IIC 2143 - Ingeniería de Software

Ruby on Rails

M. Trinidad Vargas
mtvargas1@uc.cl

¿Qué es Ruby on Rails?

Es un framework de desarrollo de **aplicaciones web** escrito en ruby

Aplicación web es un programa o software que se ejecuta en un servidor y al que se accede a través de un navegador web.

Protocolo HTTP (Hypertext Transfer Protocol): es un protocolo de comunicación utilizado para la transferencia de información en la web.

Crear una API

Creamos una aplicación, se instalan las dependencias de gemas definidas en Gemfile usando bundle install

```
> rails new my_recipes --database=postgresql --api
```

```
> cd my_recipes
```

Para más opciones:

```
> rails new --help
```

Crear el modelo Recipe

Generamos el modelo Recipe con atributos name y duration. Se generan los archivos del modelo y de la migración

```
> rails generate model Recipe name:text duration:integer
```

Crear el modelo Recipe

app/models/recipe.rb

```ruby
class Recipe < ApplicationRecord
end
```

Crear el modelo Recipe

db/migrate/[fecha]_create_recipes.rb

```ruby
class CreateRecipes < ActiveRecord::Migration[7.0]
 def change
   create_table :recipes do |t|
     t.text :name
     t.integer :duration

     t.timestamps
   end
 end
end
```

Migraciones

Las migraciones son instrucciones que permiten evolucionar el esquema de base de datos con el tiempo de manera reproducible.

Cada migración se puede pensar como una versión del esquema en la base de datos

Correr las migraciones

```
> rails db:migrate
```

```
== 20250312042814 CreateRecipes: migrating ====================================
-- create_table(:recipes)
   -> 0.0096s
== 20250312042814 CreateRecipes: migrated (0.0096s) ===========================
```

```
> rails db:rollback
```

Crear instancias de Recipe

Abrimos la consola de Rails

```
rails console

> recipe1 = Recipe.new(name: "Pastel de Choclo", duration: 50)

> recipe1.save
```

Recuperar datos de la base de datos

```
> Recipe.all

> Recipe.column_names
```

```
3.3.6 :005 > Recipe.all
  Recipe Load (0.5ms)  SELECT "recipes".* FROM "recipes"
 =>
[#<Recipe:0x0000000122a744c8
  id: 1,
  name: "Pastel de Choclo",
  duration: 50,
  created_at: Wed, 12 Mar 2025 04:40:02.557836000 UTC +00:00,
  updated_at: Wed, 12 Mar 2025 04:40:02.557836000 UTC +00:00>,
 #<Recipe:0x0000000122a74388
  id: 2,
  name: "Empanadas de Queso",
  duration: 120,
  created_at: Wed, 12 Mar 2025 04:42:47.978965000 UTC +00:00,
  updated_at: Wed, 12 Mar 2025 04:42:47.978965000 UTC +00:00>]
```

API REST

Api (Application Programming Interface): Es un conjunto de reglas y protocolos que permiten que las aplicaciones de software se comuniquen entre sí.

Arquitectura REST: es un estilo arquitectónico para crear servicios web. Las APIs REST utilizan métodos HTTP (GET, POST, PUT, DELETE) y suelen manejar datos en formato JSON. Se basan en recursos (como usuarios, productos, artículos) que se identifican por URLs, y las acciones sobre estos recursos se realizan utilizando los métodos HTTP.

Operaciones CRUD

CRUD es un acrónimo de Create, Read, Update, Delete

Son las operaciones básicas que se pueden realizar sobre información almacenada en base de datos

- Crear una receta
- Ver una o todas las recetas
- Actualizar un paso de la receta
- Eliminar los datos

Definir las rutas

Las rutas se definen en el archivo config/routes.rb

```
Rails.application.routes.draw do
 get "/recipes", to: "recipes#index"
 get "/recipes/:id", to: "recipes#show"
 post "/recipes", to: "recipes#create"
 patch "/recipes/:id", to: "recipes#update"
 delete "/recipes/:id", to: "recipes#delete"
end
```

