Clase 07:
Ruby on Rails

profesor: **Patricio López Juri** { patricio@lopezjuri.com }
créditos: 10
horario: J:7-8
sala: H3

Supuesto que hago yo en este curso:

Ustedes son personas inteligentes y pueden aprender y deducir reglas lógicas en base a ejemplos.

Repaso

- ¿Cuál es la diferencia entre Ruby, Sinatra y Ruby on Rails?
- ¿Para qué nos sirven los Status Code HTTP? ¿Qué tipo de errores hay?

RUBY ON RAILS

Ruby on Rails es un framework en Ruby

- Es mucho más grande y completo que Sinatra
- Es mucho más complejo
- Automatiza muchas de las tareas
- Usado ampliamente en la industria y Startups

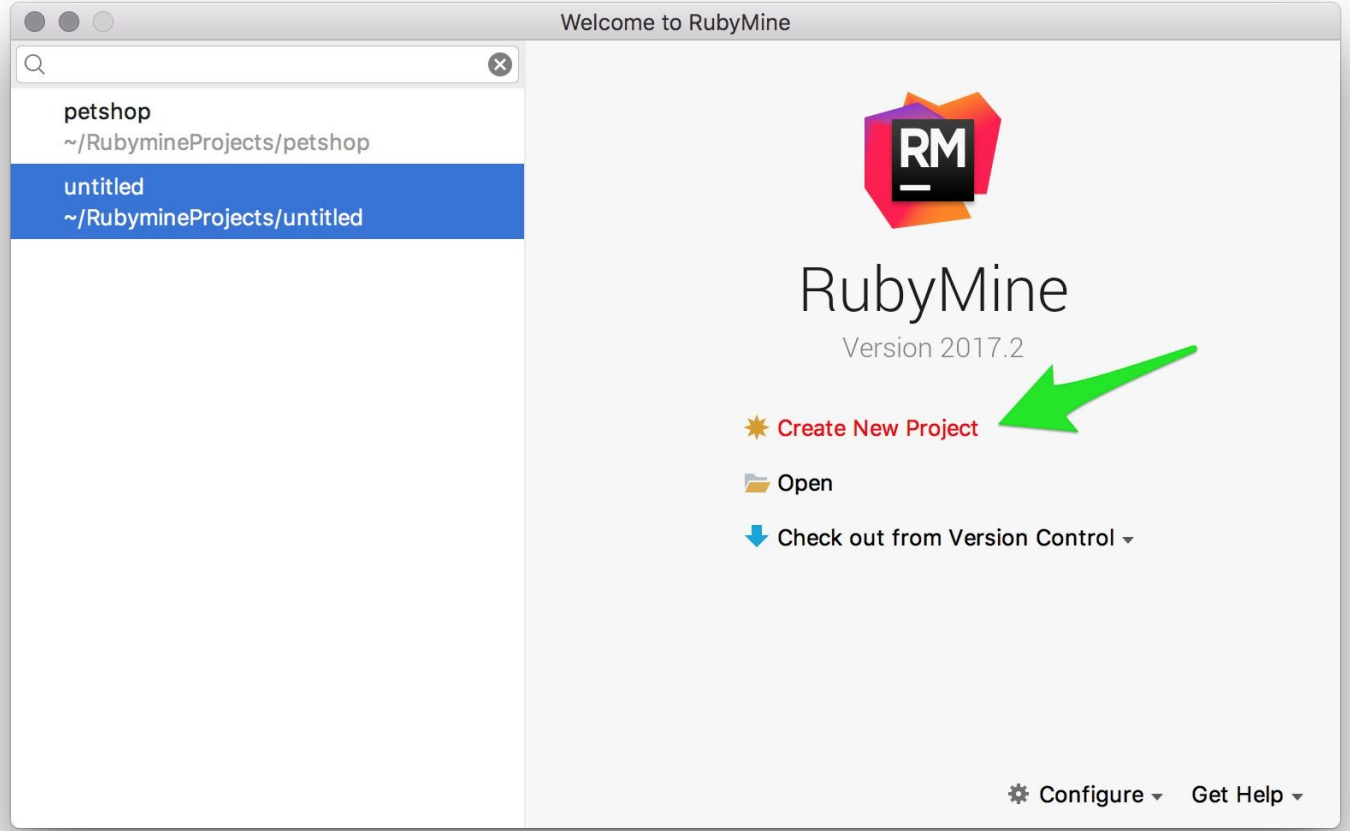
Ya no usaremos Visual Studio Code

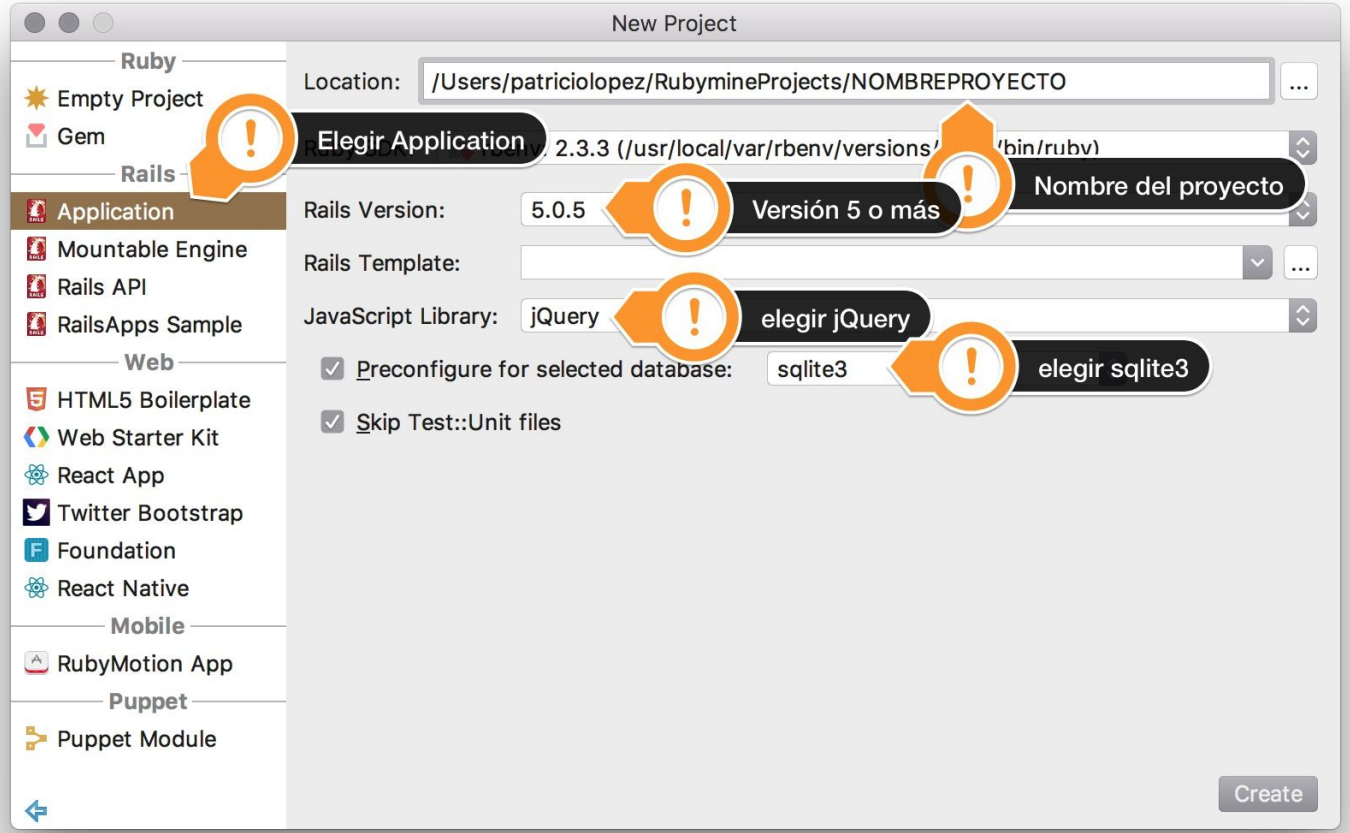
Usaremos RubyMine

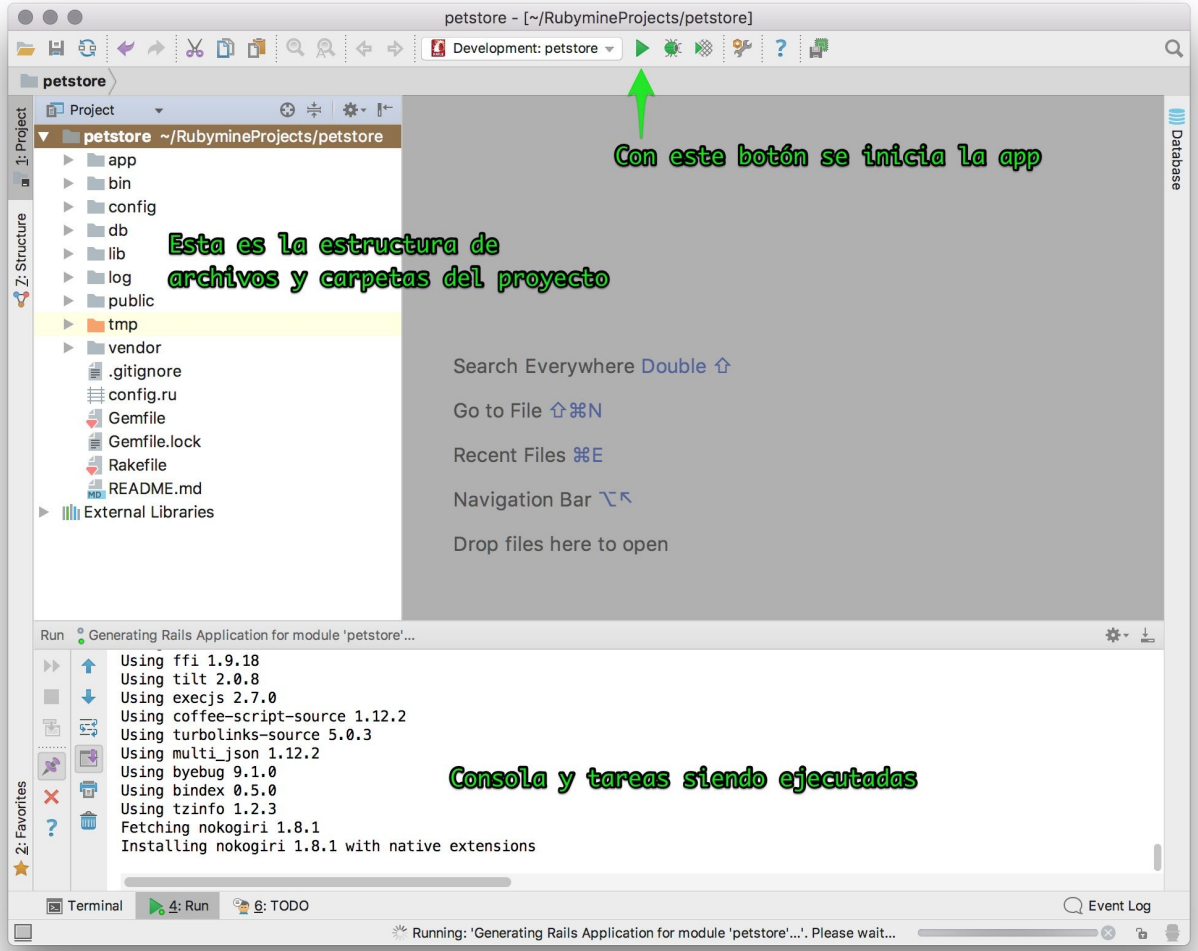
Se puede bajar con la Cuenta UC desde:

