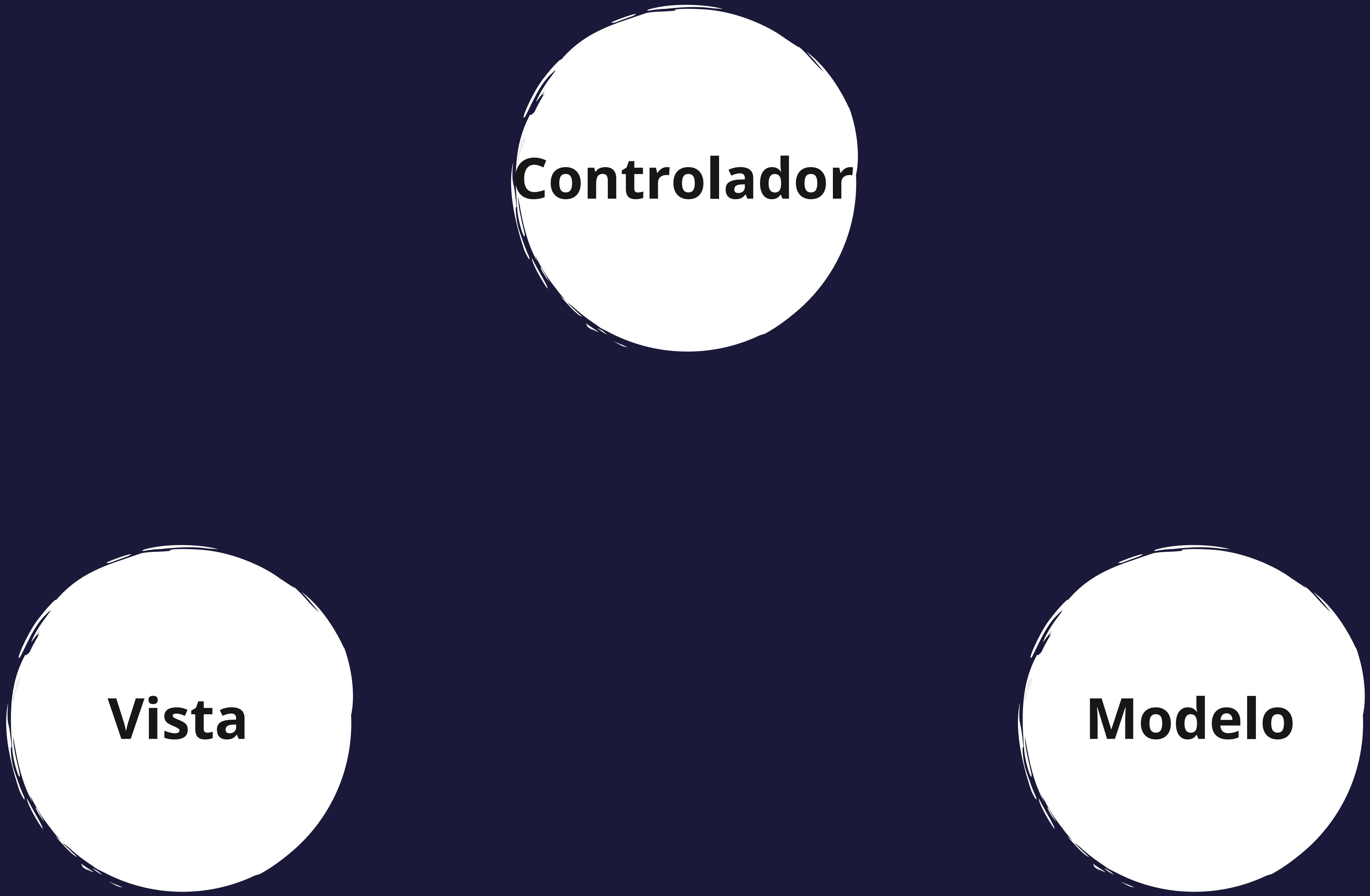
001

MIGRACIONES & ASOCIACIONES

INGENIERÍA DE SOFTWARE

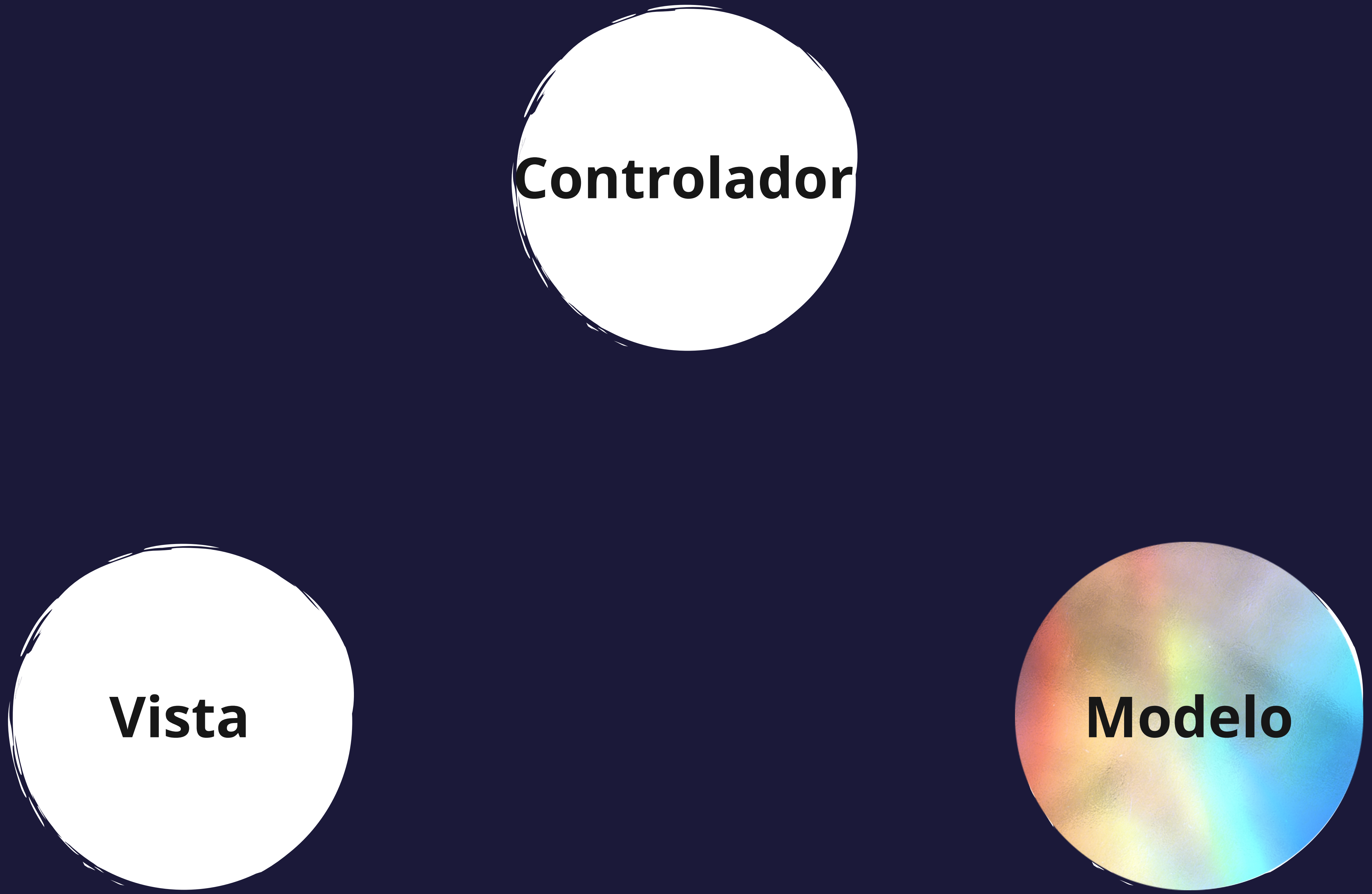
Miguel Ortiz
Felipe Fuentes
Sebastian Breguel



