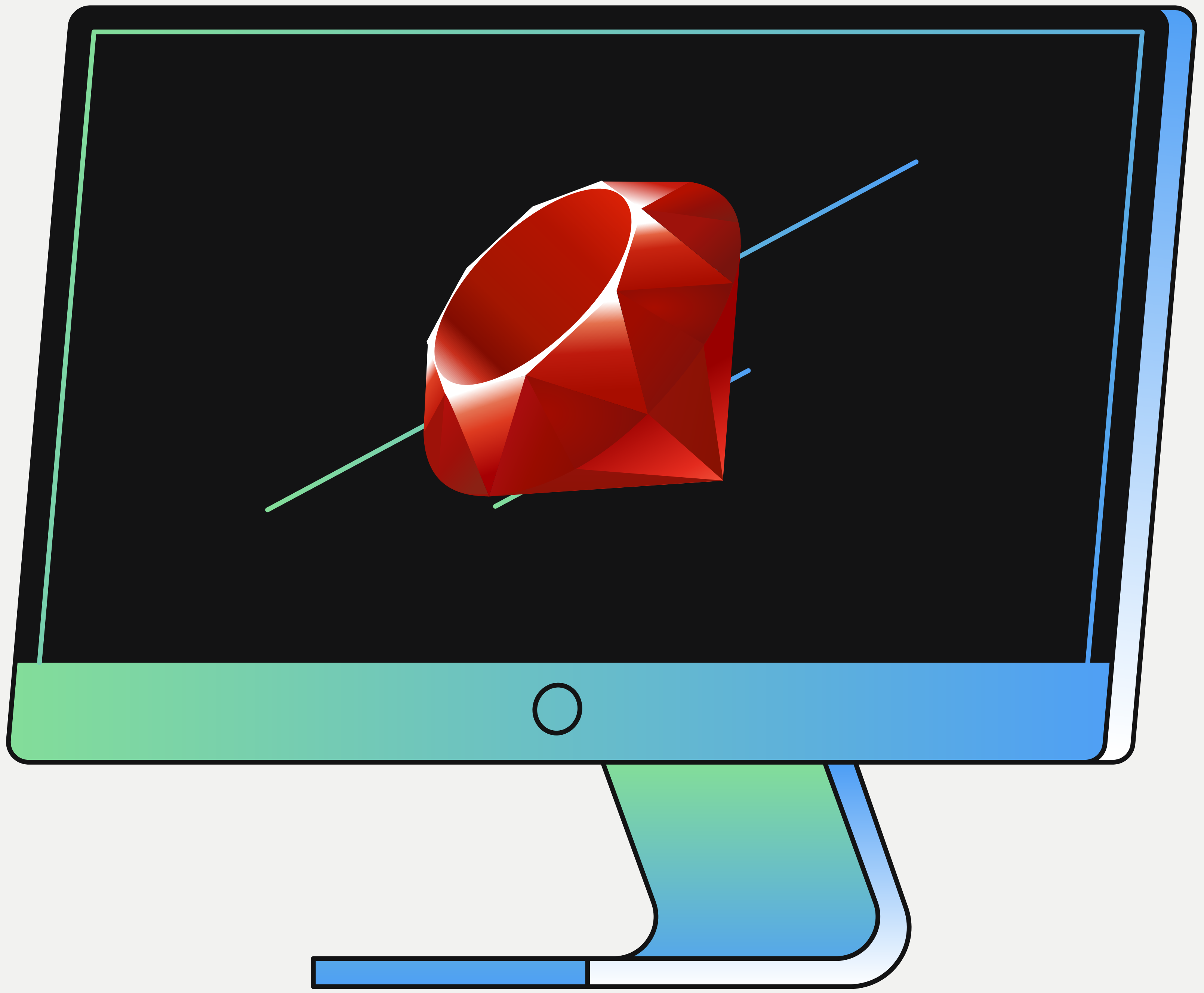
SEMESTRE
2023-2

AYUDANTES
Jean Philipe - Nicolás Fernández - Valentina Massé

IIC2143

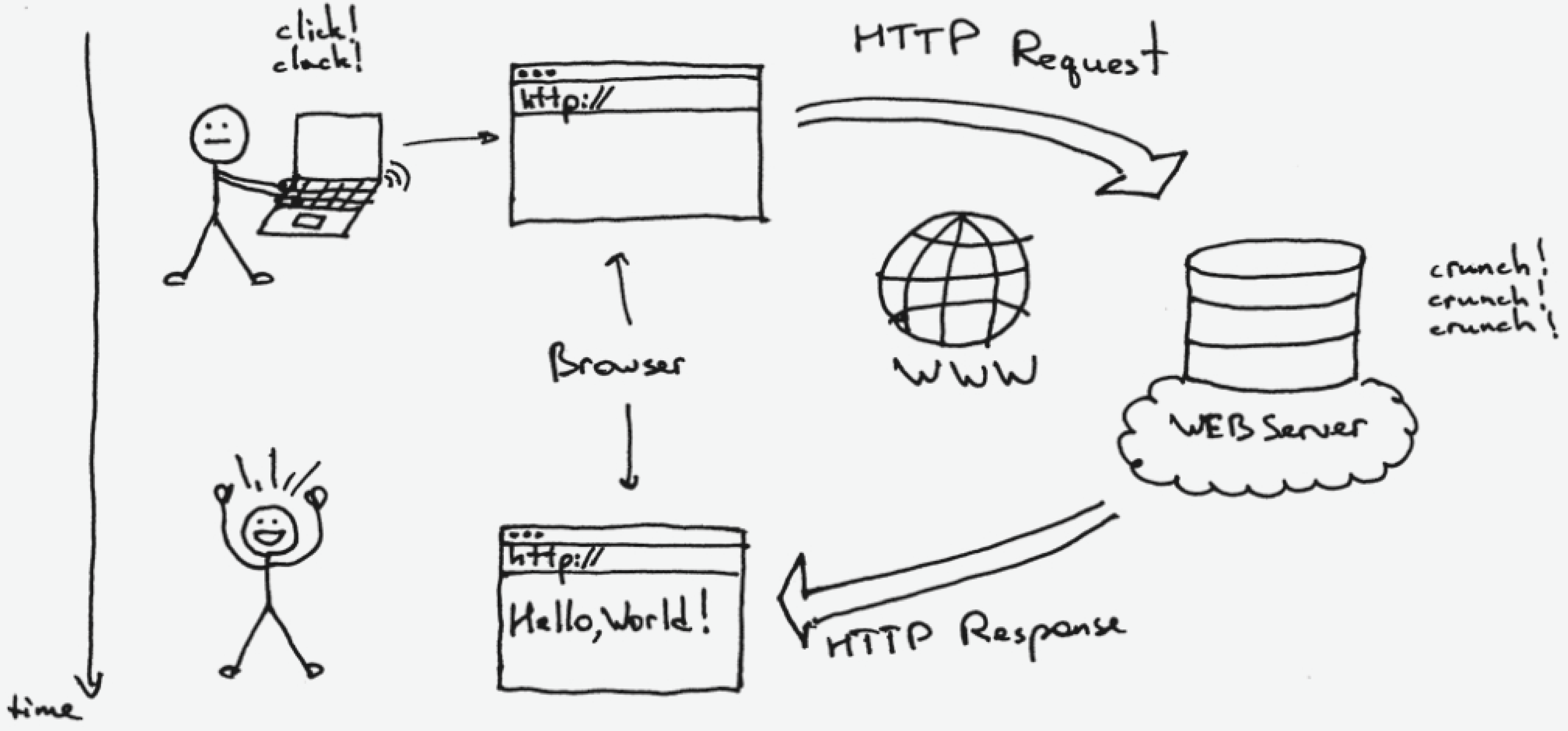
Models Views Controller



¿Qué veremos hoy?

- Protocolo HTTP
- Ruby on Rails
- Modelos, Vistas y Controladores
- Ejemplo en Rails

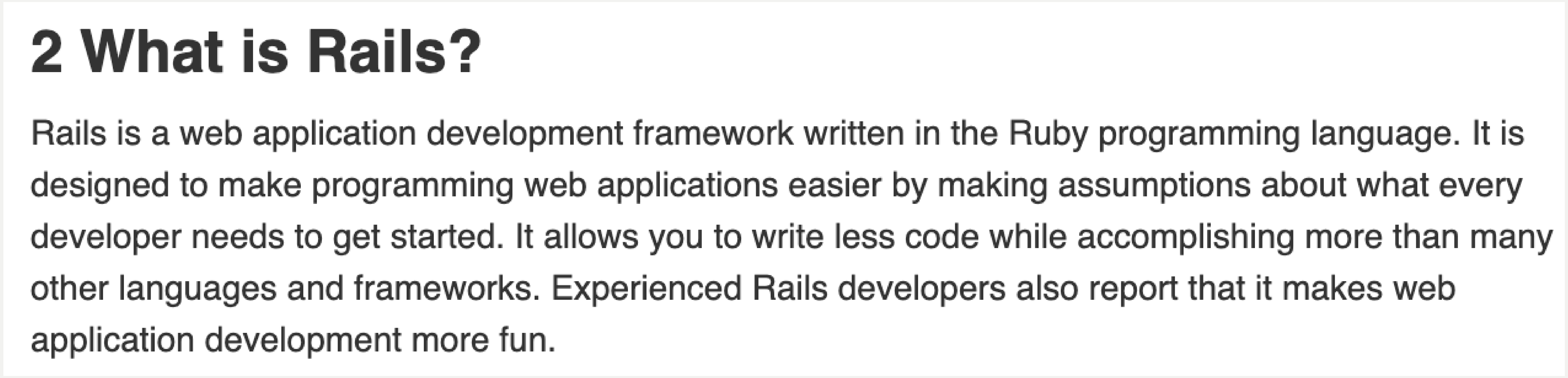
Protocolo HTTP



¿Y si queremos modificar el contenido de nuestra página haciendo requests al servidor?