Definir las rutas

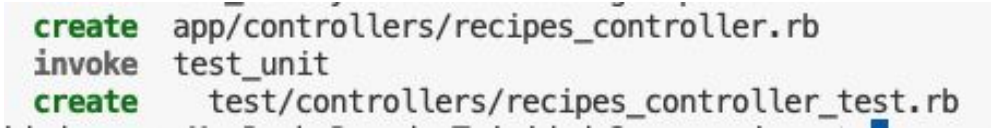
Inspeccionar las rutas definidas y el controlador - acción asociados

```
> rails routes
```

Crear el controlador

Se crea el archivo app/controllers/recipes_controller.rb

```
> rails generate controller Recipes
```



Iniciar el servidor

```
> rails s
```

Crear, leer, actualizar, eliminar estudiantes

En el archivo app/controllers/recipes_controller.rb

creamos los métodos en la clase RecipesController

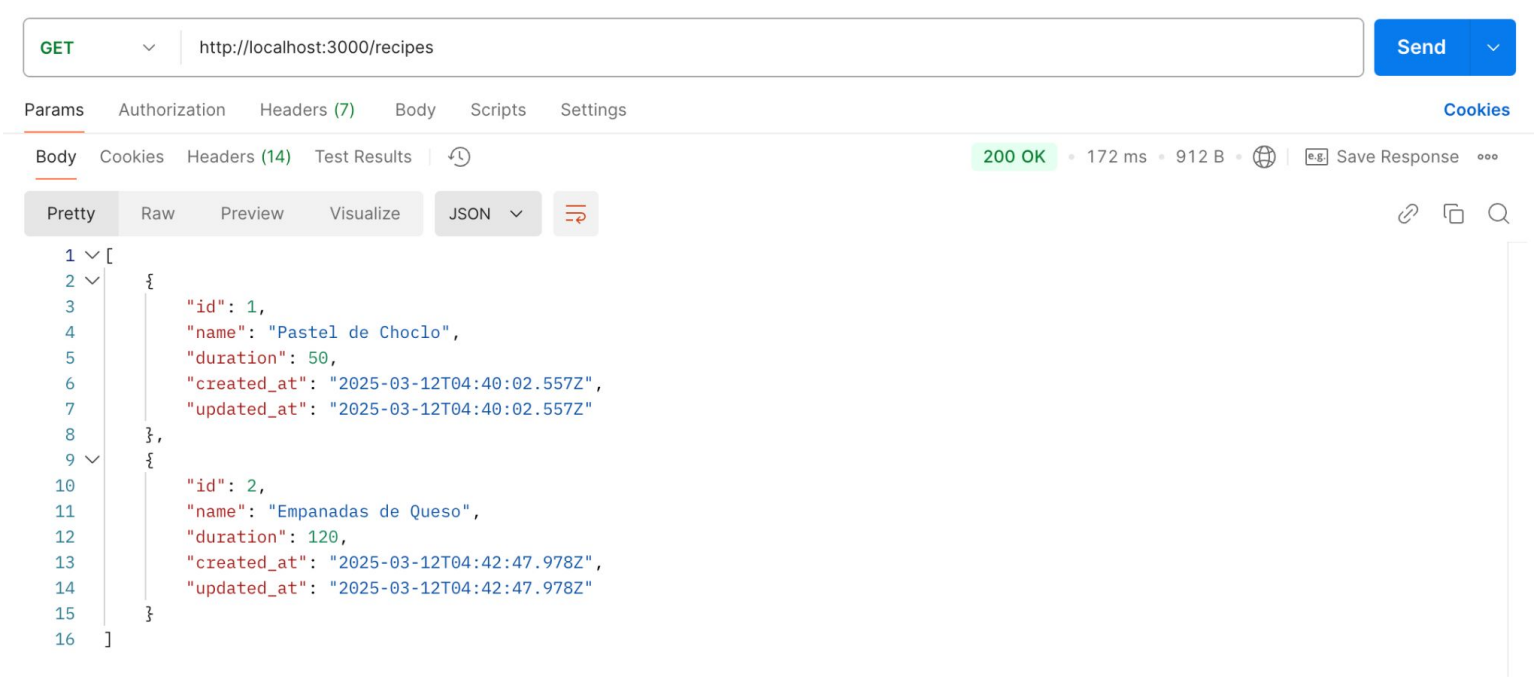
Implementar la lectura

app/controllers/recipes_controller.rb

```
class RecipesController < ApplicationController
 def index
   @recipes = Recipe.all
   render json: @recipes
 end
end
```