Controlador

Vista

Modelo

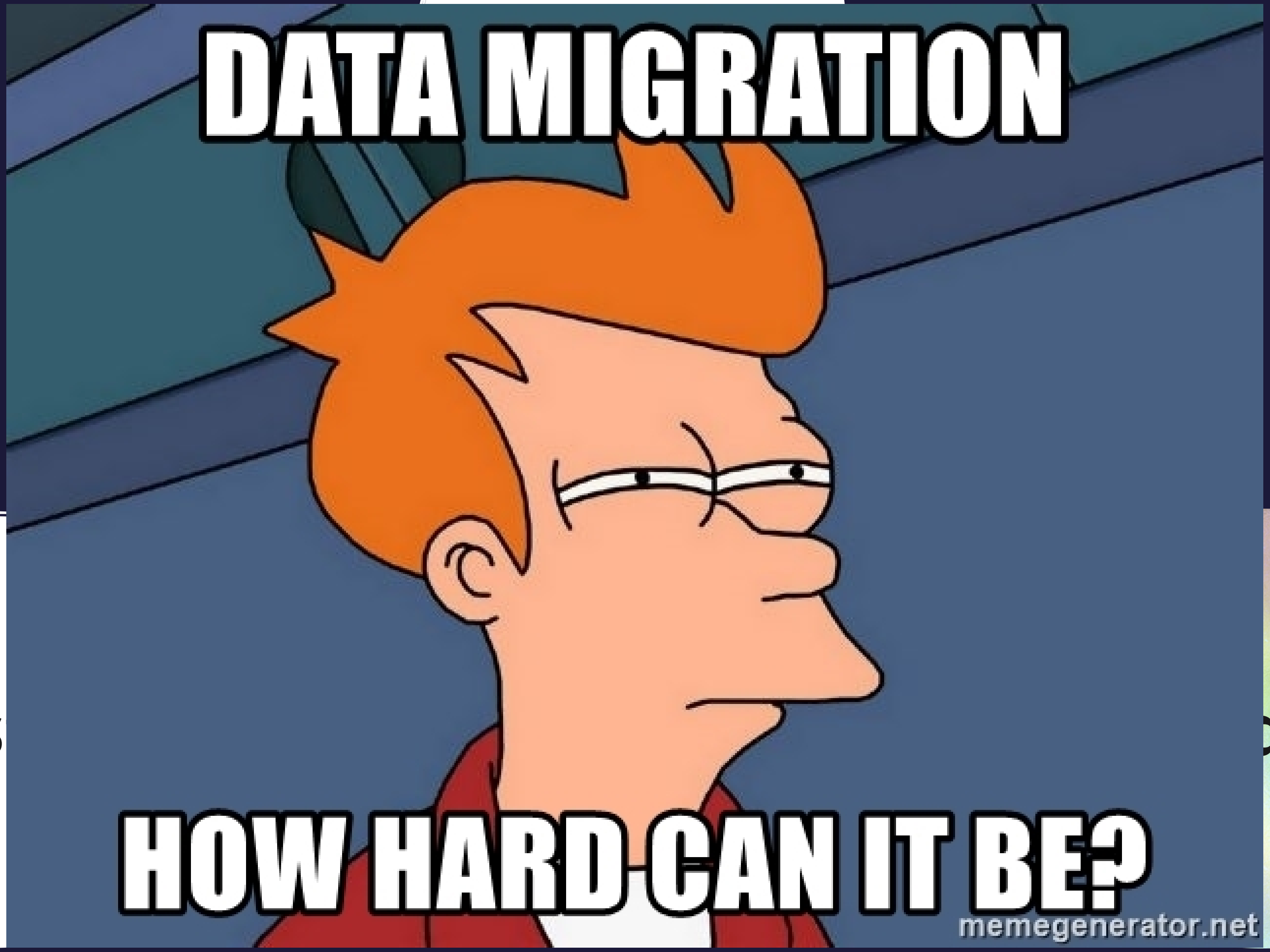


Controlador

Vista

Modelo

Vis



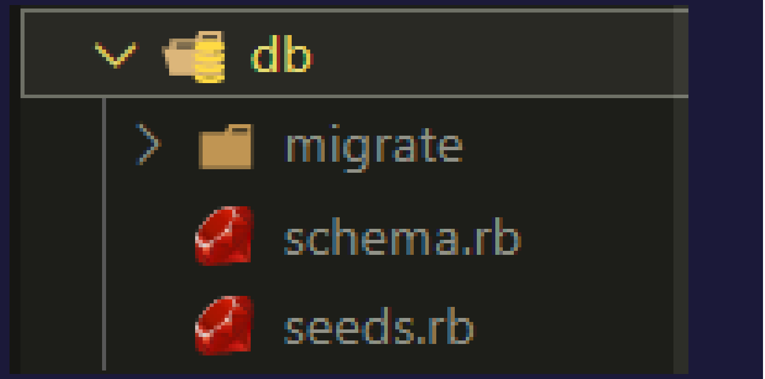
delo

MIGRACIONES ¿QUÉ SON?

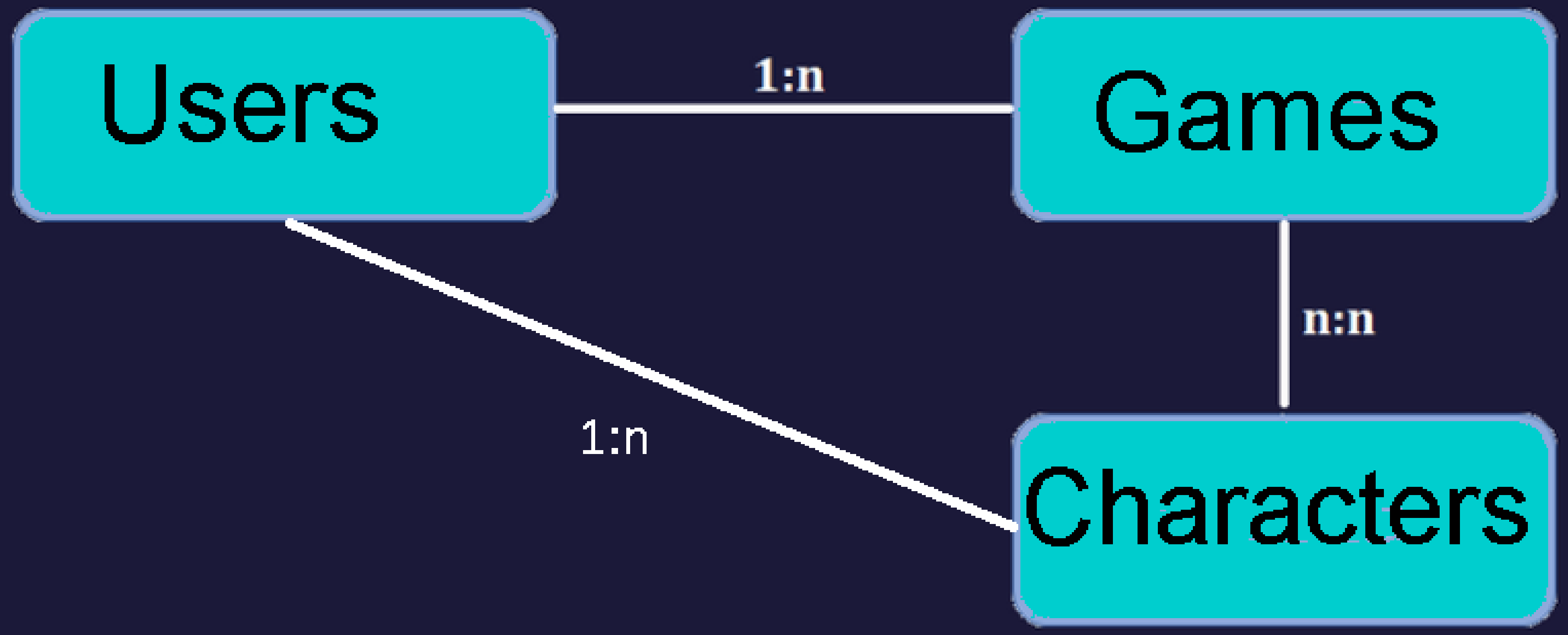
- CÓDIGO QUE PERMITE CREAR Y ALTERAR LA BASE DE DATOS.

- COMANDOS QUE EVITAN OPERAR CON COMANDOS SQL.

- CREAN ARCHIVOS DE MIGRACIÓN EN "DB/MIGRATE".



```
db
  migrate
  schema.rb
  seeds.rb
```