https://www.jetbrains.com/shop/eform/students

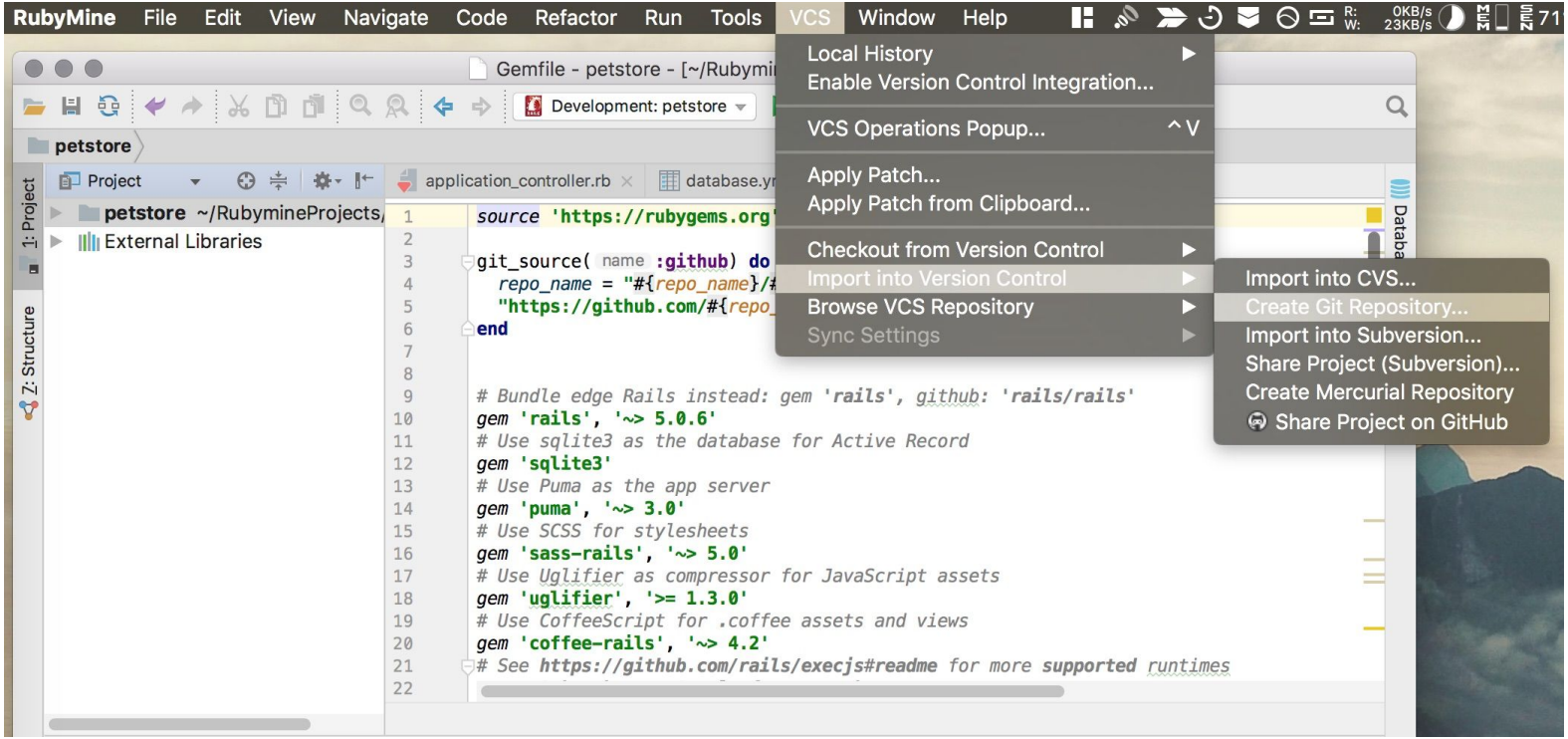
PASOS PARA CREAR UN PROYECTO


Welcome to RubyMine
petshop
~/RubymineProjects/petshop
untitled
~/RubymineProjects/untitled
RM
RubyMine
Version 2017.2
Create New Project
Open
Check out from Version Control
Configure
Get Help

