... STORED) que conta automaticamente a quantidade de empresas associadas a cada regime tributário.
   * Faz a contagem com base na tabela Companies, onde tax\_regime deve ser uma coluna referenciando tax\_regime\_id.
4. **Constraint para as 5 opções**: Caso você queira limitar os nomes do regime a 5 opções predefinidas, pode criar uma tabela com os valores permitidos ou usar uma CHECK constraint, como no exemplo abaixo:

#### **Alternativa com CHECK Constraint:**

CREATE TABLE TaxRegime (

tax\_regime\_id SERIAL PRIMARY KEY, -- ID único do regime tributário com autoincremento

tax\_regime\_name VARCHAR(50) NOT NULL CHECK (

tax\_regime\_name IN ('Simples Nacional', 'Lucro Presumido', 'Lucro Real', 'MEI', 'Outro')

), -- Permite apenas 5 opções predefinidas

num\_companies INT GENERATED ALWAYS AS (

(SELECT COUNT(\*) FROM Companies WHERE tax\_regime = tax\_regime\_id)

) STORED -- Número de empresas no regime, calculado automaticamente

);

### **Observações:**

1. A coluna num\_companies depende que a tabela Companies já exista e contenha a coluna tax\_regime como chave estrangeira para TaxRegime.tax\_regime\_id.
2. O PostgreSQL suporta colunas geradas (GENERATED AS STORED), que são úteis para cálculos automáticos como contagens.

Se precisar de ajuda para configurar a tabela Companies ou outra dependência, me avise!

—-----------------------

tabela **CompanyProducts**

CREATE TABLE CompanyProducts (

product\_id SERIAL PRIMARY KEY, -- Chave primária com autoincremento

product\_type VARCHAR(20) NOT NULL CHECK (product\_type IN ('Physical Product', 'Digital Product', 'Service')), -- Verifica tipos válidos

product\_description TEXT NOT NULL, -- Texto descritivo obrigatório

product\_brand INT NOT NULL, -- Relacionado à tabela brands

product\_price NUMERIC(10, 2) NOT NULL CHECK (product\_price >= 0), -- Preço não negativo

product\_cat INT NOT NULL, -- Relacionado à tabela product\_cat

product\_status VARCHAR(10) NOT NULL CHECK (product\_status IN ('Ativo', 'Inativo')), -- Verifica status válido

product\_license VARCHAR(20) NOT NULL CHECK (

product\_license IN ('anos', 'meses', 'dias', 'horas') -- Restringe as opções de licença

),

FOREIGN KEY (product\_brand) REFERENCES brands(brand\_id) ON DELETE CASCADE, -- Relacionamento com brands

FOREIGN KEY (product\_cat) REFERENCES product\_cat(cat\_id) ON DELETE CASCADE -- Relacionamento com product\_cat

);

### **Detalhes:**

1. **Chave Primária:**
   * product\_id foi definido como SERIAL para autoincremento.
2. **Restrições de valores:**
   * product\_type aceita apenas 'Physical Product', 'Digital Product' ou 'Service'.
   * product\_status aceita apenas 'Ativo' ou 'Inativo'.
   * product\_license aceita 'anos', 'meses', 'dias', ou 'horas'.
   * product\_price não pode ser negativo.
3. **Relacionamentos:**
   * product\_brand se relaciona com a tabela brands usando a coluna brand\_id.
   * product\_cat se relaciona com a tabela product\_cat usando a coluna cat\_id.
   * Definido ON DELETE CASCADE para que exclusões em brands ou product\_cat reflitam na tabela CompanyProducts.

Se precisar de alterações ou comentários adicionais, é só avisar!

—--------------------

tabelas **CompanyProducts** e **ProductBrand**

-- Tabela ProductBrand

CREATE TABLE ProductBrand (

brand\_id SERIAL PRIMARY KEY, -- Chave primária com autoincremento

brand\_name VARCHAR(15) NOT NULL, -- Nome da marca, obrigatório

brand\_status VARCHAR(10) NOT NULL CHECK (brand\_status IN ('Ativo', 'Inativo')) -- Status entre 'Ativo' e 'Inativo'

);

-- Tabela CompanyProducts

CREATE TABLE CompanyProducts (

product\_id SERIAL PRIMARY KEY, -- Chave primária com autoincremento

product\_type VARCHAR(20) NOT NULL CHECK (product\_type IN ('Physical Product', 'Digital Product', 'Service')), -- Tipos válidos

product\_description TEXT NOT NULL, -- Descrição obrigatória

product\_brand INT NOT NULL, -- Chave estrangeira da tabela ProductBrand

product\_price NUMERIC(10, 2) NOT NULL CHECK (product\_price >= 0), -- Preço positivo

product\_cat INT NOT NULL, -- Relacionado à tabela de categorias

product\_status VARCHAR(10) NOT NULL CHECK (product\_status IN ('Ativo', 'Inativo')), -- Status válido

product\_license VARCHAR(20) NOT NULL CHECK (

product\_license IN ('anos', 'meses', 'dias', 'horas') -- Tipos de licença válidos

),

FOREIGN KEY (product\_brand) REFERENCES ProductBrand(brand\_id) ON DELETE CASCADE -- Relacionamento com ProductBrand

);

### **Detalhes:**

1. **Tabela ProductBrand:**
   * brand\_id: Chave primária, autoincremento com SERIAL.
   * brand\_name: Aceita nomes de até 15 caracteres, obrigatório.
   * brand\_status: Restringe valores a 'Ativo' e 'Inativo'.
2. **Tabela CompanyProducts:**
   * product\_id: Chave primária, autoincremento.
   * product\_type: Restringe os valores a 'Physical Product', 'Digital Product', ou 'Service'.
   * product\_brand: Chave estrangeira relacionada a ProductBrand(brand\_id), com exclusão em cascata.
   * product\_price: Limita preços a valores positivos.
   * product\_status: Restringe valores a 'Ativo' ou 'Inativo'.
   * product\_license: Limita valores a 'anos', 'meses', 'dias', ou 'horas'.

Se precisar de mais tabelas, relacionamentos ou ajustes, é só avisar!