Users

1:n

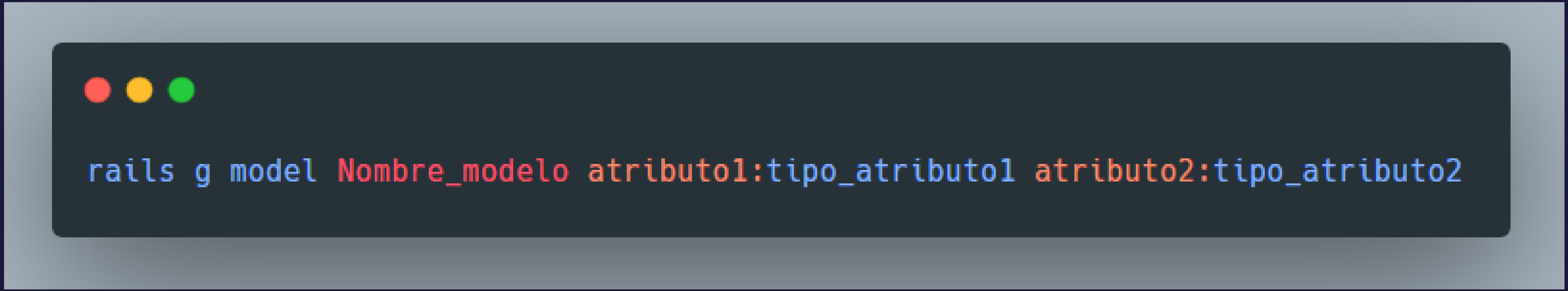
Games

n:n

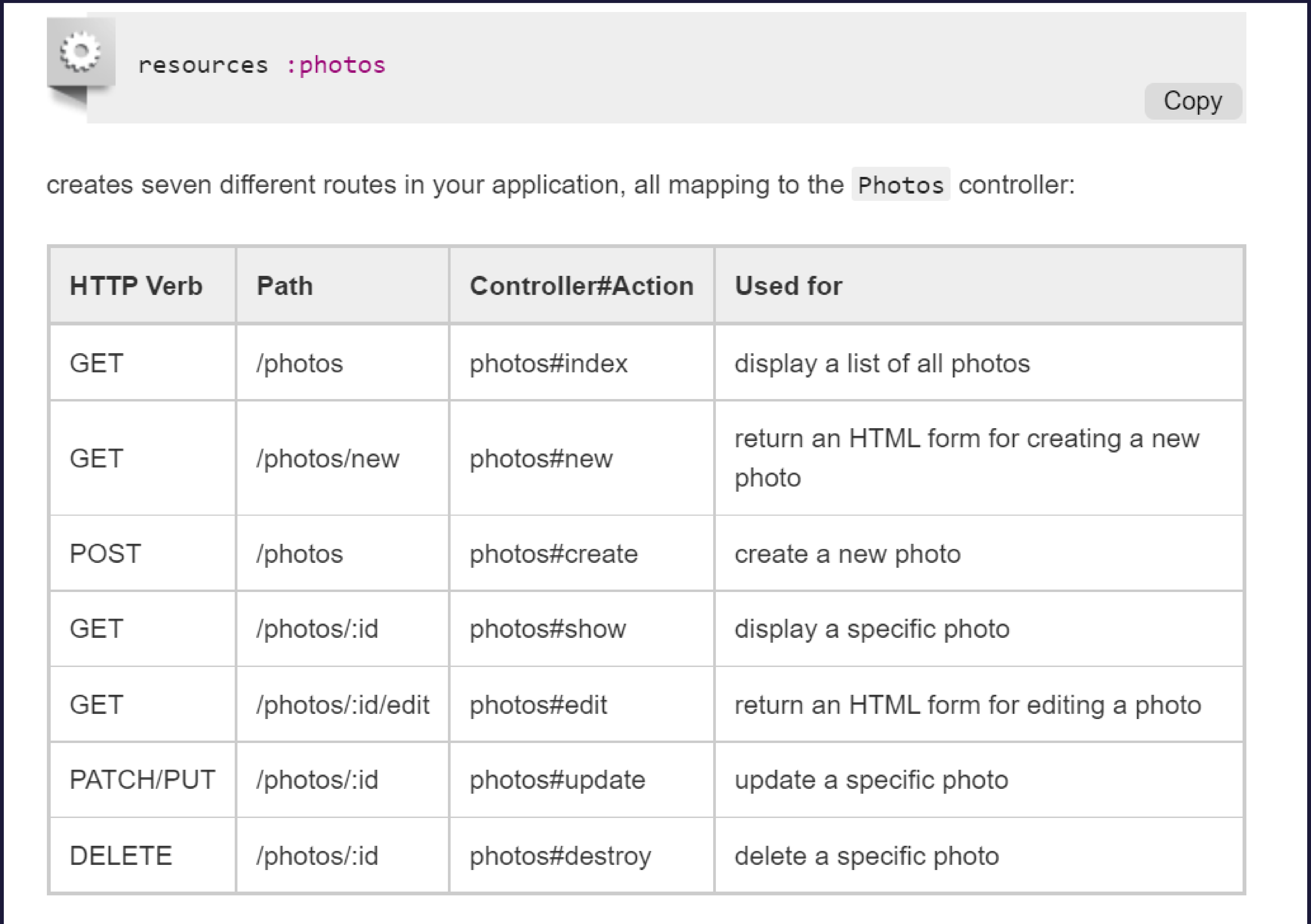
1:n

Characters

Crear un modelo



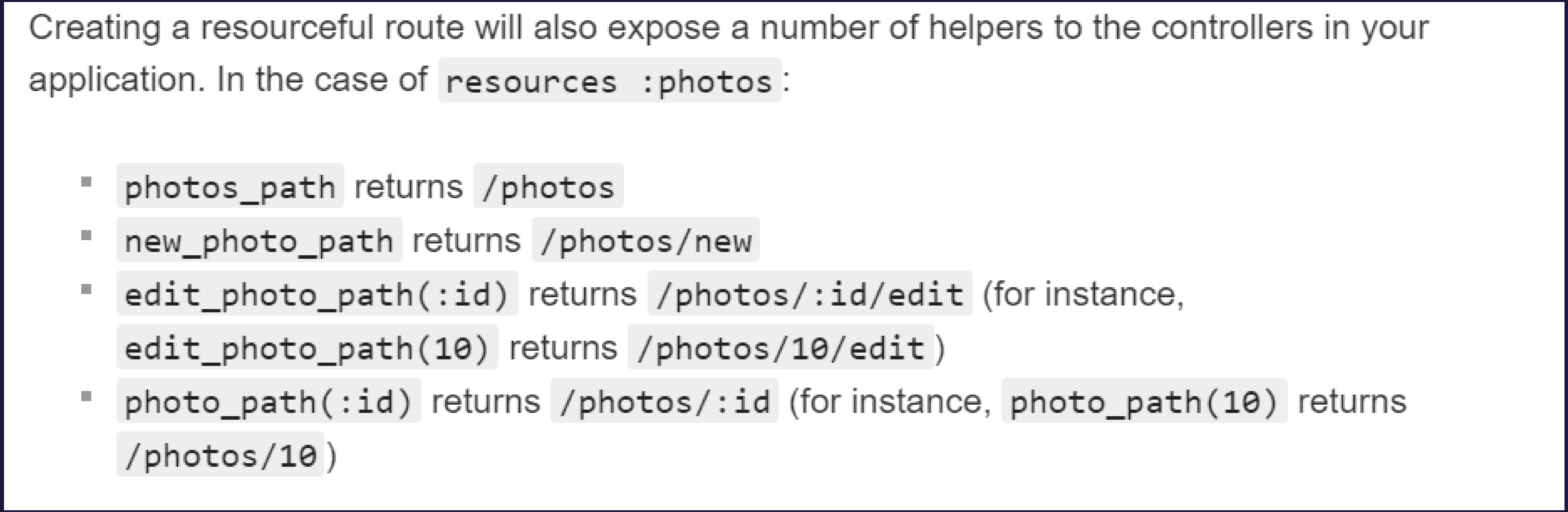
```
rails g model Nombre_modelo atributo1:tipo_atributo1 atributo2:tipo_atributo2
```



```
resources :photos
```

creates seven different routes in your application, all mapping to the `Photos` controller:

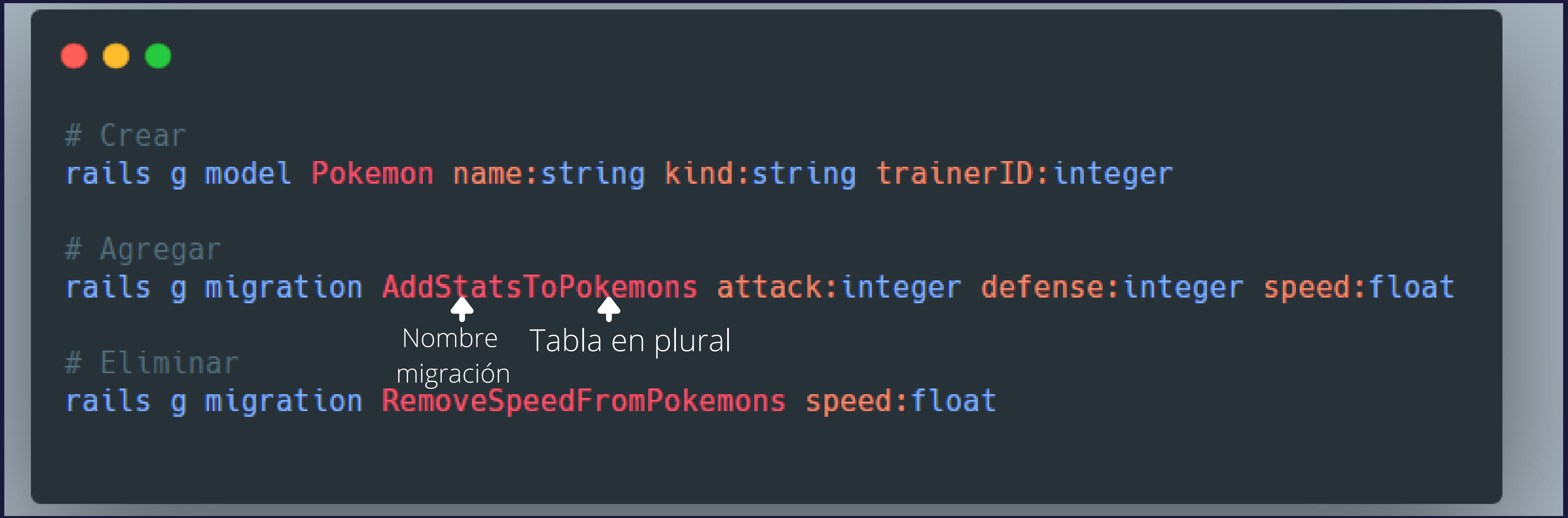
HTTP Verb	Path	Controller#Action	Used for
GET	/photos	photos#index	display a list of all photos
GET	/photos/new	photos#new	return an HTML form for creating a new photo
POST	/photos	photos#create	create a new photo
GET	/photos/:id	photos#show	display a specific photo
GET	/photos/:id/edit	photos#edit	return an HTML form for editing a photo
PATCH/PUT	/photos/:id	photos#update	update a specific photo
DELETE	/photos/:id	photos#destroy	delete a specific photo



Creating a resourceful route will also expose a number of helpers to the controllers in your application. In the case of `resources :photos`:

- `photos_path` returns `/photos`
- `new_photo_path` returns `/photos/new`
- `edit_photo_path(:id)` returns `/photos/:id/edit` (for instance, `edit_photo_path(10)` returns `/photos/10/edit`)
- `photo_path(:id)` returns `/photos/:id` (for instance, `photo_path(10)` returns `/photos/10`)

Tipos de migración



```
# Crear
rails g model Pokemon name:string kind:string trainerID:integer

# Agregar
rails g migration AddStatsToPokemons attack:integer defense:integer speed:float

# Eliminar
rails g migration RemoveSpeedFromPokemons speed:float
```