Probar la lectura

GET http://localhost:3000/recipes

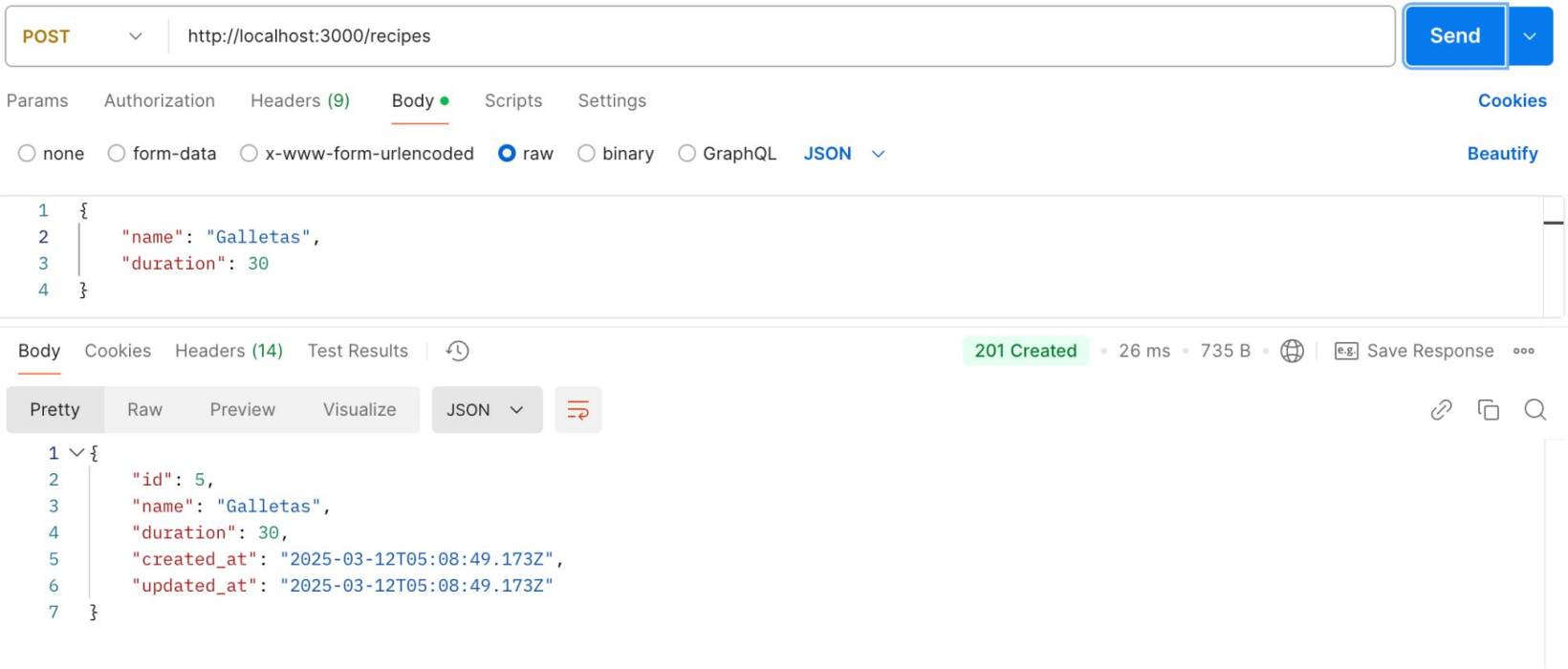


Implementar la creación

```
class RecipesController < ApplicationController
  def create
   @recipe = Recipe.new(recipe_params)
   if @recipe.save
     render json: @recipe, status: :created
   else
     render json: @recipe.errors, status: :unprocessable_entity
   end
 end
 private
 def recipe_params
   params.require(:recipe).permit(:name, :duration)
 end
end
```