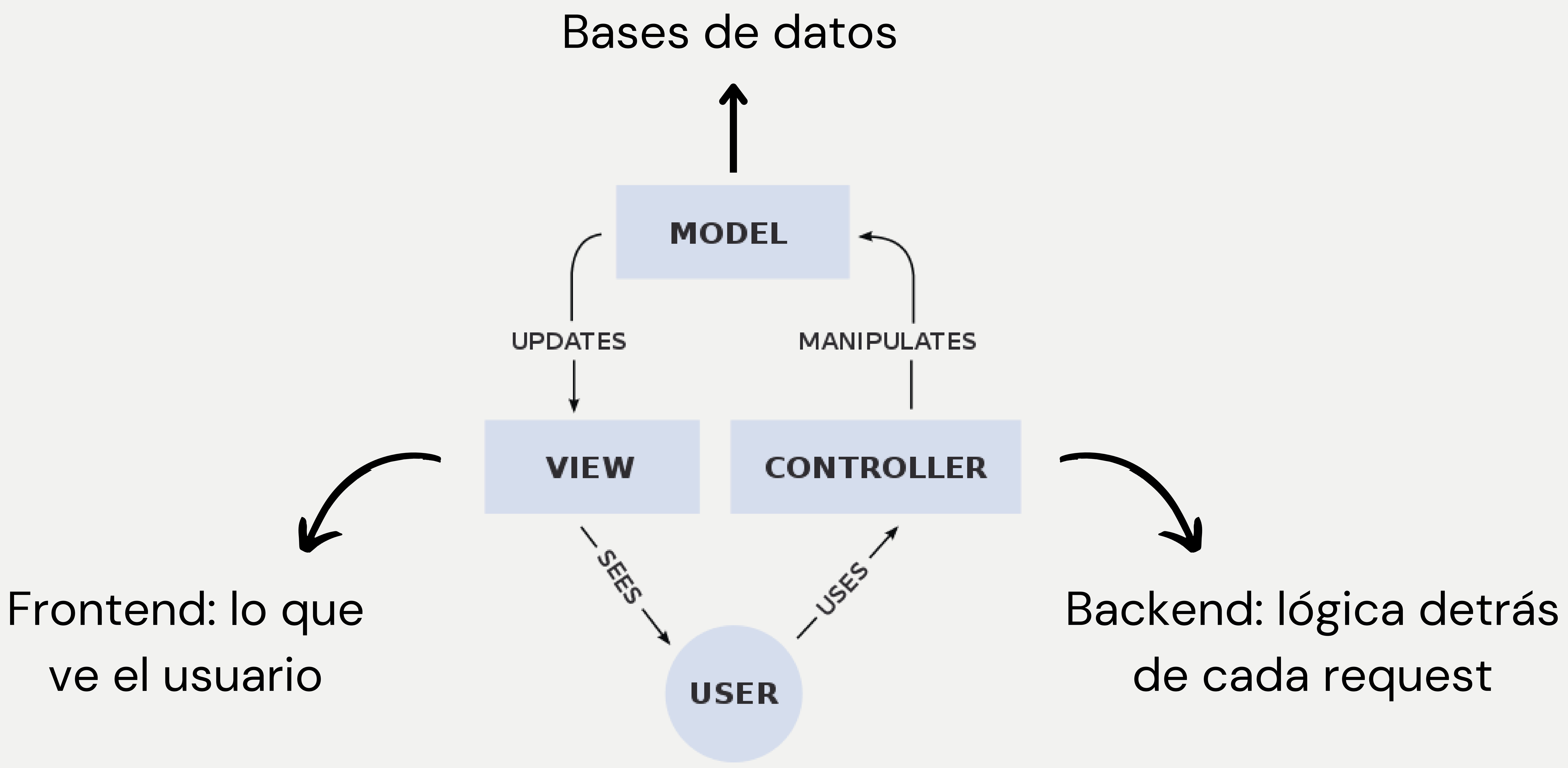
Necesitamos trabajar con lógica adicional

Ruby on Rails

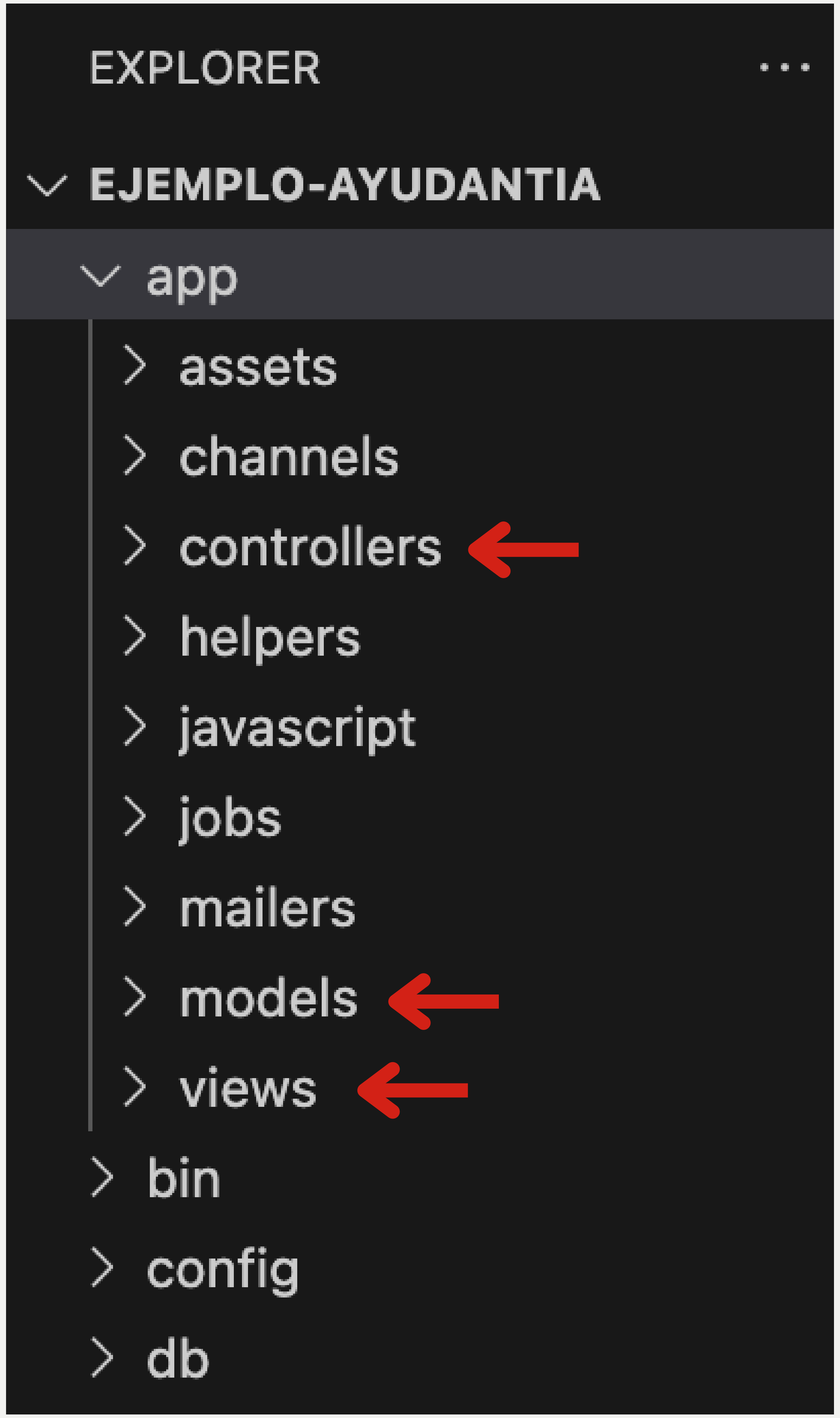


2 What is Rails?

Rails is a web application development framework written in the Ruby programming language. It is designed to make programming web applications easier by making assumptions about what every developer needs to get started. It allows you to write less code while accomplishing more than many other languages and frameworks. Experienced Rails developers also report that it makes web application development more fun.