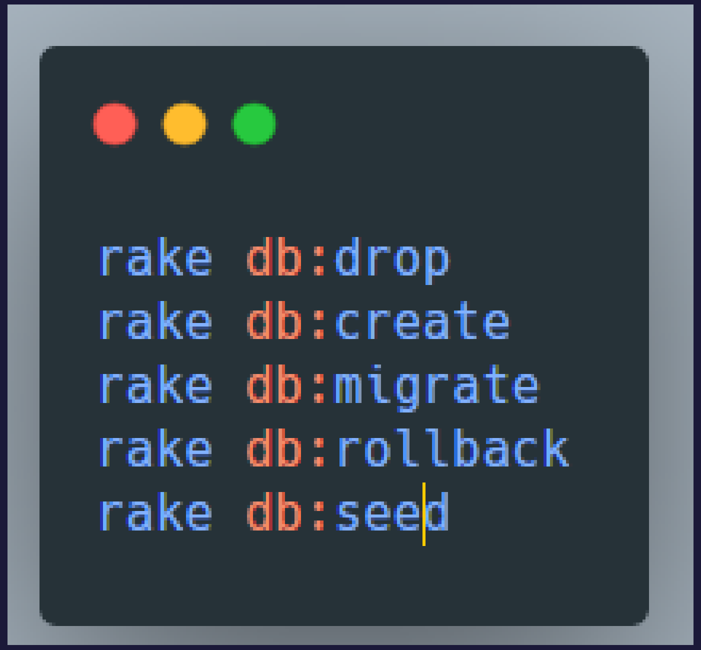
Nombre migración

Tabla en plural

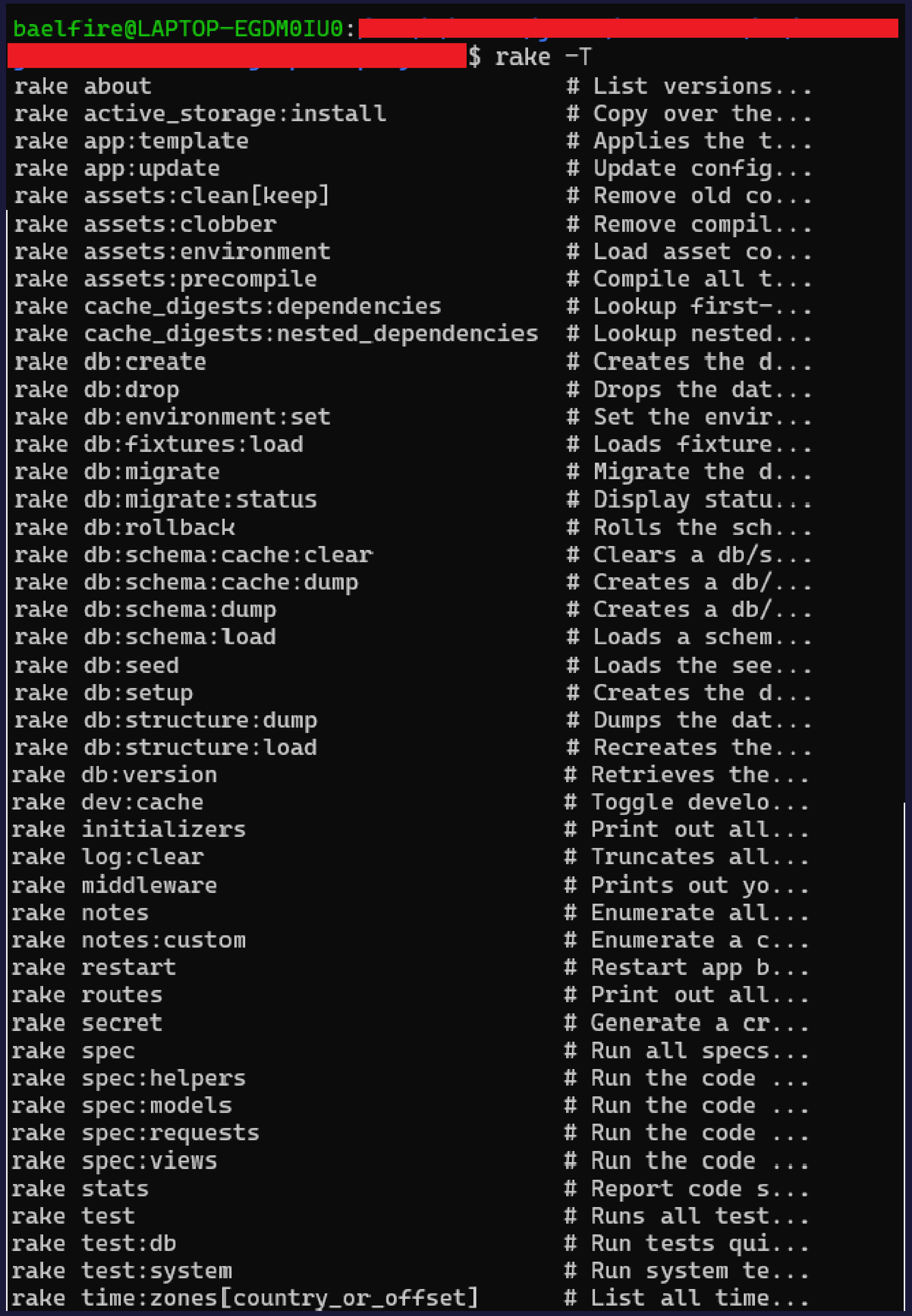
¿Qué es Rake?

ES UN CONSTRUCTOR DE RAIL QUE PERMITE MANJEAR EL MODELO

Desde Rails 5 en adelante se permite llamar a 'rails' y que este incluya 'rake'

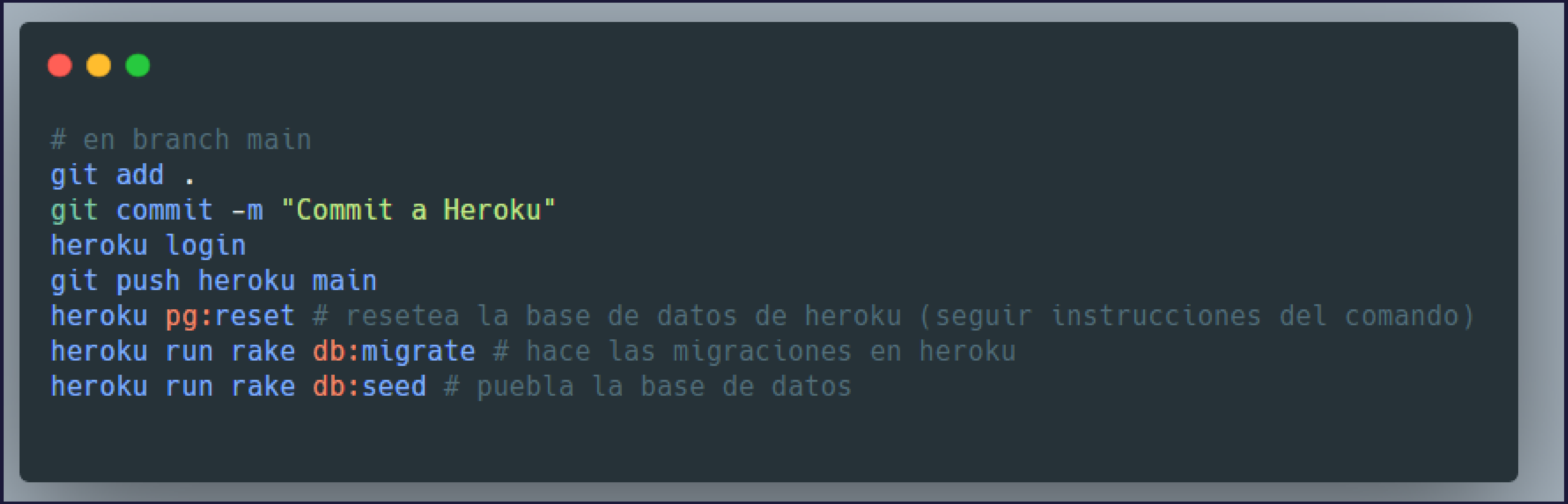


```
rake db:drop
rake db:create
rake db:migrate
rake db:rollback
rake db:seed
```



```
baelfire@LAPTOP-EGDM0IU0:[redacted]
$ rake -T
rake about                                 # List versions...
rake active_storage:install                # Copy over the...
rake app:template                          # Applies the t...
rake app:update                            # Update config...
rake assets:clean[keep]                    # Remove old co...
rake assets:clobber                        # Remove compil...
rake assets:environment                    # Load asset co...
rake assets:precompile                     # Compile all t...
rake cache_digests:dependencies            # Lookup first-...
rake cache_digests:nested_dependencies     # Lookup nested...
rake db:create                             # Creates the d...
rake db:drop                               # Drops the dat...
rake db:environment:set                    # Set the envir...
rake db:fixtures:load                      # Loads fixture...
rake db:migrate                            # Migrate the d...
rake db:migrate:status                     # Display statu...
rake db:rollback                           # Rolls the sch...
rake db:schema:cache:clear                 # Clears a db/s...
rake db:schema:cache:dump                  # Creates a db/...
rake db:schema:dump                        # Creates a db/...
rake db:schema:load                        # Loads a schem...
rake db:seed                               # Loads the see...
rake db:setup                              # Creates the d...
rake db:structure:dump                     # Dumps the dat...
rake db:structure:load                     # Recreates the...
rake db:version                            # Retrieves the...
rake dev:cache                             # Toggle develo...
rake initializers                          # Print out all...
rake log:clear                             # Truncates all...
rake middleware                            # Prints out yo...
rake notes                                 # Enumerate all...
rake notes:custom                          # Enumerate a c...
rake restart                               # Restart app b...
rake routes                                # Print out all...
rake secret                                # Generate a cr...
rake spec                                  # Run all specs...
rake spec:helpers                          # Run the code ...
rake spec:models                           # Run the code ...
rake spec:requests                         # Run the code ...
rake spec:views                            # Run the code ...
rake stats                                 # Report code s...
rake test                                  # Runs all test...
rake test:db                               # Run tests qui...
rake test:system                           # Run system te...
rake time:zones[country_or_offset]         # List all time...
```