Probar la creación

POST http://localhost:3000/recipes



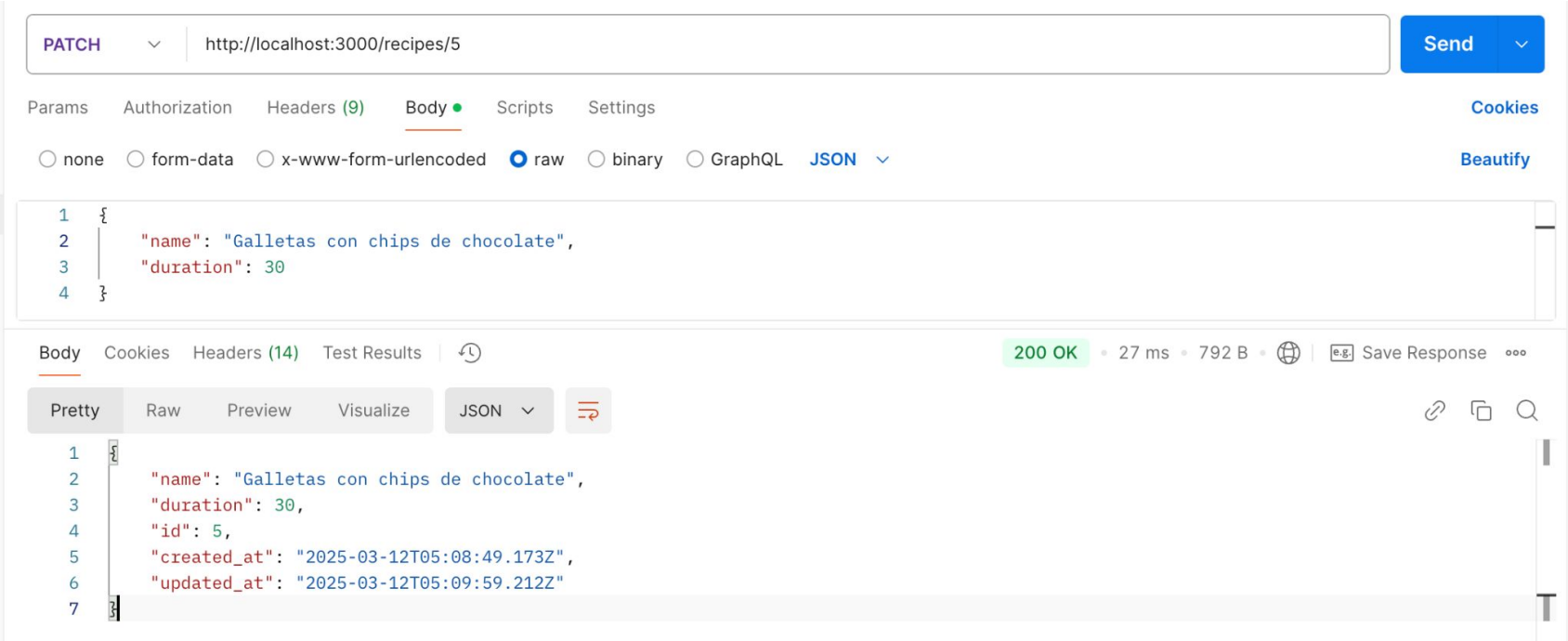
Implementar la actualización

```
class RecipesController < ApplicationController
  def update
    @recipe = Recipe.find(params[:id])
    if @recipe.update(recipe_params)
      render json: @recipe
    else
      render json: @recipe.errors, status: :unprocessable_entity
    end
  end

  private
  def recipe_params
    params.require(:recipe).permit(:name, :duration)
  end
end
```