Bases de datos
MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER
Frontend: lo que ve el usuario
Backend: lógica detrás de cada request

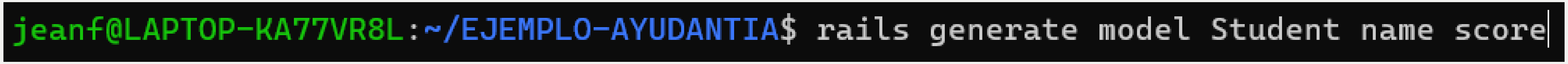
¿Dónde los encuentro en mi proyecto?



Modelos

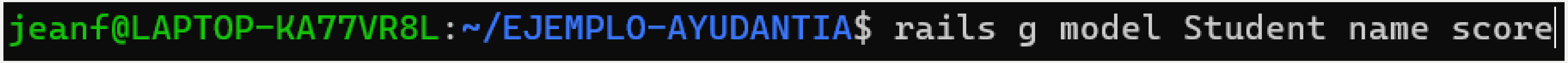
Son clases de Ruby que se usan para representar datos. Pueden interactuar con la base de datos de la aplicación a través de Active Record.

Para crear un modelo:



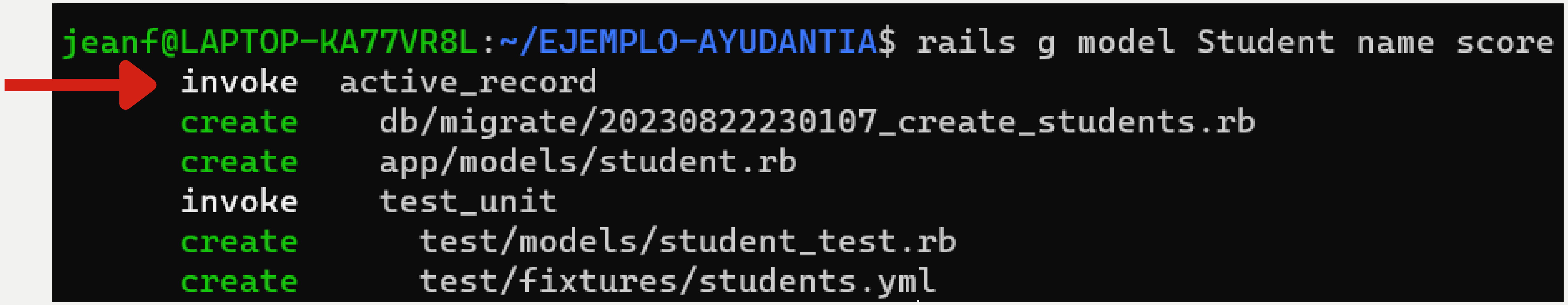
```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails generate model Student name score
```

Otra forma equivalente:

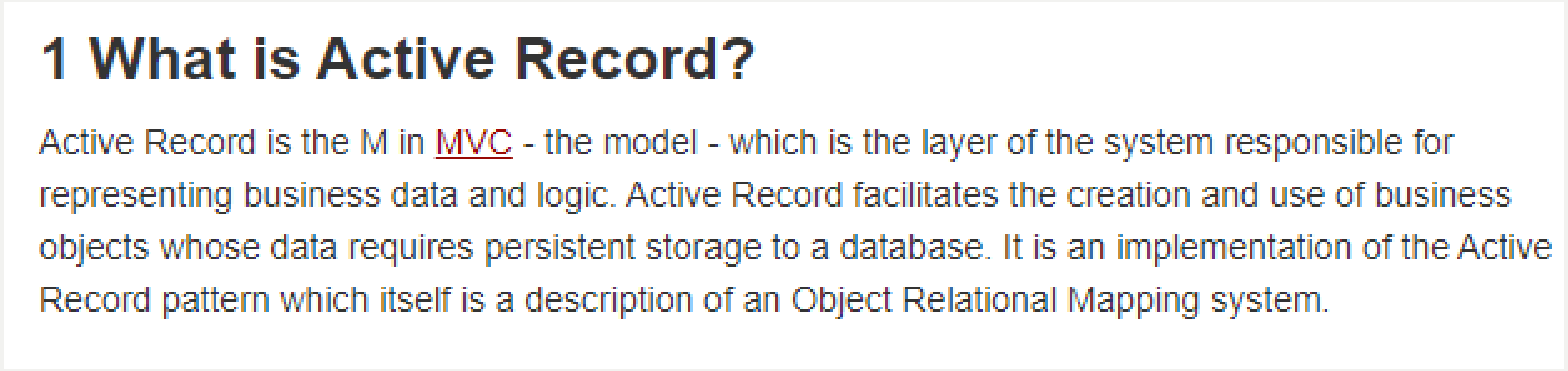


```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails g model Student name score
```

Active Record

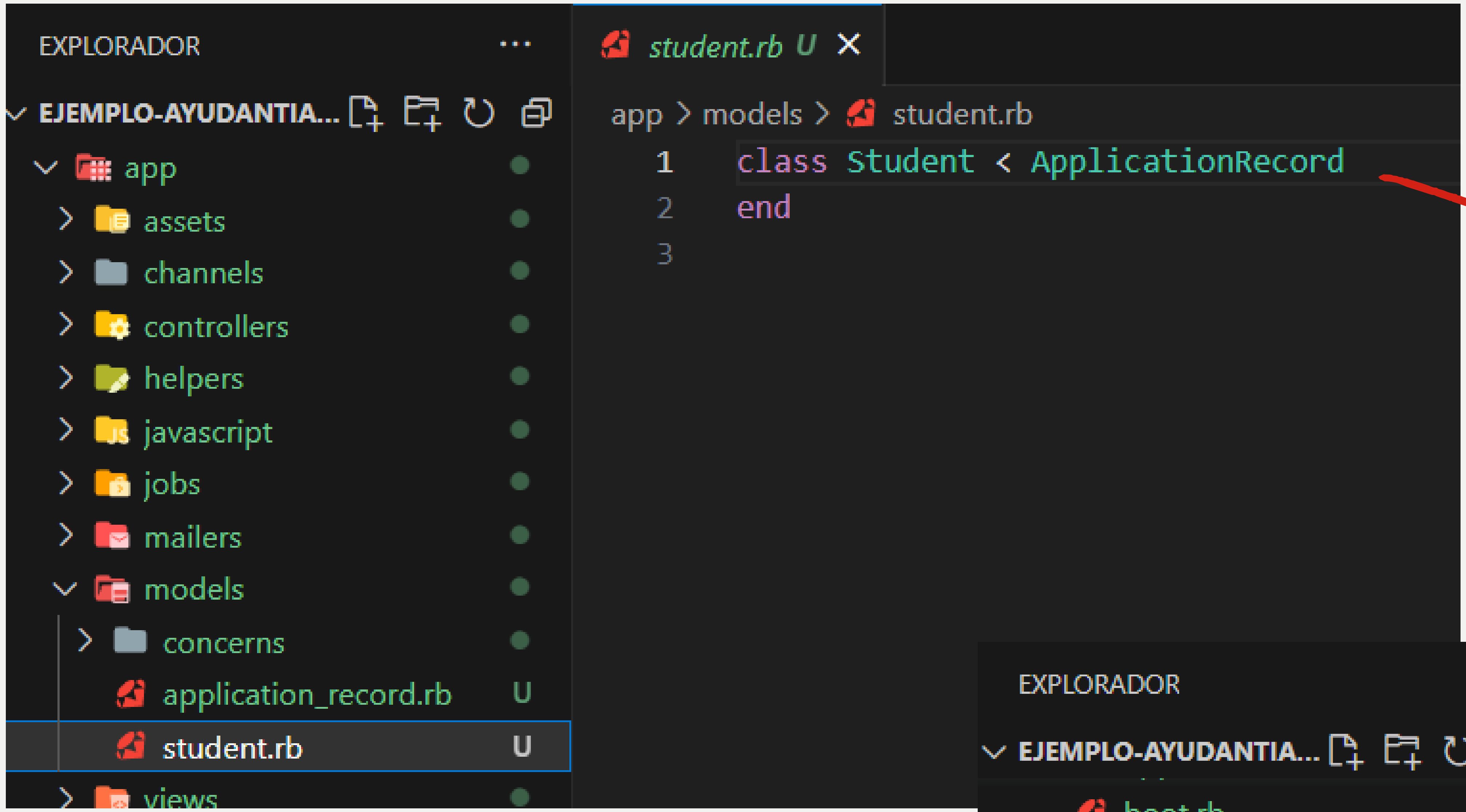


```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails g model Student name score
      invoke  active_record
      create    db/migrate/20230822230107_create_students.rb
      create    app/models/student.rb
      invoke    test_unit
      create      test/models/student_test.rb
      create      test/fixtures/students.yml
```



