O método `Field` do **SQLModel** é usado para configurar campos em modelos de banco de dados. Ele é uma extensão da funcionalidade do SQLAlchemy e oferece várias opções para definir metadados, configurações específicas e comportamentos dos campos.

Principais parâmetros e métodos de Field

A seguir está uma lista dos parâmetros comumente usados em `Field`:

1. **`default`**
   - Define um valor padrão para o campo.
   - Exemplo: `name: str = Field(default="Unknown")`
2. **`default_factory`**
   - Define uma função que gera o valor padrão.
   - Exemplo: `created_at: datetime = Field(default_factory=datetime.utcnow)`
3. **`primary_key`**
   - Define se o campo é uma chave primária.
   - Exemplo: `id: int = Field(primary_key=True)`
4. **`index`**
   - Cria um índice para o campo no banco de dados.
   - Exemplo: `email: str = Field(index=True)`
5. **`foreign_key`**
   - Define uma chave estrangeira para outro campo.
   - Exemplo: `team_id: int = Field(foreign_key="team.id")`
6. **`nullable`**
   - Especifica se o campo aceita valores nulos.
   - Exemplo: `middle_name: str | None = Field(nullable=True)`
7. **`sa_column`**
   - Permite definir uma instância personalizada de coluna SQLAlchemy.
   - Exemplo: `custom_field: str = Field(sa_column=Column("custom_column", String))`
8. **`sa_column_kwargs`**
   - Especifica argumentos adicionais para o objeto de coluna do SQLAlchemy.
   - Exemplo: `price: Decimal = Field(sa_column_kwargs={"type_": Numeric(10, 2)})`
9. **`max_length`**
   - Define o comprimento máximo para tipos de texto.
   - Exemplo: `username: str = Field(max_length=50)`
10. **`description`**
    - Adiciona uma descrição ao campo, útil para documentação.
    - Exemplo: `bio: str = Field(description="User biography")`
11. **`unique`**
    - Cria uma restrição de unicidade no banco de dados.
    - Exemplo: `email: str = Field(unique=True)`
12. **`constraints`**
    - Define restrições no nível do banco de dados.
    - Exemplo: `score: int = Field(constraints=[CheckConstraint("score >= 0")])`
13. **`title`**
    - Define um título para o campo (usado principalmente em esquemas Pydantic).
    - Exemplo: `username: str = Field(title="User Username")`
14. **`alias`**
    - Permite usar um nome alternativo no JSON ou em esquemas.
    - Exemplo: `full_name: str = Field(alias="fullName")`
15. **`repr`**
    - Define se o campo será incluído na representação string do modelo.
    - Exemplo: `password: str = Field(repr=False)`

16. **`gt`, `ge`, `lt`, `le`**
    - Definem restrições para validação de números (maior que, maior ou igual, etc.).
    - Exemplo: `age: int = Field(ge=18)`

    # **`gt`**: *greater than* (>) - maior que.
    # **`ge`**: *greater than or equal to* (>=) - maior ou igual a.
    # **`lt`**: *less than* (<) - menor que.
    # **`le`**: *less than or equal to* (<=) - menor ou igual a.

17. **`multiple_of`**
    - Define um múltiplo válido para números.
    - Exemplo: `discount: float = Field(multiple_of=0.01)`
18. **`regex`**
    - Valida strings com base em expressões regulares.
    - Exemplo: `phone: str = Field(regex="^\+?[0-9]{10,12}$")`