Probar la actualización

POST http://localhost:3000/recipes/5

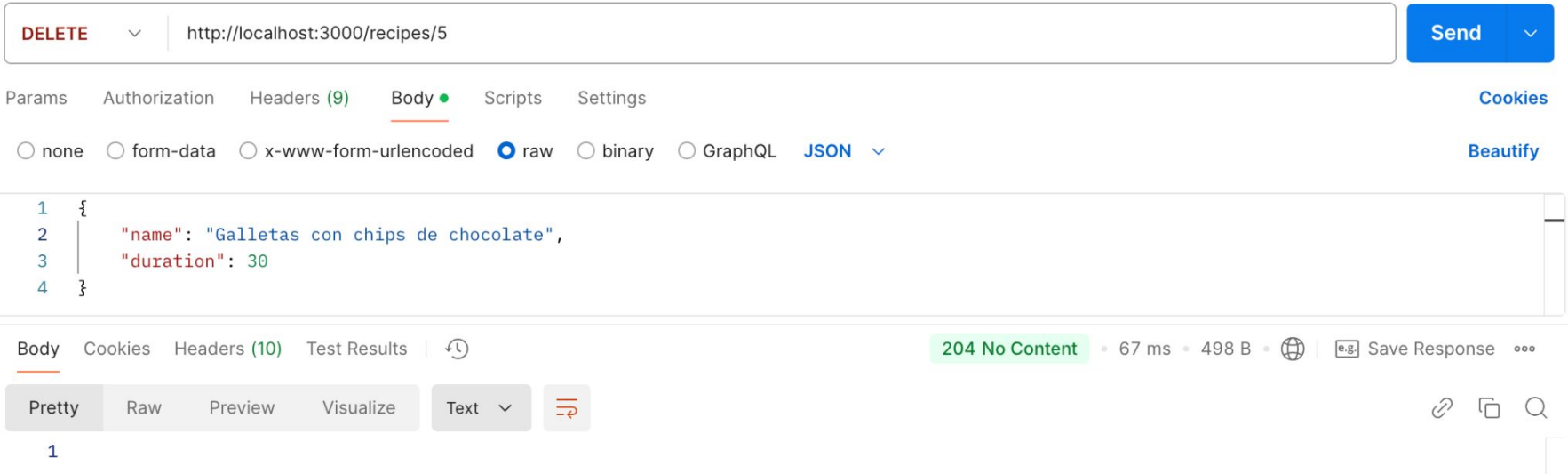


Implementar la eliminación

```
class RecipesController < ApplicationController
  def delete
    @recipe = Recipe.find(params[:id])
    @recipe.destroy
    head :no_content
 end
end
```

Probar la eliminación

DELETE http://localhost:3000/recipes/5



Crear un filtro

Paso 1: Agregar la ruta (Antes de /recipes/:id)

Paso 2: Agregar el método en el controlador

Paso 3: Probarlo

```
def filter
   @recipes = Recipe.where(duration: params[:duration])
   render json: @recipes
 end
```

Validaciones

Es obligación el nombre y la duración

```
class Recipe < ApplicationRecord
 validates :name, presence: true
 validates :duration, presence: true, numericality: { only_integer:
true, greater_than: 5 }
end
```

Validaciones

Ya no podemos crear una instancia que no cumplan las validaciones

> recipe = Recipe.new(name: "Pancakes")

> recipe.save

> recipe.errors

> recipe.valid?

```
3.3.6 :016 > recipe = Recipe.new(name:"Pancakes")
 => #<Recipe:0x0000000127478218 id: nil, name: "Pancakes", duration: nil, created_at: nil, updated_at: nil>
3.3.6 :017 > recipe.save
 => false
3.3.6 :018 > recipe.errors
 => #<ActiveModel::Errors [#<ActiveModel::Error attribute=duration, type=blank, options={}>, #<ActiveModel::Error attr
ibute=duration, type=not_a_number, options={:value=>nil}>]>
3.3.6 :019 > recipe.valid?
 => false
3.3.6 :020 >
```

