INICIANDO A MODELAGEM

O comando `rails generate model` é usado no Ruby on Rails para criar um **modelo (model)** no padrão **MVC (Model-View-Controller)**. Esse comando gera um arquivo de modelo, que representa uma tabela no banco de dados, e um **arquivo de migração**, que define a estrutura da tabela no banco.

Criação com 4bytes

`rails generate model NomeDoModel atributo1:tipo atributo2:tipocria`

`criação de seguranca`

`rails generate model User name:string --primary-key-type=uuid`

O Rails cria automaticamente os seguintes arquivos:

1. 📄 **Arquivo do modelo:**
   - Criado dentro do diretório `app/models/`
   - Responsável por representar os dados da aplicação e suas regras de negócio.
   - Exemplo: `app/models/post.rb`
2. 📄 **Arquivo de migração:**
   - Criado dentro do diretório `db/migrate/`
   - `São as coisas que alteram o estado do banco.`
   - Define como a tabela será criada no banco de dados.
   - Exemplo: `db/migrate/20250327123456_create_posts.rb`
3. 🔄 **Testes de modelo (se estiver habilitado):**
   - Criados dentro do diretório `test/models/` ou `spec/models/` (para RSpec).
   - Usados para testar a lógica do modelo.

🔹 Tipos de Dados Suportados

Ao criar um modelo, podemos definir diferentes tipos de colunas. Alguns dos tipos mais comuns são:

Tipo	Descrição
`string`	Texto curto (até 255 caracteres)
`text`	Texto longo
`integer`	Número inteiro
`float`	Número decimal

Tipo	Descrição
`decimal`	Número decimal de alta precisão
`boolean`	Verdadeiro (`true`) ou falso (`false`)
`date`	Apenas data (YYYY-MM-DD)
`datetime`	Data e hora

🔹 Exemplo Prático

📌criando uma vehicles

```
→  blog git:(main) ✗ rails generate model vehicle brand:string model
:string year:integer
      invoke  active_record
      create    db/migrate/20250327114737_create_vehicles.rb
```

blog > db > migrate > 20250327114737_create_vehicles.rb

```ruby
class CreateVehicles < ActiveRecord::Migration[8.0]
  def change
    create_table :vehicles do |t|
      t.string :brand
      t.string :model
      t.integer :year

      t.timestamps
    end
  end
end
```

🔹

Observações

-id do tipo inteiro vem por padrão 4 bytes, mas por segurança é interessante se usar Um UUID (Identificador Unívoco Universal) 16bytes é um identificador único de 128 bits que pode ser gerado de forma independente e possui uma chance extremamente baixa de colisão (dois UUIDs iguais sendo gerados). Um UUID tem 128 bits de comprimento e geralmente é representado como uma string hexadecimal dividida em 5 grupos.

🔹 Por que usar UUIDs no Rails?

✅ Vantagens

1. **Único em qualquer banco de dados** → Pode ser gerado independentemente e ainda assim garantir unicidade global.

2. **Mais seguro** → Não revela quantos registros já existem, ao contrário dos IDs sequenciais (`1, 2, 3...`).
3. **Útil em sistemas distribuídos** → Diferentes servidores podem gerar IDs sem risco de colisão.
4. **Evita ataques de enumeração de ID** → Um usuário mal-intencionado não pode simplesmente modificar a URL `/users/1` para `/users/2` e acessar outro perfil.

❌ Desvantagens

1. **Mais espaço no banco** → Um UUID ocupa **16 bytes**, enquanto um inteiro ocupa **4 bytes**.
2. **Mais lento para índices** → Pesquisar por um UUID pode ser um pouco mais lento do que por um inteiro.
3. **Menos legível** → Um ID como `42` é muito mais fácil de lembrar e usar do que `c1a7d4f8-89d5-4e12-b9d7-f1d3e24a58b5`.

Gerar o model com uudy

rails generate model User name:string --primary-key-type=uuid

Aplicando a Migração

Depois de gerar o modelo, precisamos **aplicar a migração** para criar a tabela no banco de dados. Para isso, rodamos o comando: `primeiro você cria e depois aplica migrate`