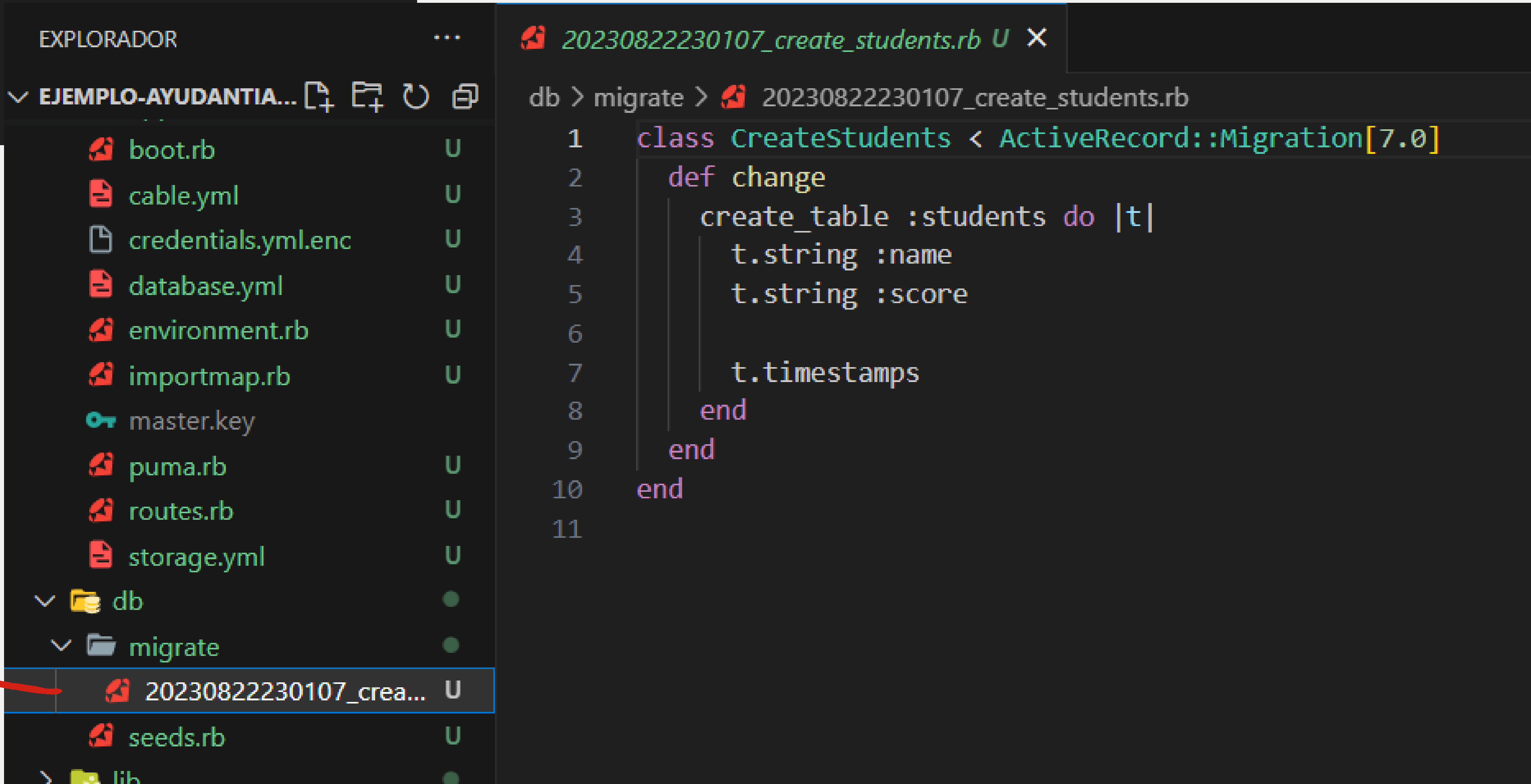
1 What is Active Record?

Active Record is the M in MVC - the model - which is the layer of the system responsible for representing business data and logic. Active Record facilitates the creation and use of business objects whose data requires persistent storage to a database. It is an implementation of the Active Record pattern which itself is a description of an Object Relational Mapping system.



```ruby
class Student < ApplicationRecord
end
```

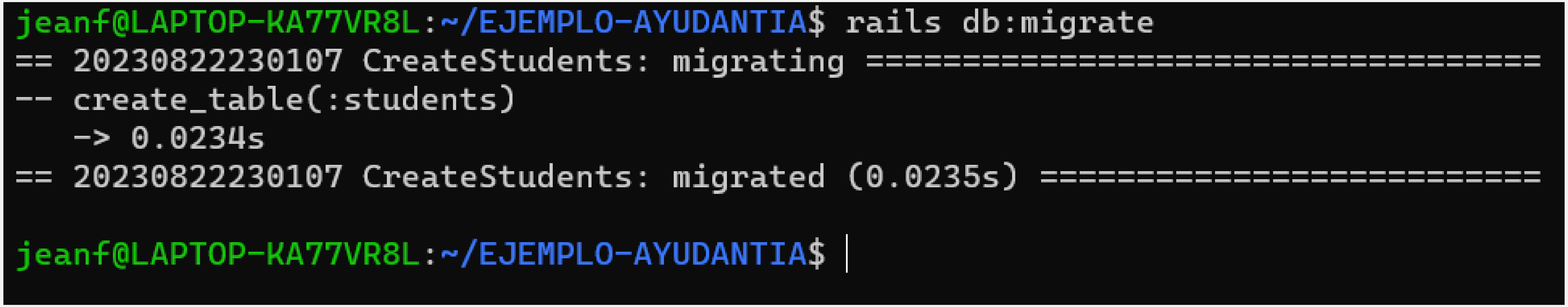
Archivo del modelo



```ruby
class CreateStudents < ActiveRecord::Migration[7.0]
  def change
    create_table :students do |t|
      t.string :name
      t.string :score

      t.timestamps
    end
  end
end
```

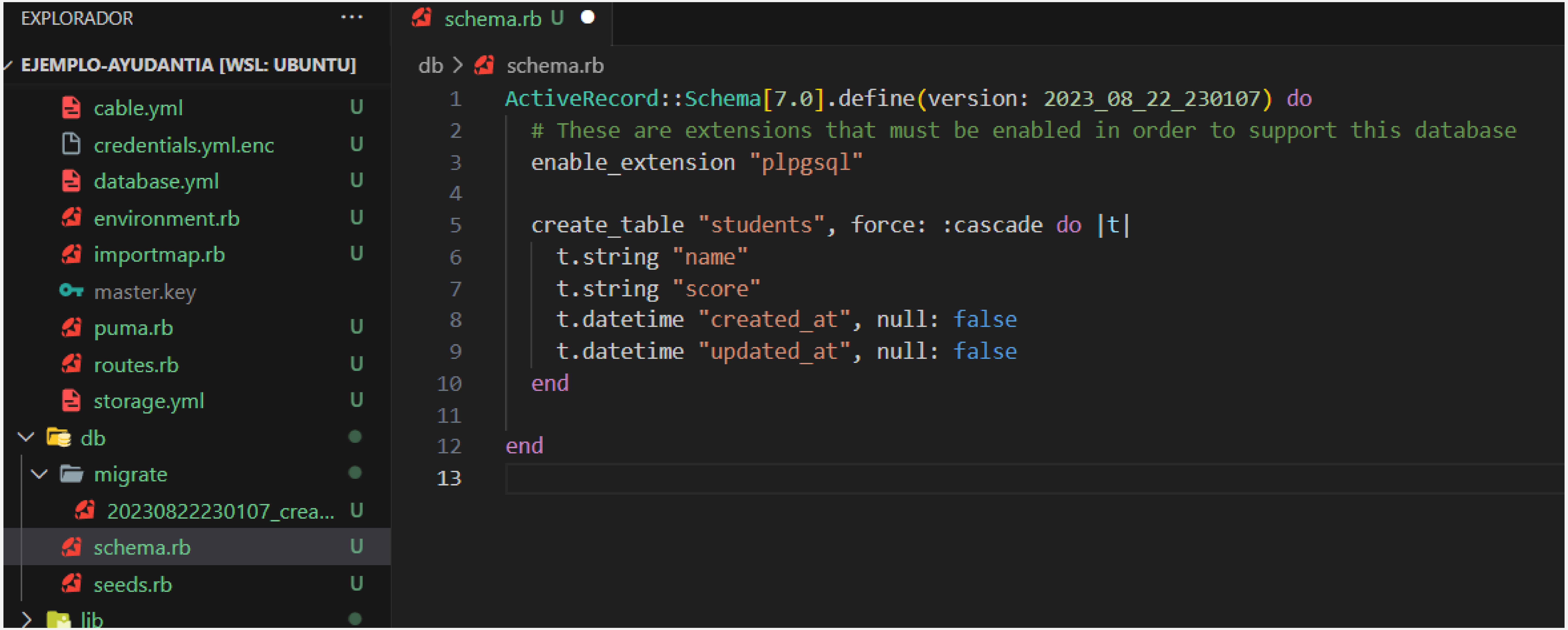
Archivo de la migración

¡Falta correr la migración para crear la tabla en la base de datos!