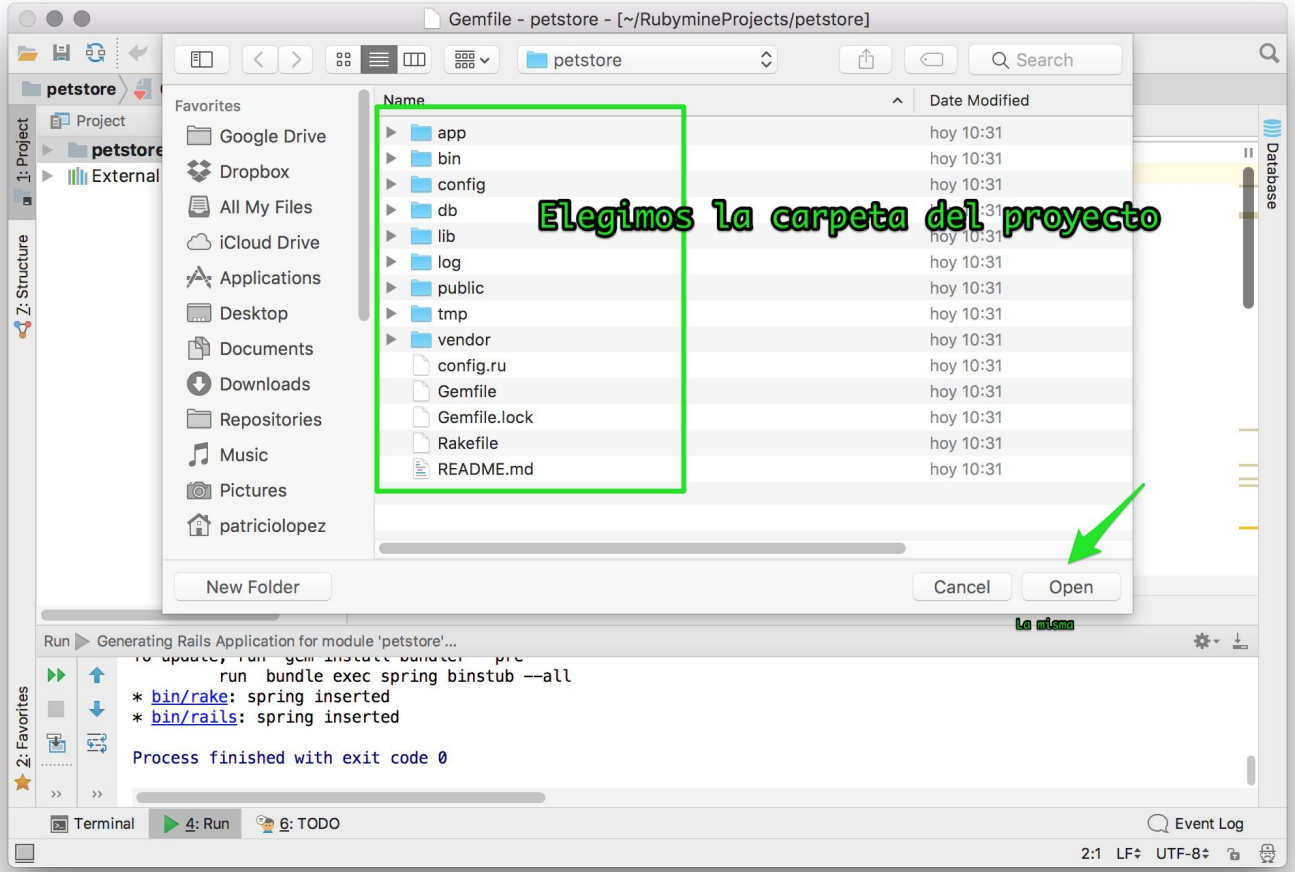

New Project
Ruby
Empty Project
Gem
Rails
Application
Mountable Engine
Rails API
RailsApps Sample
Web
HTML5 Boilerplate
Web Starter Kit
React App
Twitter Bootstrap
Foundation
React Native
Mobile
RubyMotion App
Puppet
Puppet Module
Location:
/Users/patriciolopez/RubymineProjects/NOMBREPROYECTO
Elegir Application
Nombre del proyecto
Rails Version:
5.0.5
Versión 5 o más
Rails Template:
JavaScript Library:
jQuery
elegir jQuery
Preconfigure for selected database:
sqlite3
elegir sqlite3
Skip Test::Unit files
Create


petstore - [~/RubymineProjects/petstore]
Development: petstore
petstore
Con este botón se inicia la app
Project
petstore ~/RubymineProjects/petstore
app
bin
config
db
lib
log
public
tmp
vendor
.gitignore
config.ru
Gemfile
Gemfile.lock
Rakefile