Validaciones

Existen muchas validaciones disponibles para los distintos tipos de datos

https://guides.rubyonrails.org/active_record_validations.html

Asociaciones

Los modelos pueden estar relacionados de distintas formas

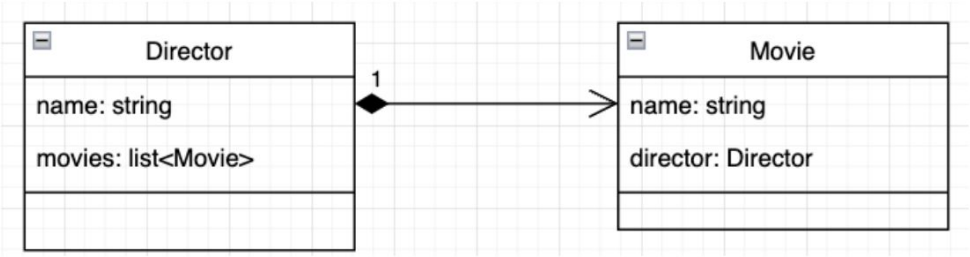
> rails new movies --api --database=postgresql

> Modelos

> rails db:create

Asociaciones

- Un director puede haber dirigido varias películas
- Una película pertenece a un director



> rails generate model Director name:string

> rails generate model Movie name:string

Asociaciones

- Agregamos las asociaciones a la migración y al modelo

/db/migrate/[fecha]_create_movies.rb

```
class CreateMovies < ActiveRecord::Migration[7.1]
 def change
   create_table :movies do |t|
     t.string :name
     t.belongs_to :director
     t.timestamps
   end
 end
end
```

Asociaciones

- Agregamos las asociaciones a la migración y al modelo

/db/migrate/[fecha]_create_directors.rb

```
class CreateDirectors < ActiveRecord::Migration[7.0]
 def change
   create_table :directors do |t|
     t.string :name

     t.timestamps
   end
 end
end
```

Asociaciones

- Agregamos las asociaciones a la migración y al modelo

/app/models/director.rb

```
class Director < ApplicationRecord
 has_many :movies
end
```

/app/models/movie.rb

```
class Movie < ApplicationRecord
 belongs_to :director
end
```

Probar las asociaciones

› rails db:create

› rails db:migrate

› rails c

```
3.3.6 :002 > tarantino = Director.new(name: "Tarantino")
 => #<Director:0x0000000120787de8 id: nil, name: "Tarantino", created_at: nil, updated_at: nil>
3.3.6 :003 > tarantino.save
  TRANSACTION (0.5ms)  BEGIN
  Director Create (25.4ms)  INSERT INTO "directors" ("name", "created_at", "updated_at") VALUES ($1, $2, $3) RE
TURNING "id"  [["name", "Tarantino"], ["created_at", "2025-03-12 13:27:55.562095"], ["updated_at", "2025-03-12
13:27:55.562095"]]
  TRANSACTION (9.5ms)  COMMIT
 => true
3.3.6 :004 > pulp_fiction = Movie.new(name: "Pulp Fiction", director: tarantino)
 => #<Movie:0x0000000120424598 id: nil, name: "Pulp Fiction", director_id: 1, created_at: nil, updated_at: nil>

3.3.6 :005 > pulp_fiction.save
  TRANSACTION (0.4ms)  BEGIN
  Movie Create (5.5ms)  INSERT INTO "movies" ("name", "director_id", "created_at", "updated_at") VALUES ($1, $2
, $3, $4) RETURNING "id"  [["name", "Pulp Fiction"], ["director_id", 1], ["created_at", "2025-03-12 13:30:57.53
8778"], ["updated_at", "2025-03-12 13:30:57.538778"]]
  TRANSACTION (0.3ms)  COMMIT
 => true
```