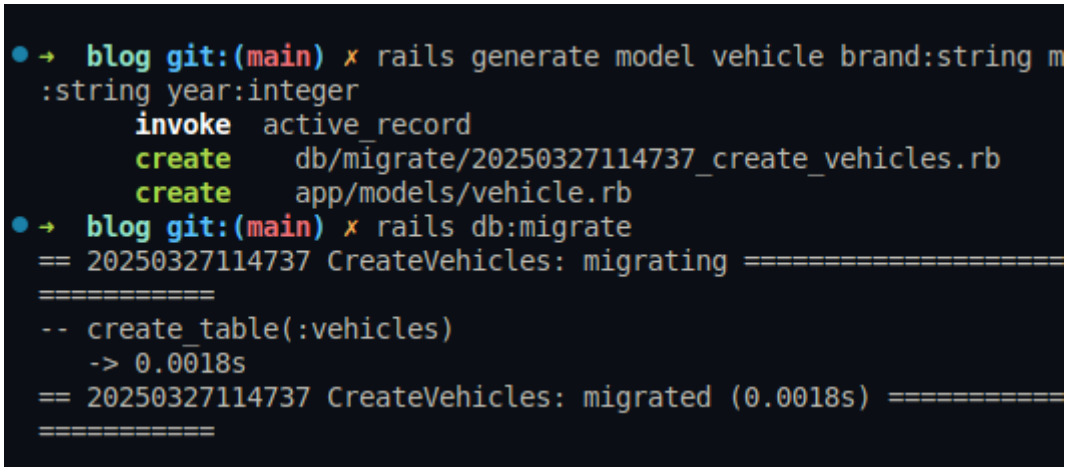
`rails db:create vai criar o banco`

`rails db:migrate →vai migrar tudo que já foi criado sobre o banco`

rails db:prepare → vai fazer o creater e vai rodar o migrate , sendo a 1 vez que roda

`rails db:drop para apagar o banco`

Esse comando executa o código da migração e cria a tabela `posts` no banco de dados.



Após isso ele irá criar um arquivo chamado schema que não deve ser modificado, gerado automaticamente.
O arquivo `db/schema.rb` no Ruby on Rails é uma **representação da estrutura do banco de dados** em formato Ruby. Ele é gerado automaticamente pelo Rails sempre que você executa uma **migração (`rails db:migrate`)** e serve como uma referência para o estado atual do banco de dados.

🔹 Pontos a observar

-Esqueci de colocar um item na após a criação da model, deve se analisar em que momento da utilização da model está:

1.Criou o documento e depois de um tempo descobre que tem que adicioanr algo a uma tabela já criada ,criar outro arquivo de migração.

2. Esta em momento de desenvolvimento, antes de subir para produção você ve que faltou alguma coisa vai ter que alterar no momento no mesmo arquivo

o comando

```
blog git:(main) ✗ rails g migration AddColumnKindToVehicle kind:integer
```

Adiciona coluna -nome da coluna -para → add column king to vehicle – criando outra migrate na pasta com adição.

```
blog git:(main) ✗ rails g migration AddColumnKindToVehicle kind:integer
      invoke  active_record
      create    db/migrate/20250327123357_add_column_kind_to_vehicle.rb
blog git:(main) ✗
```

blog > db > migrate > 20250327123357_add_column_kind_to_vehicle.rb

```ruby
class AddColumnKindToVehicle < ActiveRecord::Migrati
  def change
    add_column :vehicles, :kind, :integer
  end
end
```

OBSERVAÇÃO:Se rodar ele não apagara a migrate anterior, assim fica presente na primeira migrate o timestamps que registra que aquela migrate já está no banco, assim ele passa para nova que ainda não está executada. Assim você olha para o schema e a modificação não estará la ainda , entçao roda uma nova migrate → rails db:migrate ele adiciona o movo alterando o estado atual do banco no schema

-CASO TENHA ERRADO UMA ULTIMA ADIÇÃO DAR UM rails db:rollback

```ruby
ActiveRecord::Schema[8.0].define(version: 2025_03_27
  create_table "vehicles", force: :cascade do |t|
    t.string "brand"
    t.string "model"
    t.integer "year"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.integer "kind"
  end
```

-Lembrar de observar que o banco deve esta de pé para fazer migrate

- consultar o status do banco e suas modificações rails db:migrate:status

```
→  blog git:(main) ✗ rails db:migrate:status

database: storage/development.sqlite3

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20250327114737  Create vehicles
   up     20250327123357  Add column kind to vehicle
  down    20250327124801  Add column plate to vehicle
```

caso eu deseje voltar ao rollback para uma versão especifica PERIGOSO

rails db:migrate:down VERSION=

```
down    20250327124801  Add column plate to vehicle

→  blog git:(main) ✗ rails db:migrate:down VERSION=
→  blog git:(main) ✗ rails db:migrate:down VERSION=2025032712
== 20250327123357 AddColumnKindToVehicle: reverting ==========
-- remove_column(:vehicles, :kind, :integer)
   -> 0.0224s
== 20250327123357 AddColumnKindToVehicle: reverted (0.0233s)
==========
```