README.md
External Libraries
Esta es la estructura de archivos y carpetas del proyecto
Search Everywhere Double ⇧
Go to File ⇧⌘N
Recent Files ⌘E
Navigation Bar ⌥↖
Drop files here to open
Run
Generating Rails Application for module 'petstore'...
Using ffi 1.9.18
Using tilt 2.0.8
Using execjs 2.7.0
Using coffee-script-source 1.12.2
Using turbolinks-source 5.0.3
Using multi_json 1.12.2
Using byebug 9.1.0
Using bindex 0.5.0
Using tzinfo 1.2.3
Fetching nokogiri 1.8.1
Installing nokogiri 1.8.1 with native extensions
Consola y tareas siendo ejecutadas
Terminal
4: Run
6: TODO
Event Log
Running: 'Generating Rails Application for module 'petstore'...'. Please wait...
1: Project
7: Structure
2: Favorites
Database

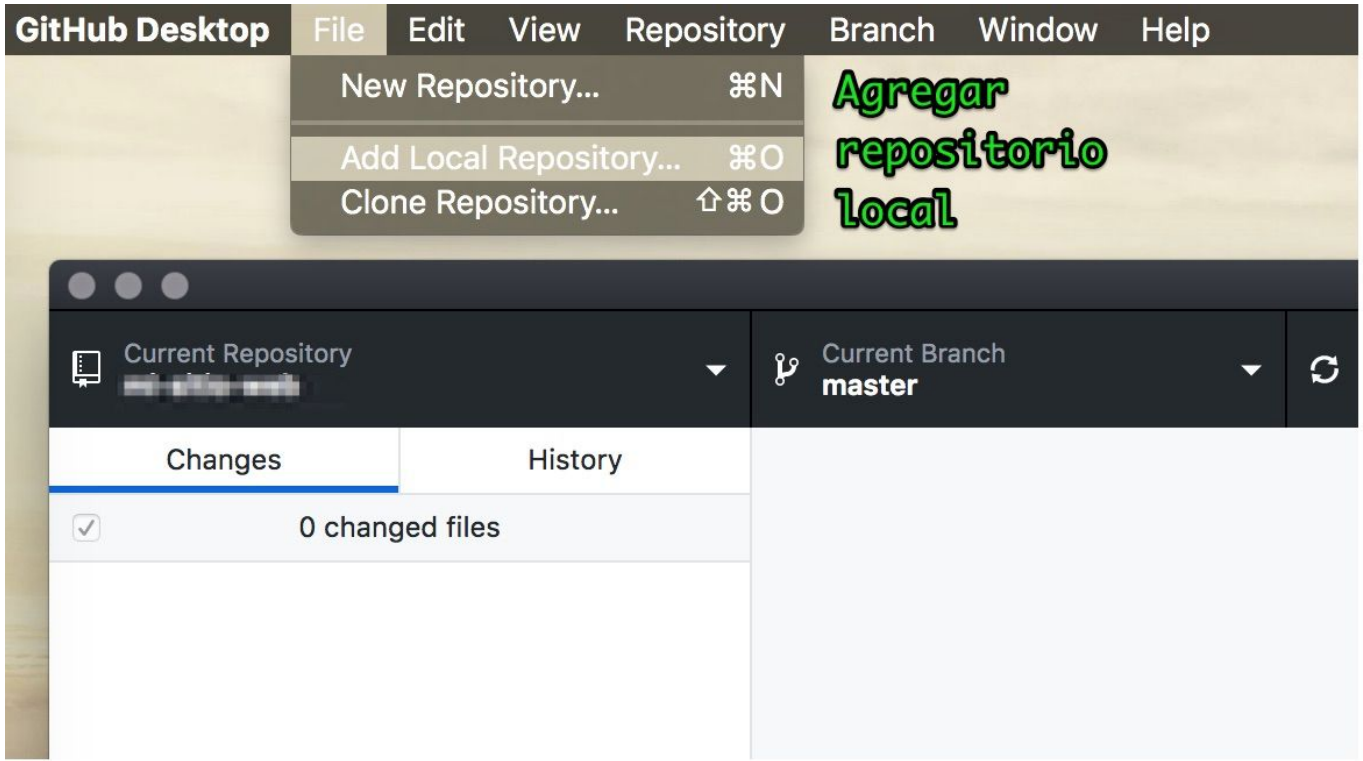
Usemos git para guardar nuestro proyecto