Seeds en Heroku

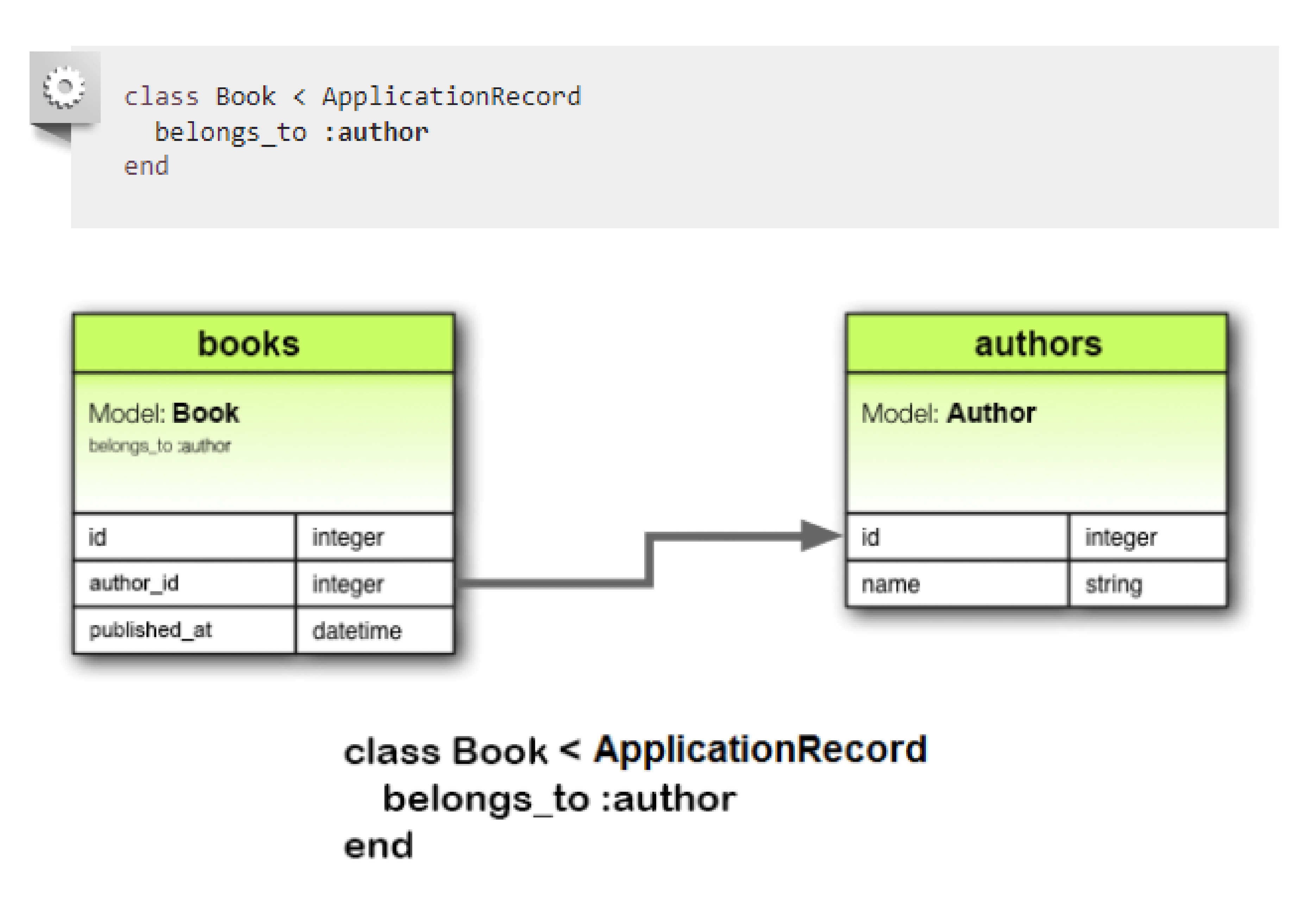


```
# en branch main
git add .
git commit -m "Commit a Heroku"
heroku login
git push heroku main
heroku pg:reset # resetea la base de datos de heroku (seguir instrucciones del comando)
heroku run rake db:migrate # hace las migraciones en heroku
heroku run rake db:seed # puebla la base de datos
```

ASOCIACIONES ¿QUÉ SON?

- ES LA FORMA POR LA QUE HACEMOS LAS RELACIONES ENTRE LOS MODELOS DE ACTIVE RECORDS

belongs_to



```
class Book < ApplicationRecord
  belongs_to :author
end
```

books

Model: **Book**

belongs_to :author

id	integer
author_id	integer
published_at	datetime

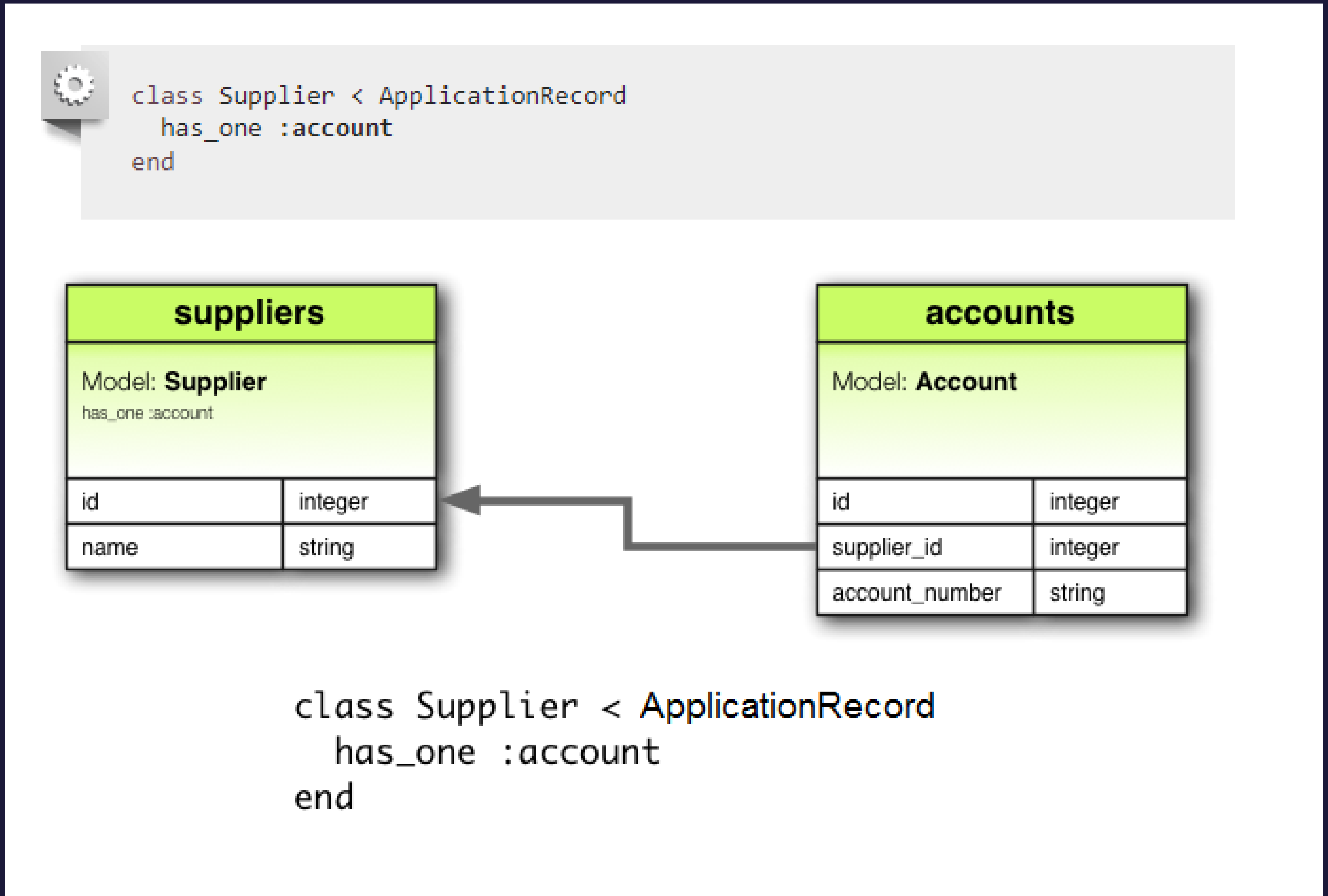
authors

Model: **Author**

id	integer
name	string

```
class Book < ApplicationRecord
  belongs_to :author
end
```