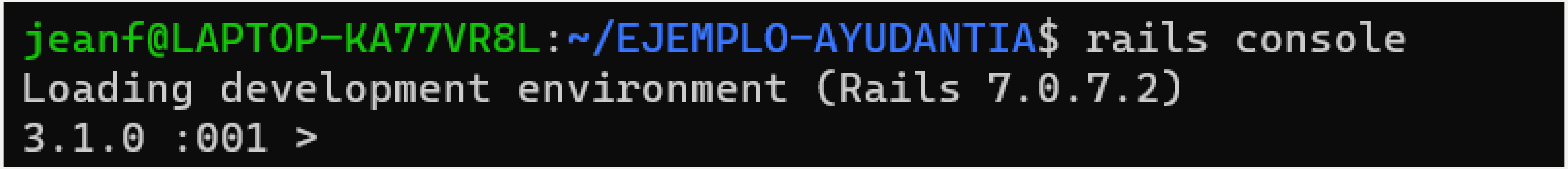
```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails db:migrate
== 20230822230107 CreateStudents: migrating ===================================
-- create_table(:students)
   -> 0.0234s
== 20230822230107 CreateStudents: migrated (0.0235s) ==========================

jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$
```

Rails agrega la tabla al schema donde se pueden ver todas las tablas creadas en el proyecto con sus atributos

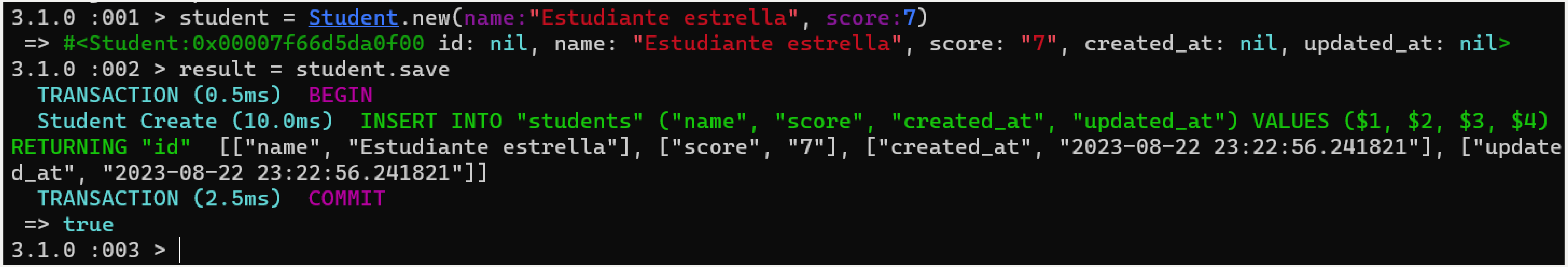


```ruby
ActiveRecord::Schema[7.0].define(version: 2023_08_22_230107) do
  # These are extensions that must be enabled in order to support this database
  enable_extension "plpgsql"

  create_table "students", force: :cascade do |t|
    t.string "name"
    t.string "score"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

end

```

Podemos probar manipular el modelo desde la consola de rails

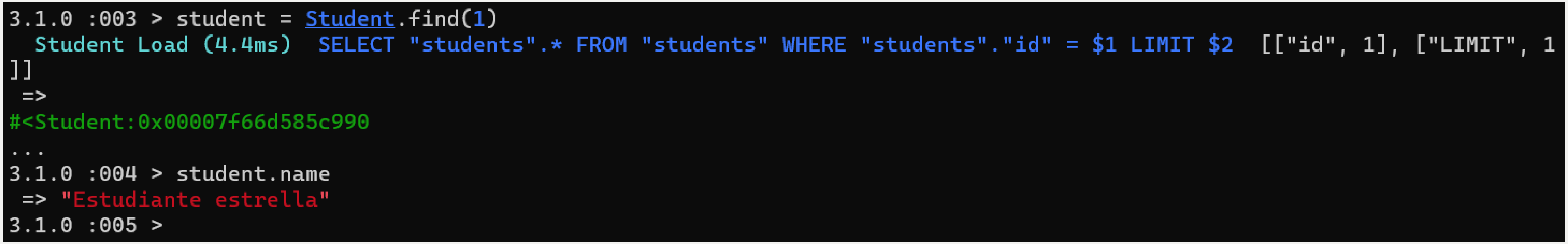


```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails console
Loading development environment (Rails 7.0.7.2)
3.1.0 :001 >
```



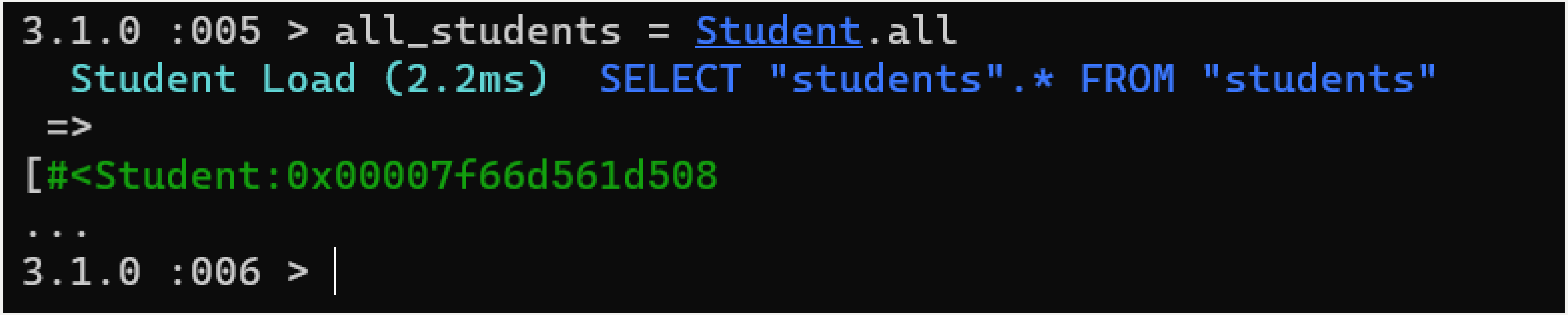
```
3.1.0 :001 > student = Student.new(name:"Estudiante estrella", score:7)
 => #<Student:0x00007f66d5da0f00 id: nil, name: "Estudiante estrella", score: "7", created_at: nil, updated_at: nil>
3.1.0 :002 > result = student.save
  TRANSACTION (0.5ms)  BEGIN
  Student Create (10.0ms)  INSERT INTO "students" ("name", "score", "created_at", "updated_at") VALUES ($1, $2, $3, $4)
RETURNING "id"  [["name", "Estudiante estrella"], ["score", "7"], ["created_at", "2023-08-22 23:22:56.241821"], ["update
d_at", "2023-08-22 23:22:56.241821"]]
  TRANSACTION (2.5ms)  COMMIT
 => true
3.1.0 :003 >
```

Podemos filtrar en la db por Id



```
3.1.0 :003 > student = Student.find(1)
  Student Load (4.4ms)  SELECT "students".* FROM "students" WHERE "students"."id" = $1 LIMIT $2  [["id", 1], ["LIMIT", 1
]]
 =>
#<Student:0x00007f66d585c990
...
3.1.0 :004 > student.name
 => "Estudiante estrella"
3.1.0 :005 >
```

Podemos recuperar todos los elementos de una tabla

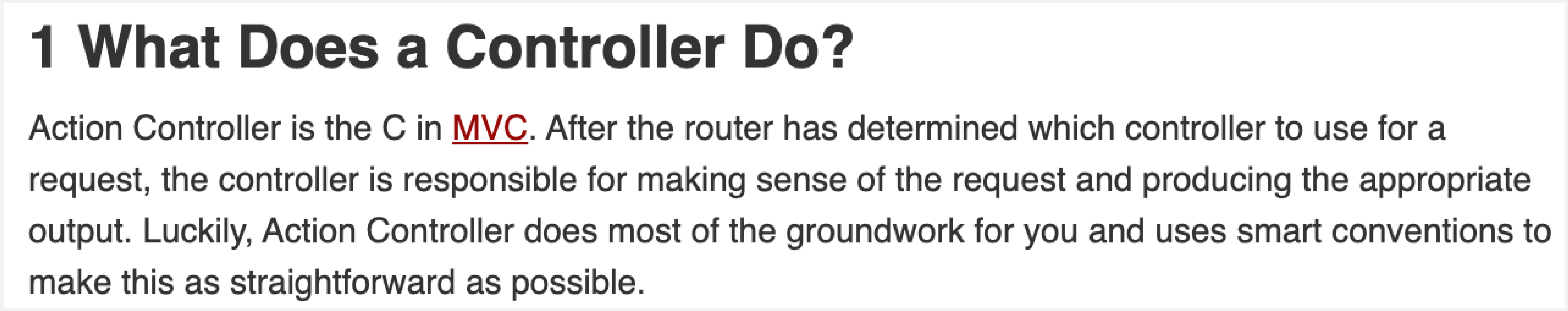


```
3.1.0 :005 > all_students = Student.all
  Student Load (2.2ms)  SELECT "students".* FROM "students"
 =>
[#<Student:0x00007f66d561d508
...
3.1.0 :006 >
```

¿De qué me sirve manejarme con la consola de Rails?