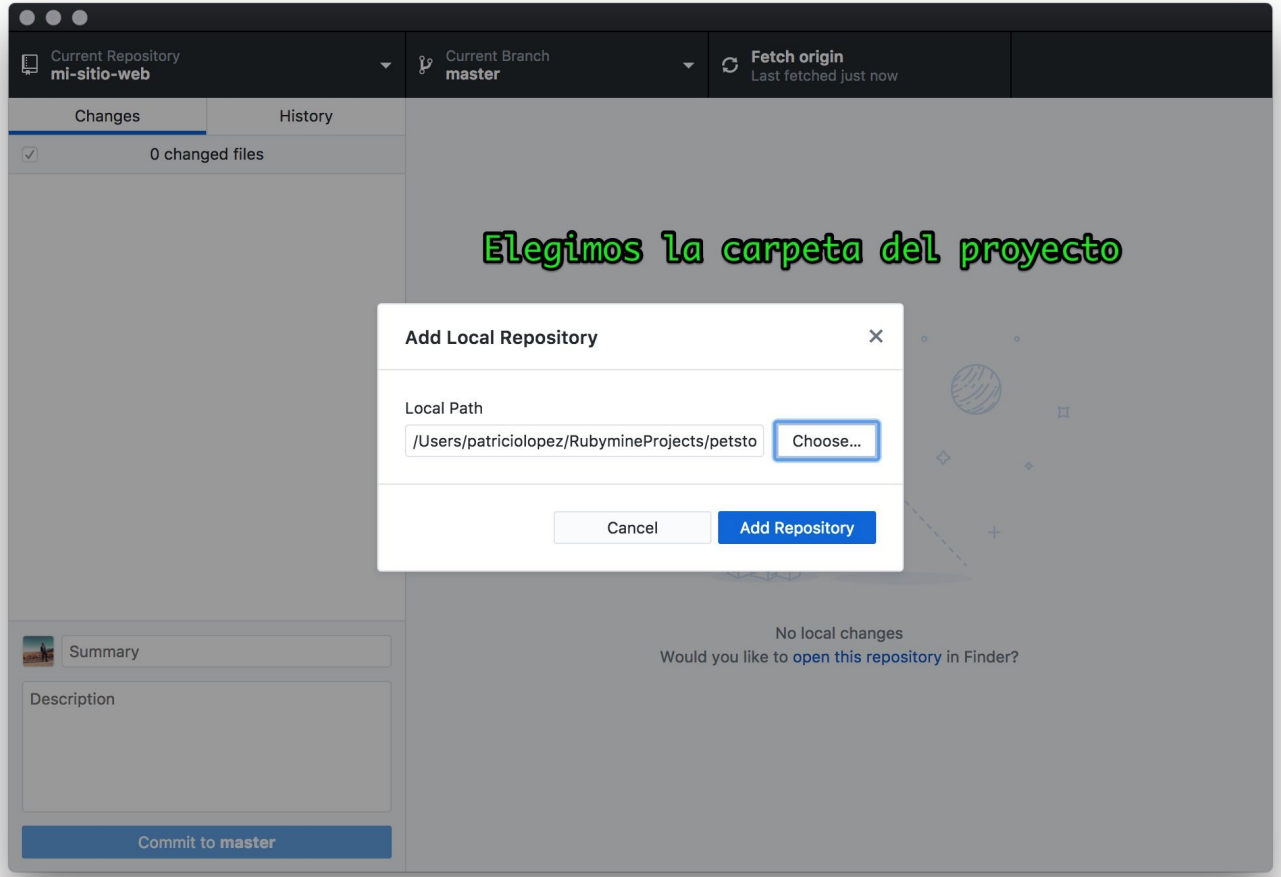


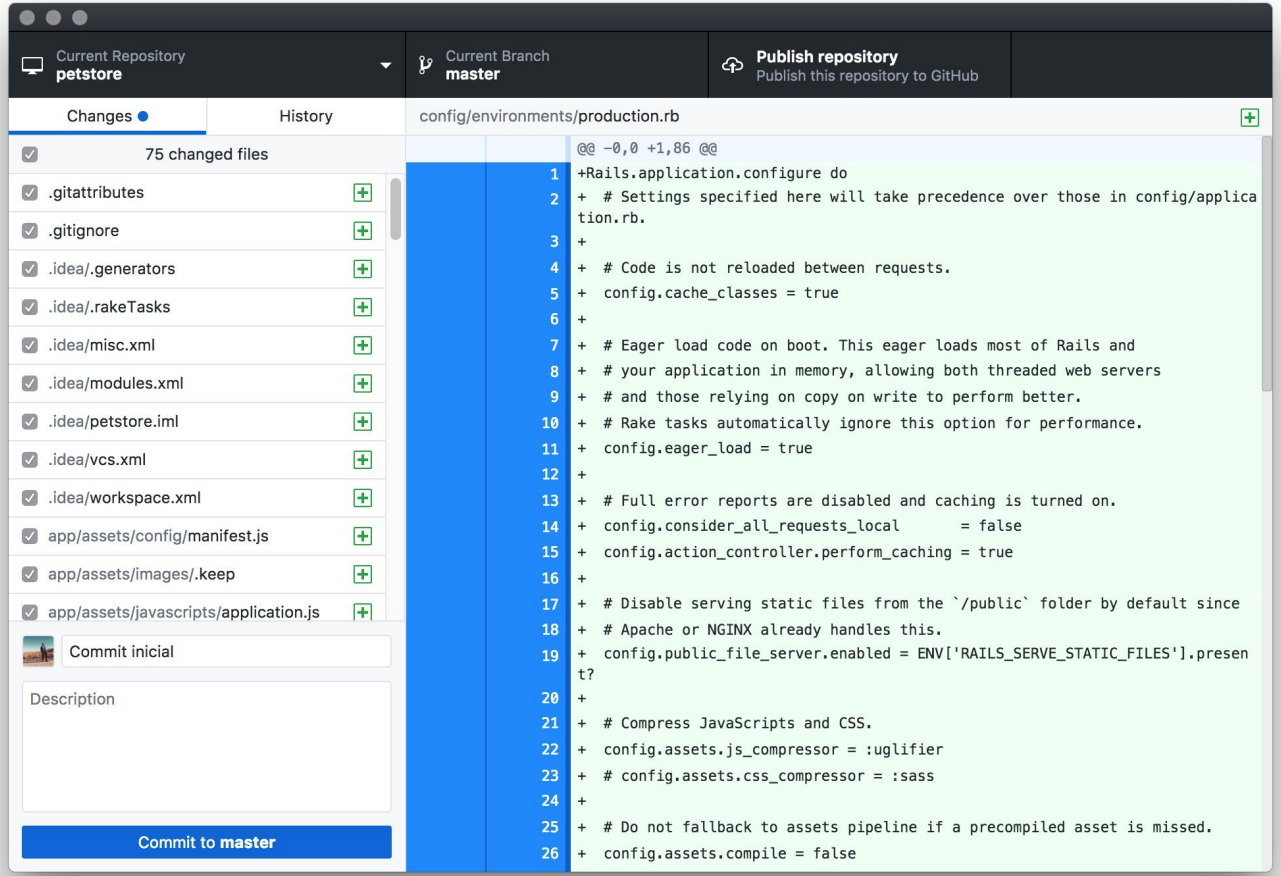
Primero le decimos a RubyMine que queremos usar git:
`VCS -> Import into Version Control -> Create Git Repository`


Gemfile - petstore - [~/RubymineProjects/petstore]
petstore
Search
Favorites
Google Drive
Dropbox
All My Files
iCloud Drive
Applications
Desktop
Documents
Downloads
Repositories
Music
Pictures
patriciolopez
Name
Date Modified
app
bin
config
db
lib
log
public
tmp
vendor
config.ru
Gemfile
Gemfile.lock
Rakefile
README.md
hoy 10:31
Elegimos la carpeta del proyecto
New Folder
Cancel
Open
La misma
Run
Generating Rails Application for module 'petstore'...
run bundle exec spring binstub --all
* bin/rake: spring inserted
* bin/rails: spring inserted
Process finished with exit code 0
Terminal
4: Run
6: TODO
Event Log
2:1
LF
UTF-8
1: Project
Z: Structure
2: Favorites
Database

En la app de Github:

