Corpo - Campos

Da mesma forma que você pode declarar validações adicionais e metadados nos parâmetros de *funções de operações de rota* com `Query`, `Path` e `Body`, você pode declarar validações e metadados dentro de modelos do Pydantic usando `Field` do Pydantic.

Importe `Field`

Primeiro, você tem que importá-lo:

```
Python hl_lines="4" {!../../../docs_src/body_fields/tutorial001.py!}
```

!!! warning "Aviso" Note que `Field` é importado diretamente do `pydantic`, não do `fastapi` como todo o resto (`Query`, `Path`, `Body`, etc).

Declare atributos do modelo

Você pode então utilizar `Field` com atributos do modelo:

```
Python hl_lines="11-14" {!../../../docs_src/body_fields/tutorial001.py!}
```

`Field` funciona da mesma forma que `Query`, `Path` e `Body`, ele possui todos os mesmos parâmetros, etc.

!!! note "Detalhes técnicos" Na realidade, `Query`, `Path` e outros que você verá em seguida, criam objetos de subclasses de uma classe `Param` comum, que é ela mesma uma subclasse da classe `FieldInfo` do Pydantic.

```
E `Field` do Pydantic retorna uma instância de `FieldInfo` também.

`Body` também retorna objetos de uma subclasse de `FieldInfo` diretamente. E tem outras que

Lembre-se que quando você importa `Query`, `Path`, e outros de `fastapi`, esse são na realic
```

!!! tip "Dica" Note como cada atributo do modelo com um tipo, valor padrão e `Field` possuem a mesma estrutura que parâmetros de *funções de operações de rota*, com `Field` ao invés de `Path`, `Query` e `Body`.

Adicione informações extras

Você pode declarar informação extra em `Field`, `Query`, `Body`, etc. E isso será incluído no JSON Schema gerado.

Você irá aprender mais sobre adicionar informações extras posteriormente nessa documentação, quando estiver aprendendo a declarar exemplos.

Recapitulando

Você pode usar `Field` do Pydantic para declarar validações extras e metadados para atributos do modelo.

Você também pode usar os argumentos de palavras-chave extras para passar metadados do JSON Schema adicionais.