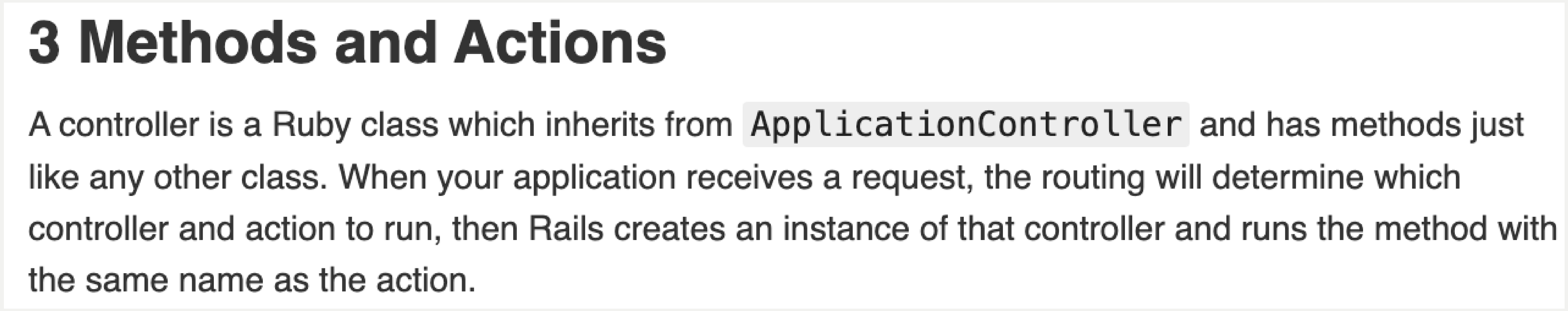
Los mismos metodos se pueden ocupar desde los controladores!

Controladores



1 What Does a Controller Do?

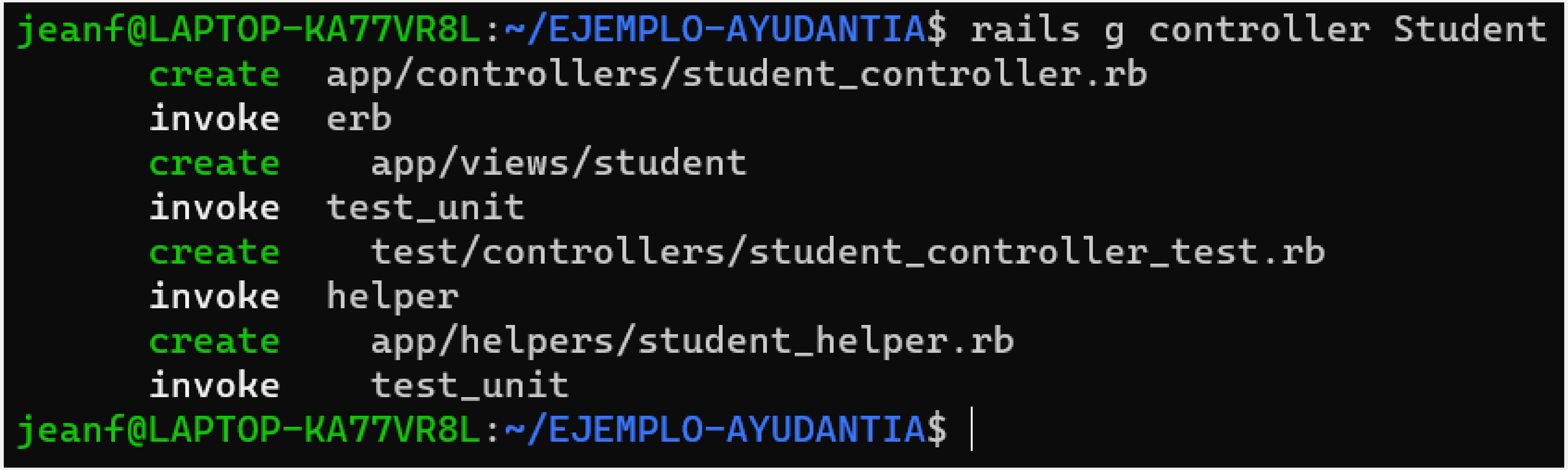
Action Controller is the C in MVC. After the router has determined which controller to use for a request, the controller is responsible for making sense of the request and producing the appropriate output. Luckily, Action Controller does most of the groundwork for you and uses smart conventions to make this as straightforward as possible.



3 Methods and Actions

A controller is a Ruby class which inherits from `ApplicationController` and has methods just like any other class. When your application receives a request, the routing will determine which controller and action to run, then Rails creates an instance of that controller and runs the method with the same name as the action.

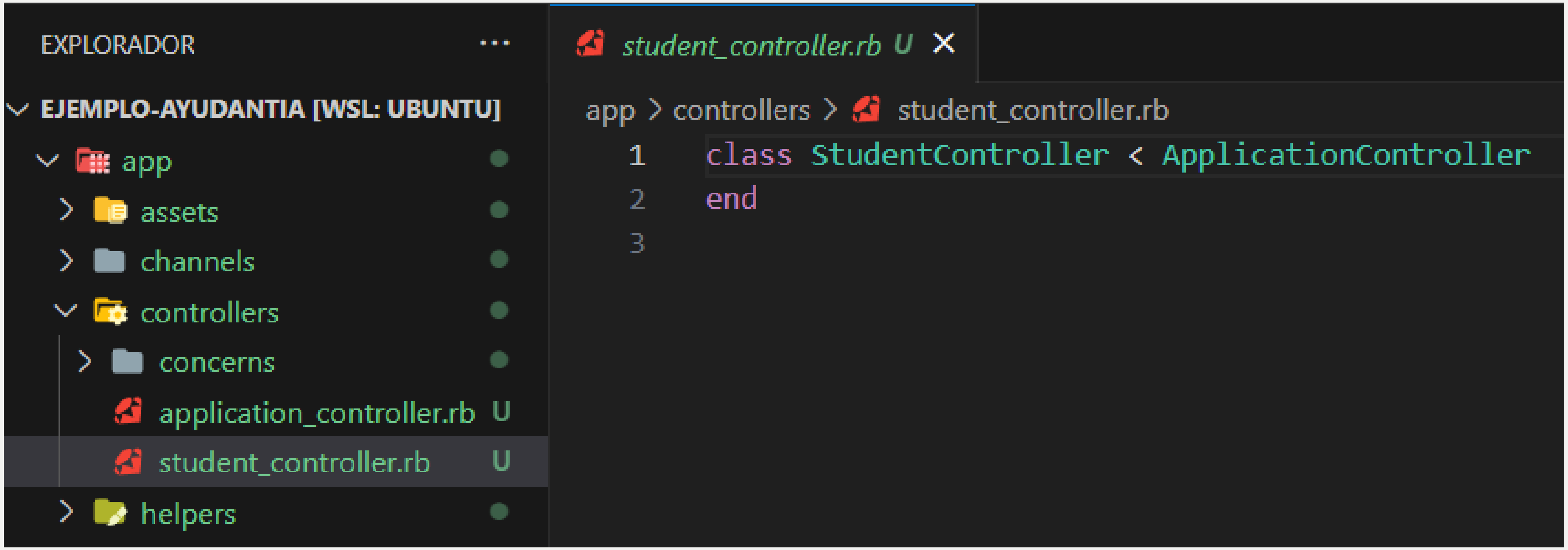
Crear un controlador



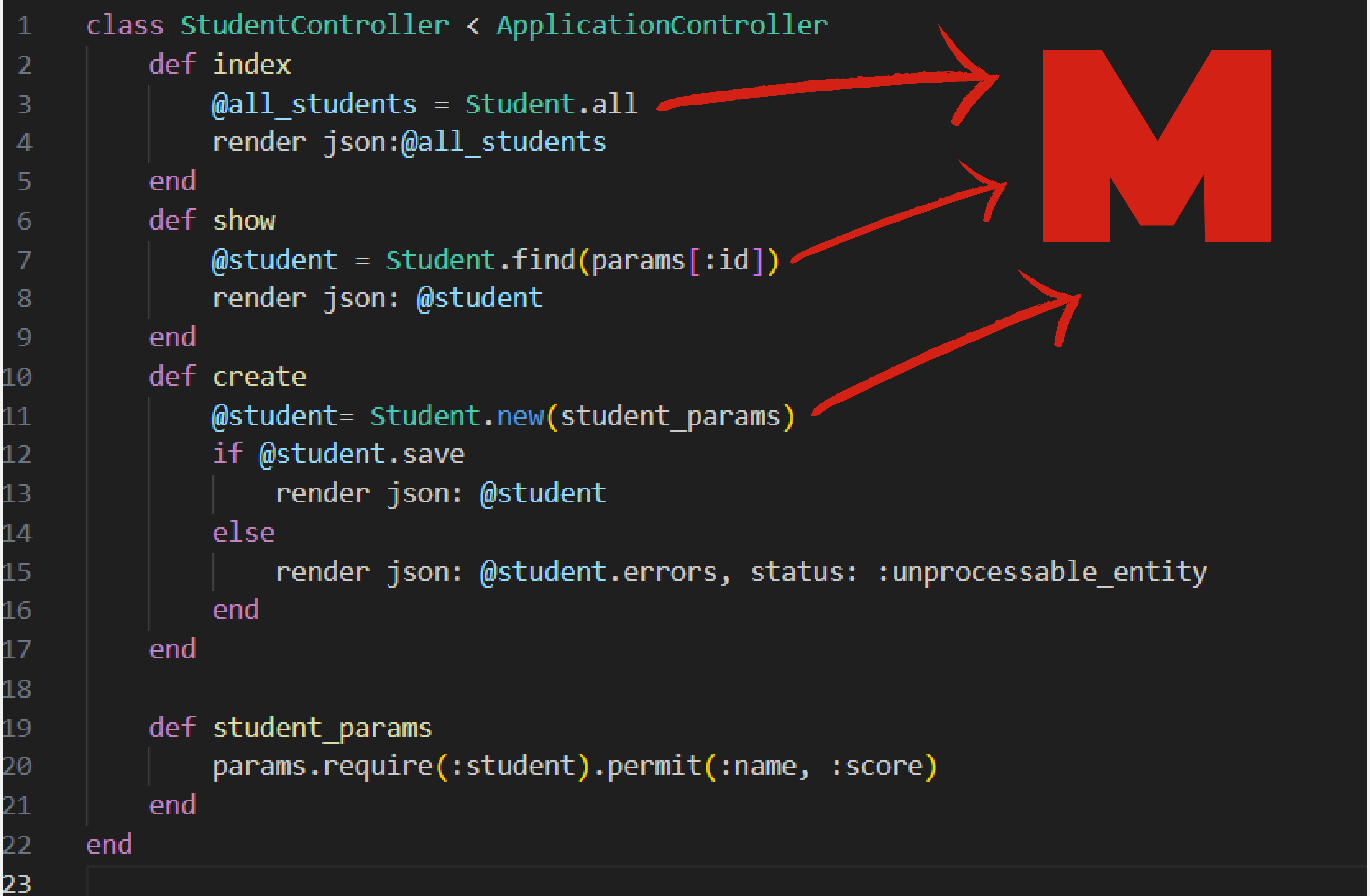
```
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ rails g controller Student
      create  app/controllers/student_controller.rb
      invoke  erb
      create    app/views/student
      invoke  test_unit
      create    test/controllers/student_controller_test.rb
      invoke  helper
      create    app/helpers/student_helper.rb
      invoke    test_unit
jeanf@LAPTOP-KA77VR8L:~/EJEMPLO-AYUDANTIA$ 
```