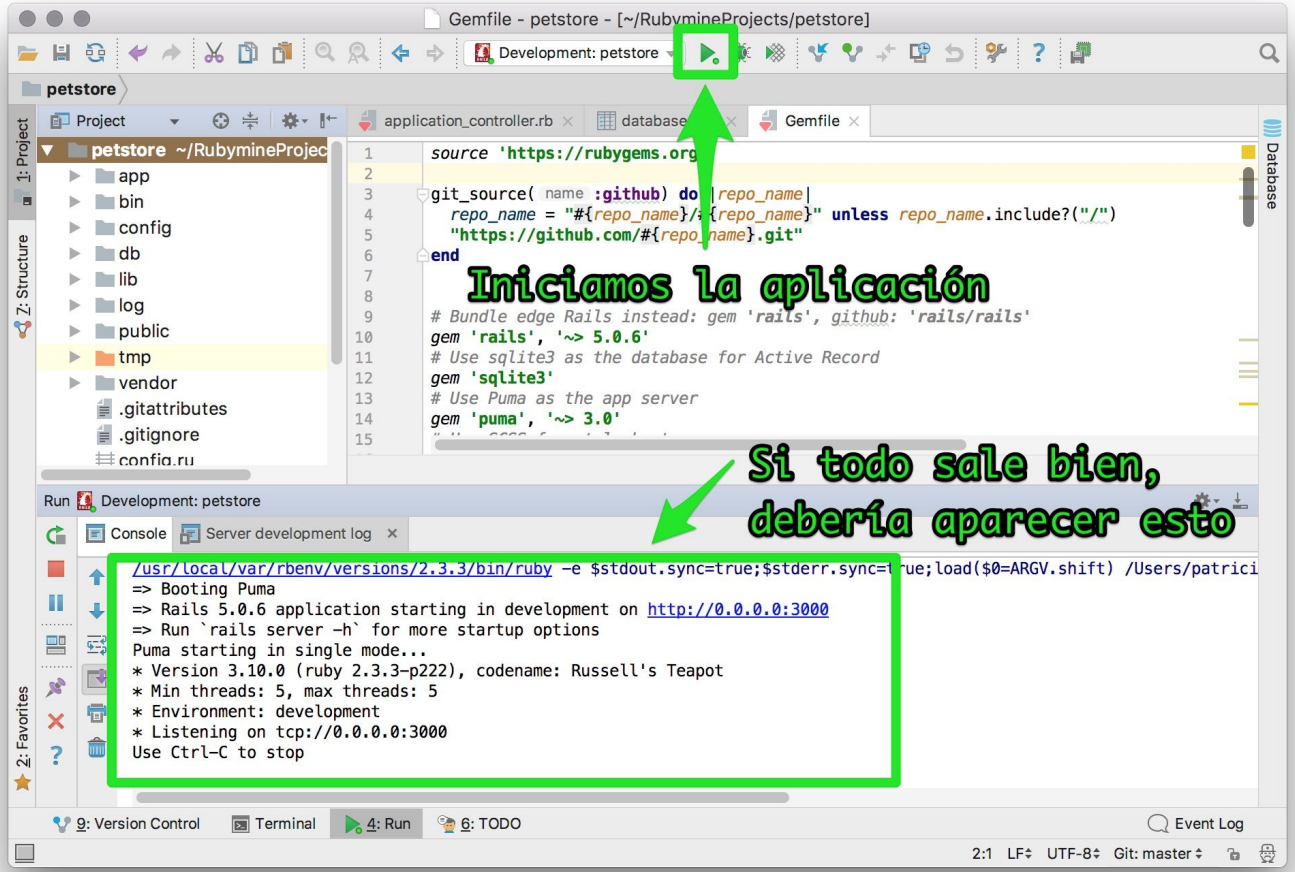


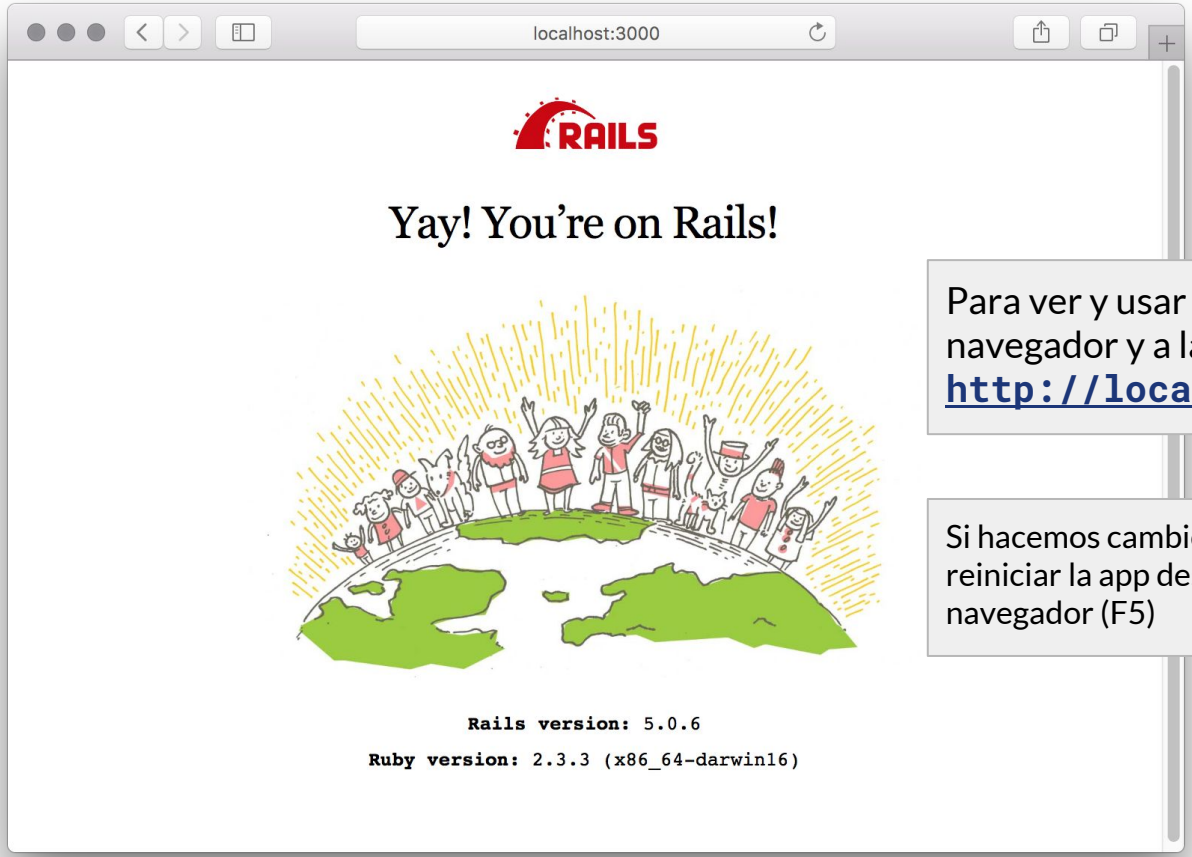

Current Repository
mi-sitio-web
Current Branch
master
Fetch origin
Last fetched just now
Changes
History
0 changed files
Elegimos la carpeta del proyecto
Add Local Repository
Local Path
/Users/patriciolopez/RubymineProjects/petsto
Choose...
Cancel
Add Repository
Summary
Description
Commit to master
No local changes
Would you like to open this repository in Finder?



Ahora podemos usar Git como siempre

Esto es importante porque si vamos a probar cosas en Rails y no resultan, **podemos usar Git para revertir los cambios.**


Gemfile - petstore - [~/RubymineProjects/petstore]
Development: petstore
petstore
Project
application_controller.rb
database
Gemfile
petstore ~/RubymineProjec
app
bin
config
db
lib
log
public
tmp
vendor
.gitattributes
.gitignore
config.ru
source 'https://rubygems.org'
git_source(name :github) do |repo_name|
repo_name = "#{repo_name}/#{repo_name}" unless repo_name.include?("/")
"https://github.com/#{repo_name}.git"
end
Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '~> 5.0.6'
Use sqlite3 as the database for Active Record
gem 'sqlite3'
Use Puma as the app server
gem 'puma', '~> 3.0'
Iniciamos la aplicación
Si todo sale bien,
debería aparecer esto
Run
Development: petstore
Console
Server development log
/usr/local/var/rbenv/versions/2.3.3/bin/ruby -e $stdout.sync=true;$stderr.sync=true;load($0=ARGV.shift) /Users/patrici
=> Booting Puma
=> Rails 5.0.6 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.10.0 (ruby 2.3.3-p222), codename: Russell's Teapot
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop
9: Version Control
Terminal
4: Run
6: TODO
Event Log
2:1
LF
UTF-8
Git: master
1: Project
Z: Structure
2: Favorites
Database