has_one



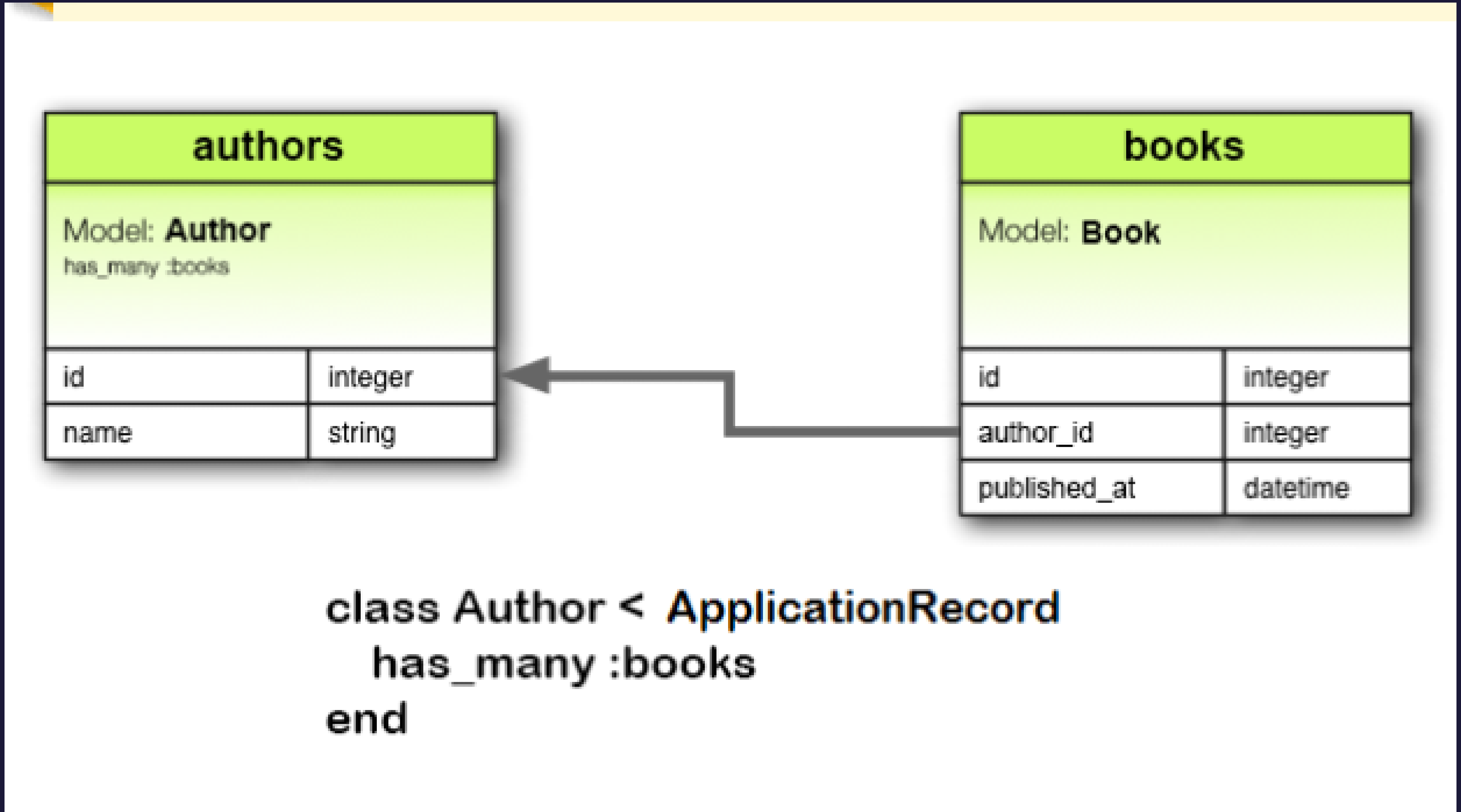
```ruby
class Supplier < ApplicationRecord
  has_one :account
end
```

suppliers	
Model: **Supplier** has_one :account	
id	integer
name	string

accounts	
Model: **Account**	
id	integer
supplier_id	integer
account_number	string

```ruby
class Supplier < ApplicationRecord
  has_one :account
end
```

has_many



authors

Model: **Author**
has_many :books

id	integer
name	string

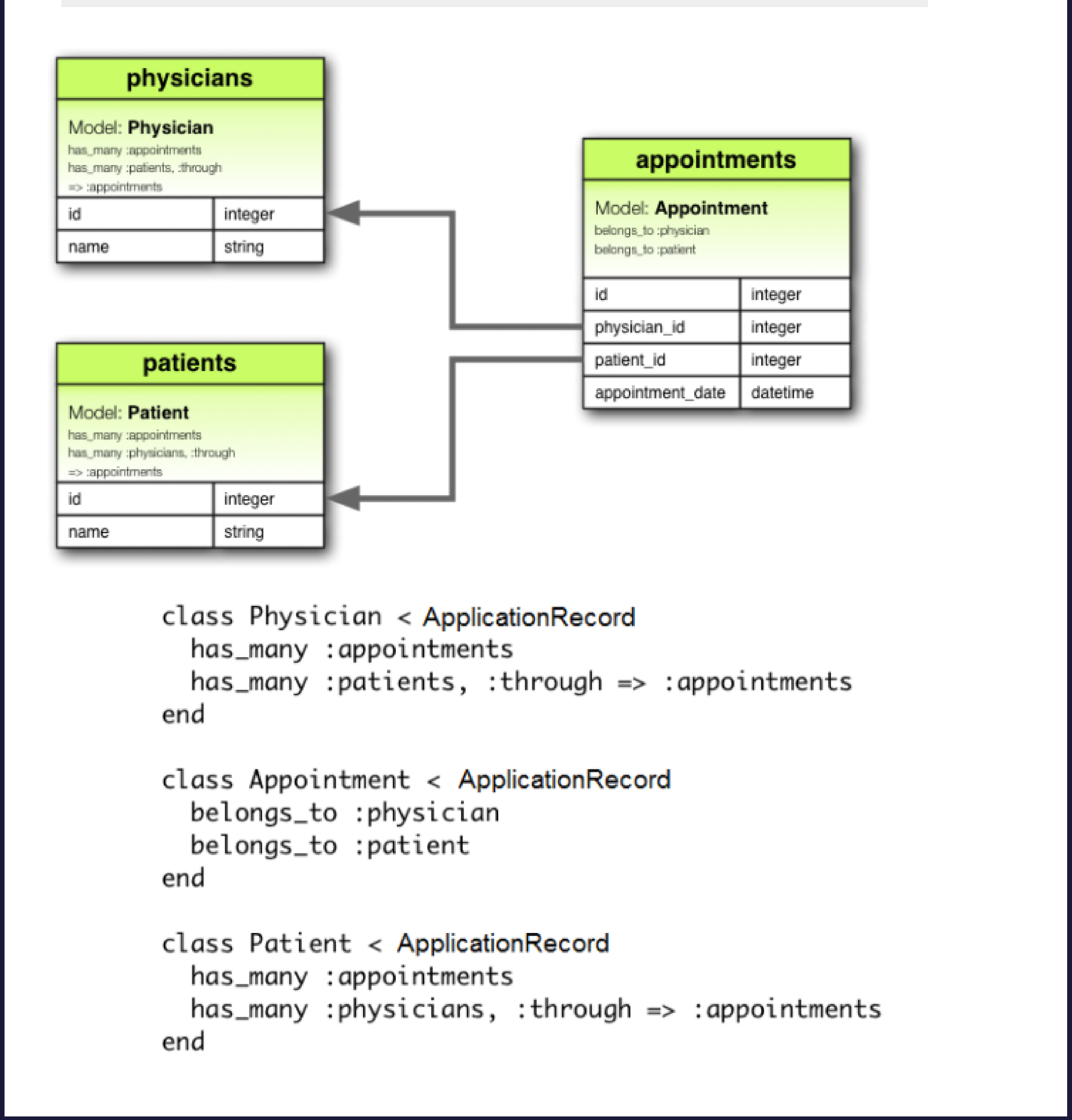
books

Model: **Book**

id	integer
author_id	integer
published_at	datetime

```
class Author <  ApplicationRecord
  has_many :books
end
```