Mantenemos consistencia entre el nombre del modelo y del controlador por convención

¿Dónde lo encuentro?

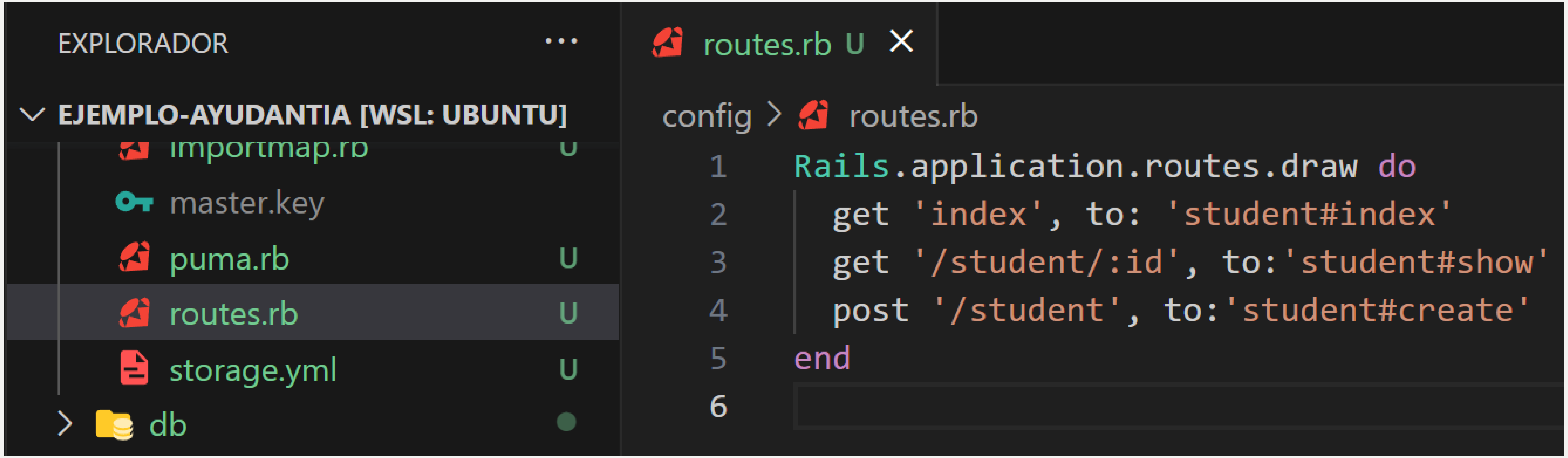

EXPLORADOR
EJEMPLO-AYUDANTIA [WSL: UBUNTU]
app
assets
channels
controllers
concerns
application_controller.rb U
student_controller.rb U
helpers
student_controller.rb U
app > controllers > student_controller.rb

```ruby
class StudentController < ApplicationController
end
```



```ruby
class StudentController < ApplicationController
    def index
        @all_students = Student.all
        render json:@all_students
    end
    def show
        @student = Student.find(params[:id])
        render json: @student
    end
    def create
        @student= Student.new(student_params)
        if @student.save
            render json: @student
        else
            render json: @student.errors, status: :unprocessable_entity
        end
    end

    def student_params
        params.require(:student).permit(:name, :score)
    end
end
```

Las rutas se manejan en routes.rb



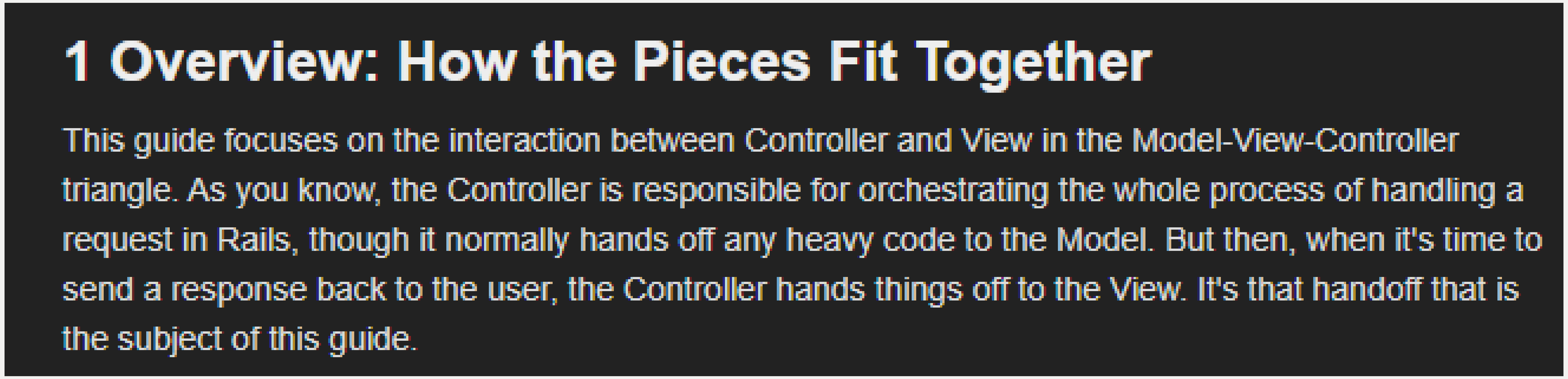
```ruby
Rails.application.routes.draw do
  get 'index', to: 'student#index'
  get '/student/:id', to:'student#show'
  post '/student', to:'student#create'
end
```

El formato siempre es:

Metodo 'ruta (URL)', to: 'nombre_controlador#metodo_controlador'

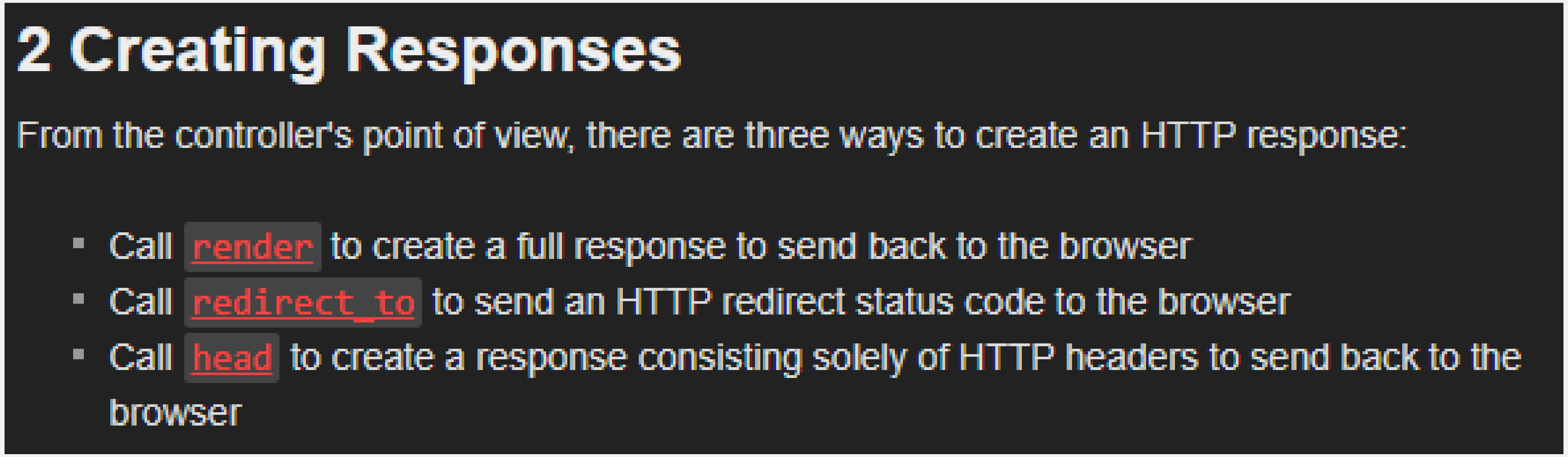
$\searrow$ **get, post, delete, patch**

Vistas



1 Overview: How the Pieces Fit Together

This guide focuses on the interaction between Controller and View in the Model-View-Controller triangle. As you know, the Controller is responsible for orchestrating the whole process of handling a request in Rails, though it normally hands off any heavy code to the Model. But then, when it's time to send a response back to the user, the Controller hands things off to the View. It's that handoff that is the subject of this guide.