Probar las asociaciones

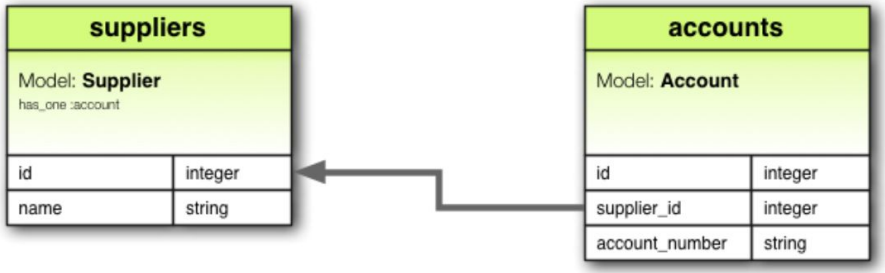
› tarantino.movies

```
3.3.6 :008 > tarantino.movies
  Movie Load (7.9ms)  SELECT "movies".* FROM "movies" WHERE "movies"."director_id" = $1  [["director_id", 1]]
 =>
[#<Movie:0x000000012084ed80
  id: 1,
  name: "Pulp Fiction",
  director_id: 1,
  created_at: Wed, 12 Mar 2025 13:30:57.538778000 UTC +00:00,
  updated_at: Wed, 12 Mar 2025 13:30:57.538778000 UTC +00:00>,
 #<Movie:0x000000012084ec40
  id: 2,
  name: "Kill Bill",
  director_id: 1,
  created_at: Wed, 12 Mar 2025 13:33:42.266909000 UTC +00:00,
  updated_at: Wed, 12 Mar 2025 13:33:42.266909000 UTC +00:00>]
```

Otras asociaciones

Asociación 1 a 1

- Una cuenta tiene un Proveedor
- en Rails: has_one

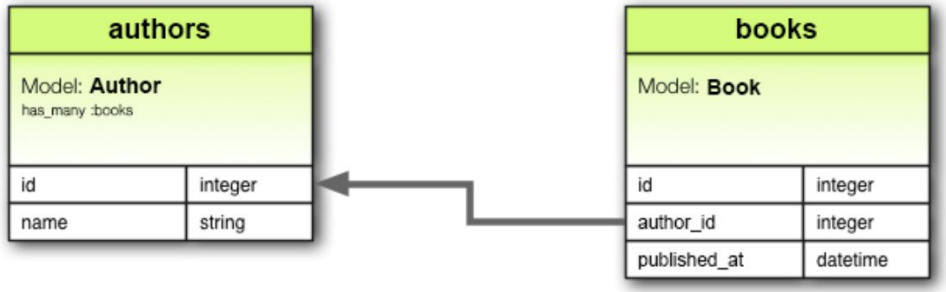



```
class Supplier < ApplicationRecord
  has_one :account
end
```

Otras asociaciones

Asociación 1 a muchos

- Un autor tiene muchos libros
- en Rails: has_many

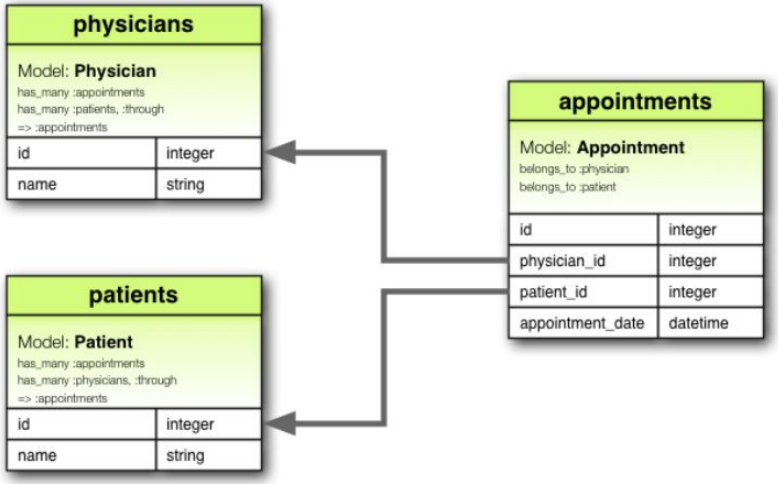



```
class Author < ApplicationRecord
  has_many :books
end
```

Otras asociaciones

Asociación muchos a muchos

- Un paciente tiene muchos médicos a través de horas agendadas
- en Rails: has_many :through




```ruby
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
```

Documentación

Doc ActiveRecord:

https://guides.rubyonrails.org/active_record_basics.html

Doc asociaciones:

https://guides.rubyonrails.org/association_basics.html

Doc migraciones:

https://guides.rubyonrails.org/active_record_migrations.html

Debugging en rails:

https://guides.rubyonrails.org/v6.1/debugging_rails_applications.html

Convenciones

- Nombre modelos/clases y tablas/esquemas
- favorece los nombre en plural para mantener el uso de generadores de rutas por defecto