has_many: through



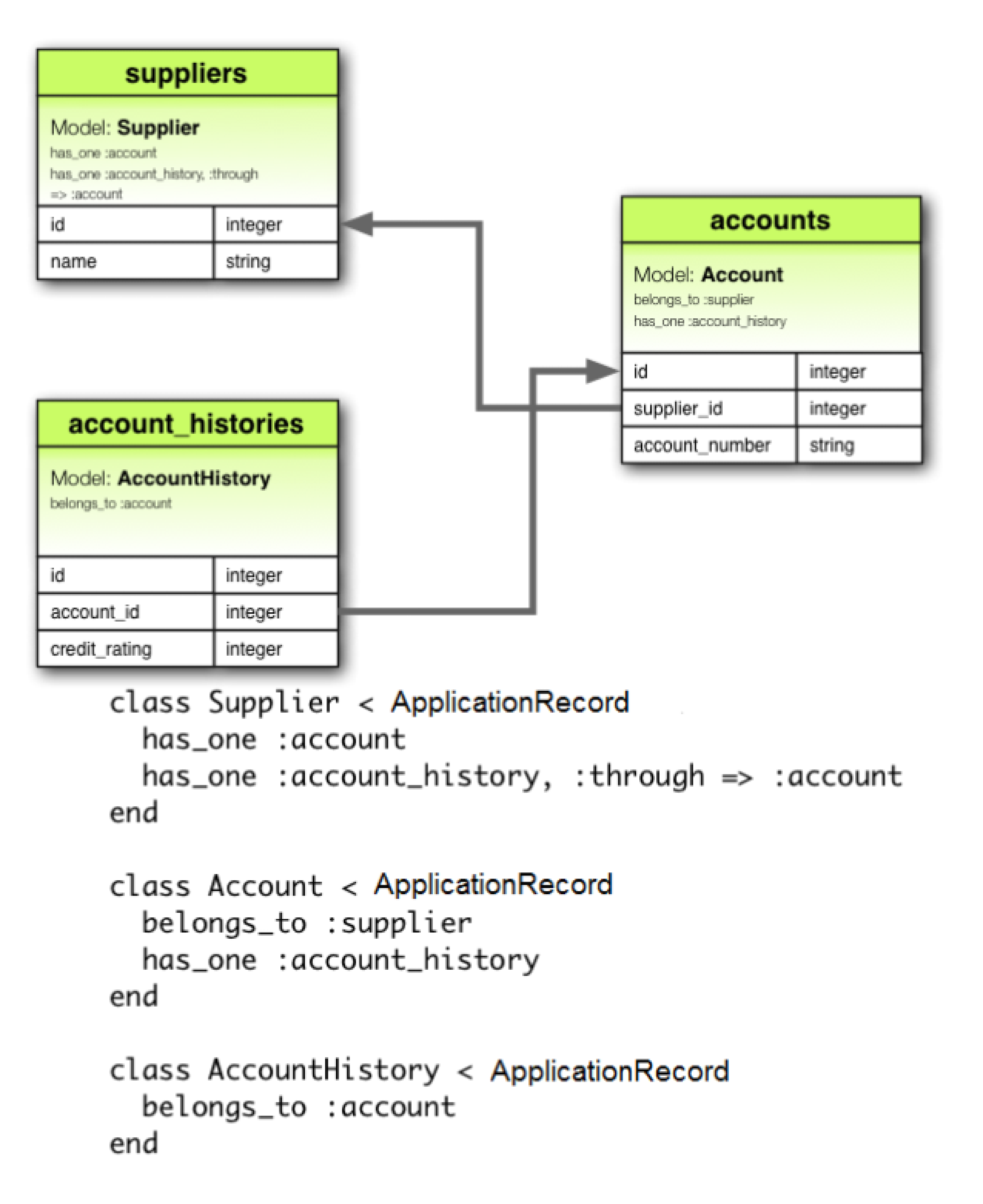
id	integer
name	string

id	integer
name	string

id	integer
physician_id	integer
patient_id	integer
appointment_date	datetime

```
class Physician < ApplicationRecord
  has_many :appointments
  has_many :patients, :through => :appointments
end

class Appointment < ApplicationRecord
  belongs_to :physician
  belongs_to :patient
end

class Patient < ApplicationRecord
  has_many :appointments
  has_many :physicians, :through => :appointments
end
```

has_one: through



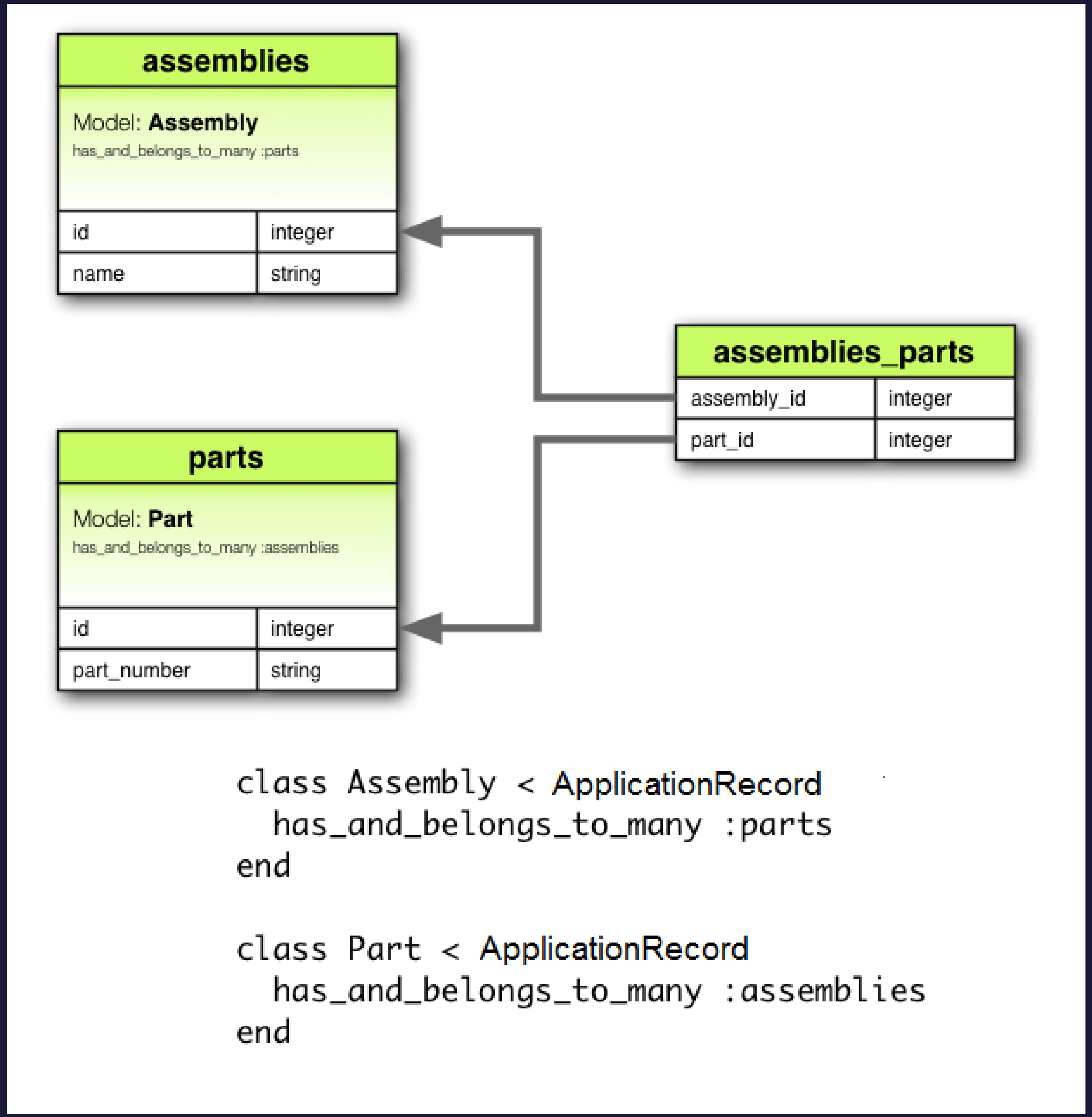
suppliers	
Model: **Supplier** has_one :account has_one :account_history, :through => :account	
id	integer
name	string

accounts	
Model: **Account** belongs_to :supplier has_one :account_history	
id	integer
supplier_id	integer
account_number	string

account_histories	
Model: **AccountHistory** belongs_to :account	
id	integer
account_id	integer
credit_rating	integer

```
class Supplier < ApplicationRecord
  has_one :account
  has_one :account_history, :through => :account
end

class Account < ApplicationRecord
  belongs_to :supplier
  has_one :account_history
end

class AccountHistory < ApplicationRecord
  belongs_to :account
end
```

has_and_belongs_to_many



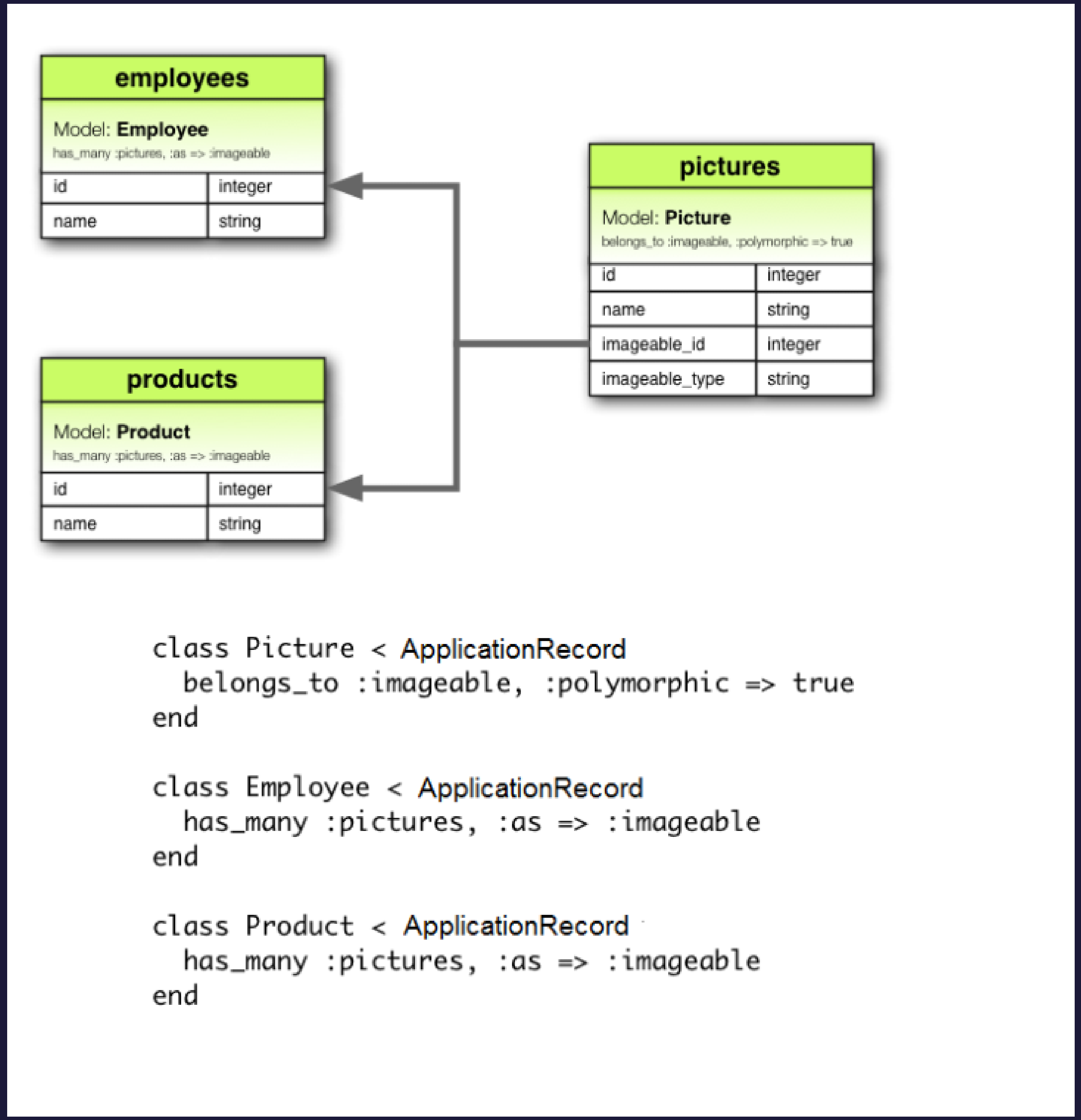
assemblies	
Model: **Assembly** has_and_belongs_to_many :parts	
id	integer
name	string