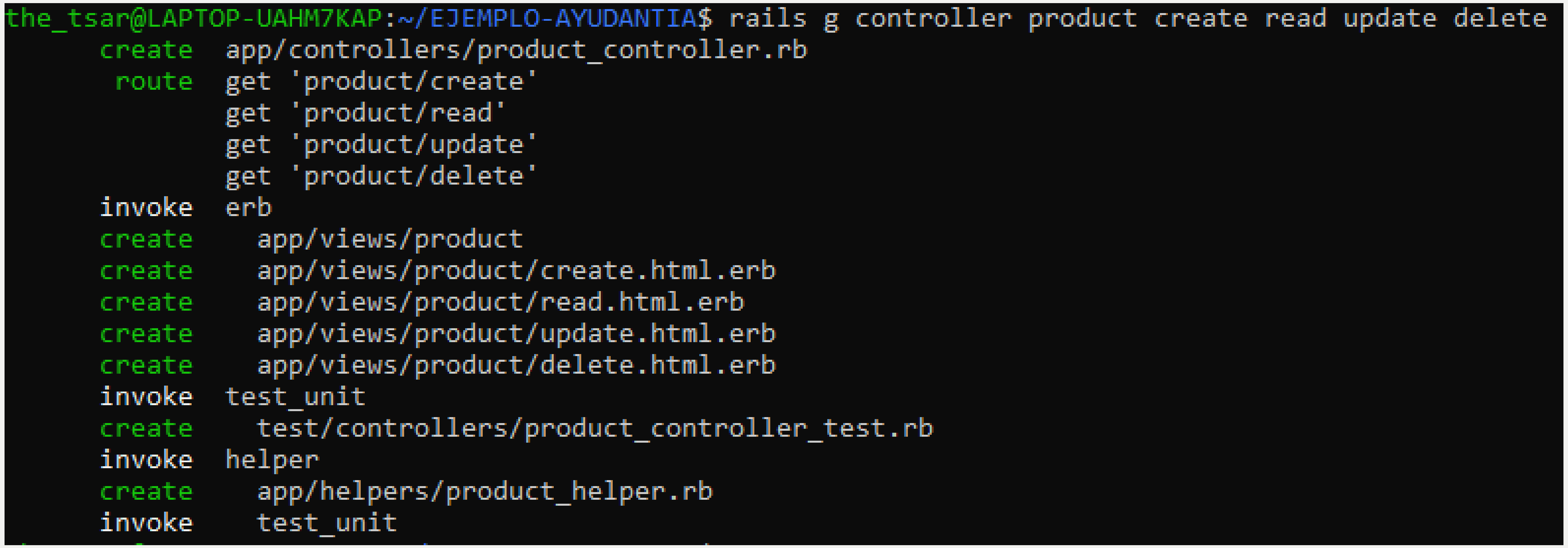


2 Creating Responses

From the controller's point of view, there are three ways to create an HTTP response:

- Call `render` to create a full response to send back to the browser
- Call `redirect_to` to send an HTTP redirect status code to the browser
- Call `head` to create a response consisting solely of HTTP headers to send back to the browser

Crear views a partir de un controlador



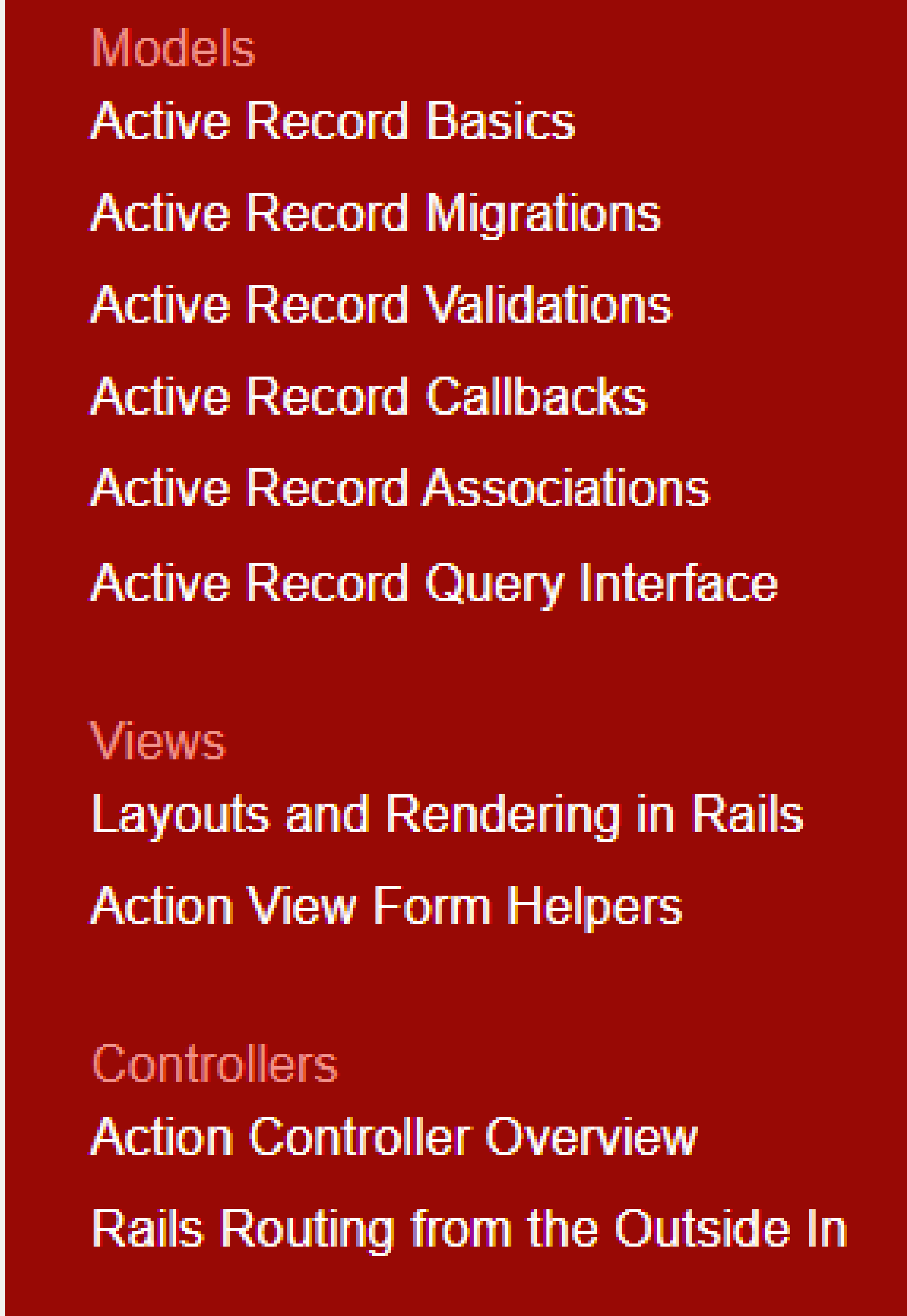
```
the_tsar@LAPTOP-UAHM7KAP:~/EJEMPLO-AYUDANTIA$ rails g controller product create read update delete
      create  app/controllers/product_controller.rb
       route  get 'product/create'
              get 'product/read'
              get 'product/update'
              get 'product/delete'
      invoke  erb
      create    app/views/product
      create    app/views/product/create.html.erb
      create    app/views/product/read.html.erb
      create    app/views/product/update.html.erb
      create    app/views/product/delete.html.erb
      invoke  test_unit
      create    test/controllers/product_controller_test.rb
      invoke  helper
      create    app/helpers/product_helper.rb
      invoke    test_unit
```

Mantenemos consistencia entre el nombre del modelo y del controlador por convención

Material complementario

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

RAILS GUIDES



Models

Active Record Basics

Active Record Migrations

Active Record Validations

Active Record Callbacks

Active Record Associations

Active Record Query Interface

Views

Layouts and Rendering in Rails

Action View Form Helpers

Controllers

Action Controller Overview

Rails Routing from the Outside In