parts	
Model: **Part** has_and_belongs_to_many :assemblies	
id	integer
part_number	string

assemblies_parts	
assembly_id	integer
part_id	integer

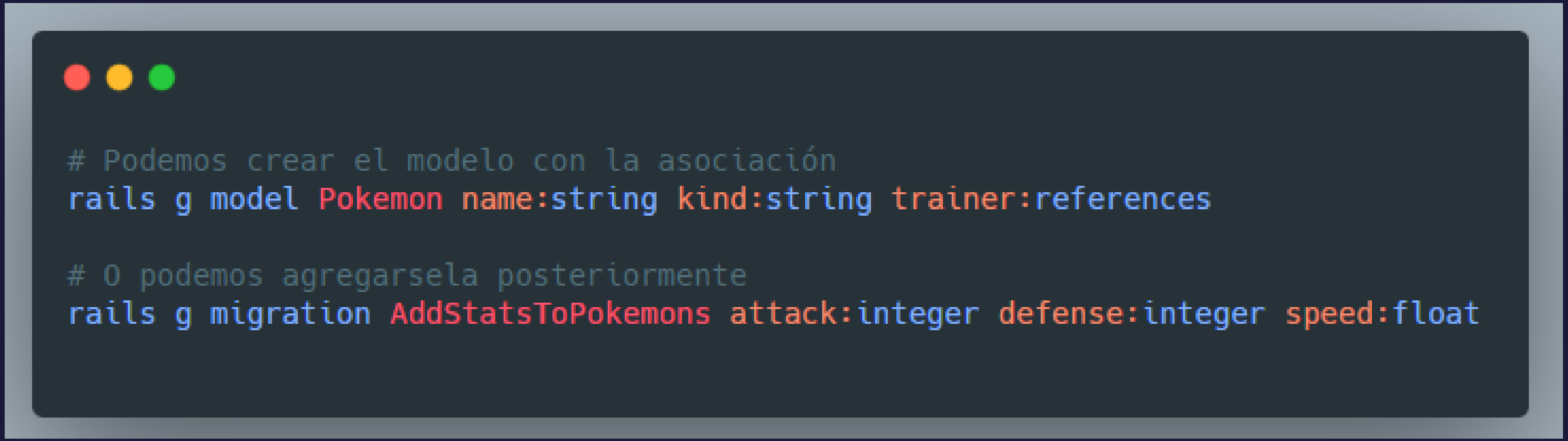
```
class Assembly < ApplicationRecord
  has_and_belongs_to_many :parts
end

class Part < ApplicationRecord
  has_and_belongs_to_many :assemblies
end
```

polymorphic

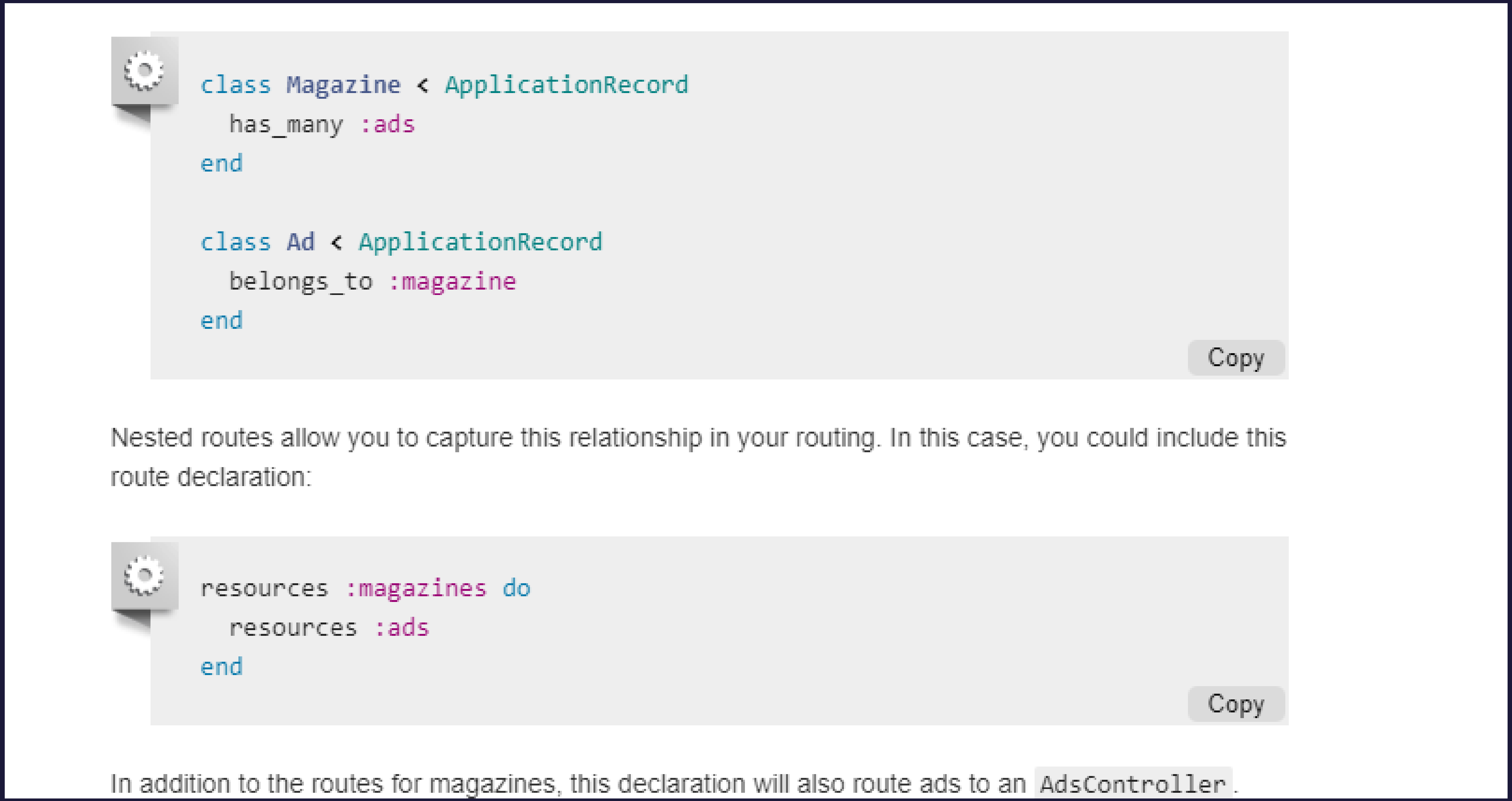


```
class Picture < ApplicationRecord
  belongs_to :imageable, :polymorphic => true
end

class Employee < ApplicationRecord
  has_many :pictures, :as => :imageable
end

class Product < ApplicationRecord
  has_many :pictures, :as => :imageable
end
```

¡Importante! al hacer asociaciones se debe crear la columna asociada a la key (id) del modelo al cual se le quiere hacer la asociación. (al usar belongs_to)



```
# Podemos crear el modelo con la asociación
rails g model Pokemon name:string kind:string trainer:references

# O podemos agregarsela posteriormente
rails g migration AddStatsToPokemons attack:integer defense:integer speed:float
```



```
class Magazine < ApplicationRecord
  has_many :ads
end

class Ad < ApplicationRecord
  belongs_to :magazine
end
```

Nested routes allow you to capture this relationship in your routing. In this case, you could include this route declaration:

```
resources :magazines do
  resources :ads
end
```

In addition to the routes for magazines, this declaration will also route ads to an `AdsController`.

Cuando entiendes migraciones y asociaciones

Active Record Migrations

Active Record MigrationsMigrations are a feature of Active Record that allows you to evolve your database schema over time. Rather than write schema modifications in pure SQL, migrations allow you to use a Ruby DSL to describe changes to your tables.After reading this guide, you will know: The generators you can use to create them.

Ruby on Rails Guides

Active Record Associations

Active Record AssociationsThis guide covers the association features of Active Record.After reading this guide, you will know: How to declare associations between Active Record models. How to understand the various types of Active Record associations.

Ruby on Rails Guides