Para ver y usar nuestra app vamos al navegador y a la URL: **http://localhost:3000**

Si hacemos cambios en el código, no tenemos que reiniciar la app desde RubyMine, solo refrescar el navegador (F5)

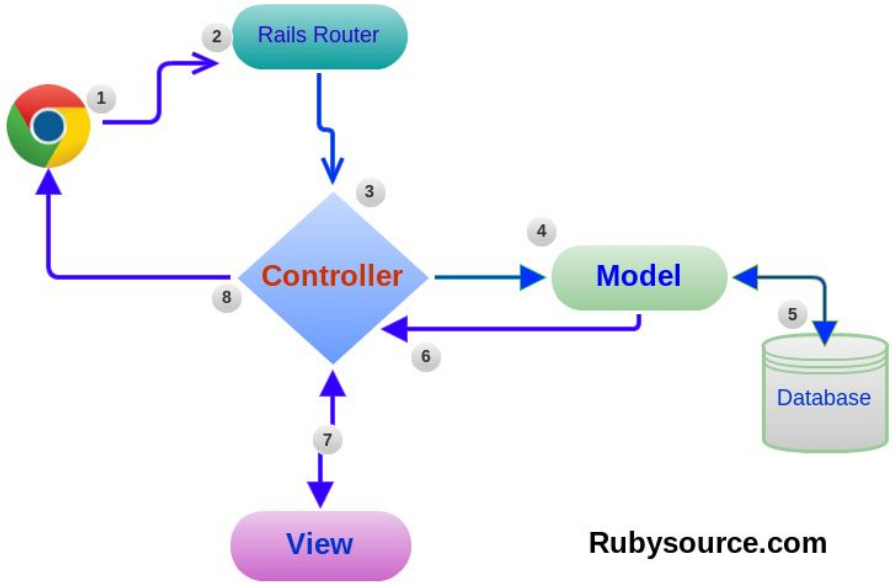
ahora

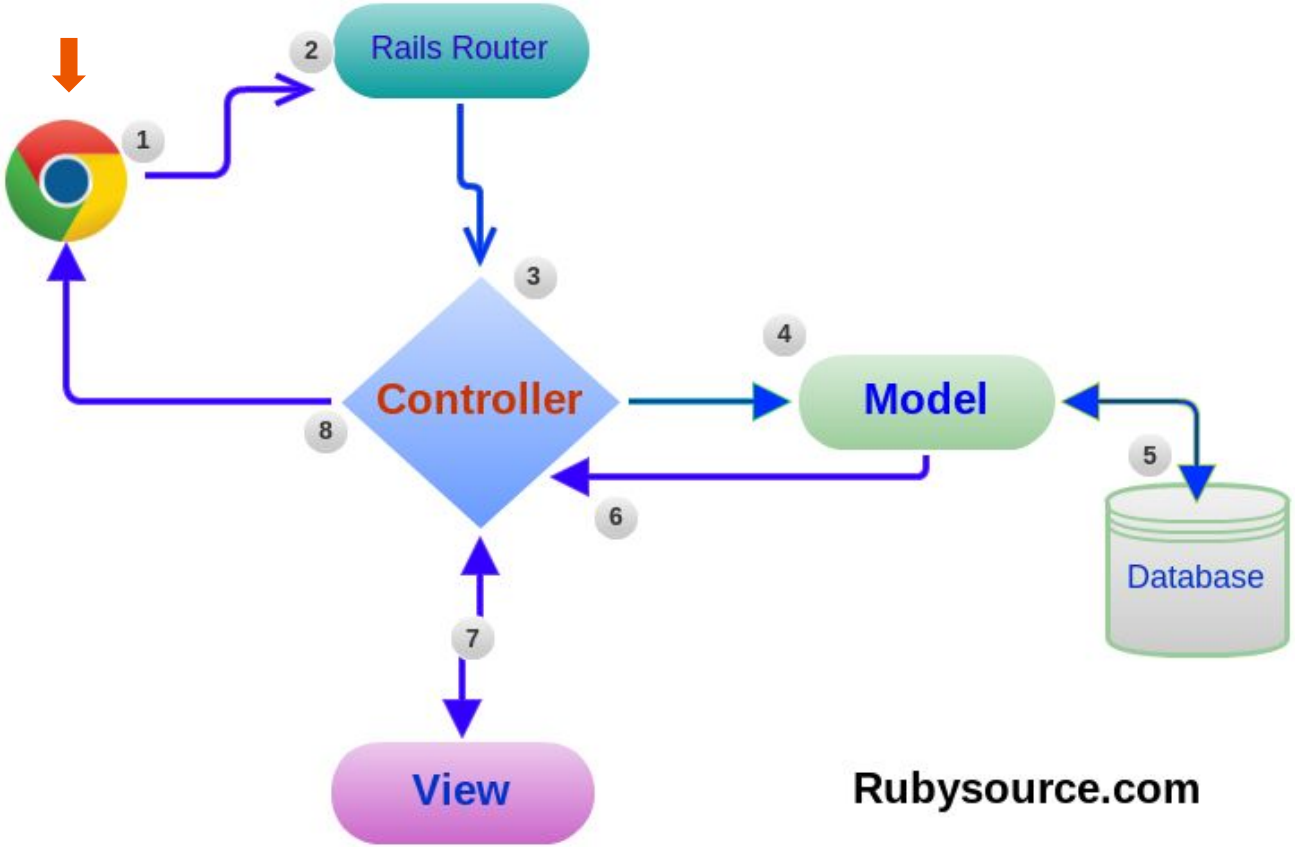
RAILS EN DETALLE

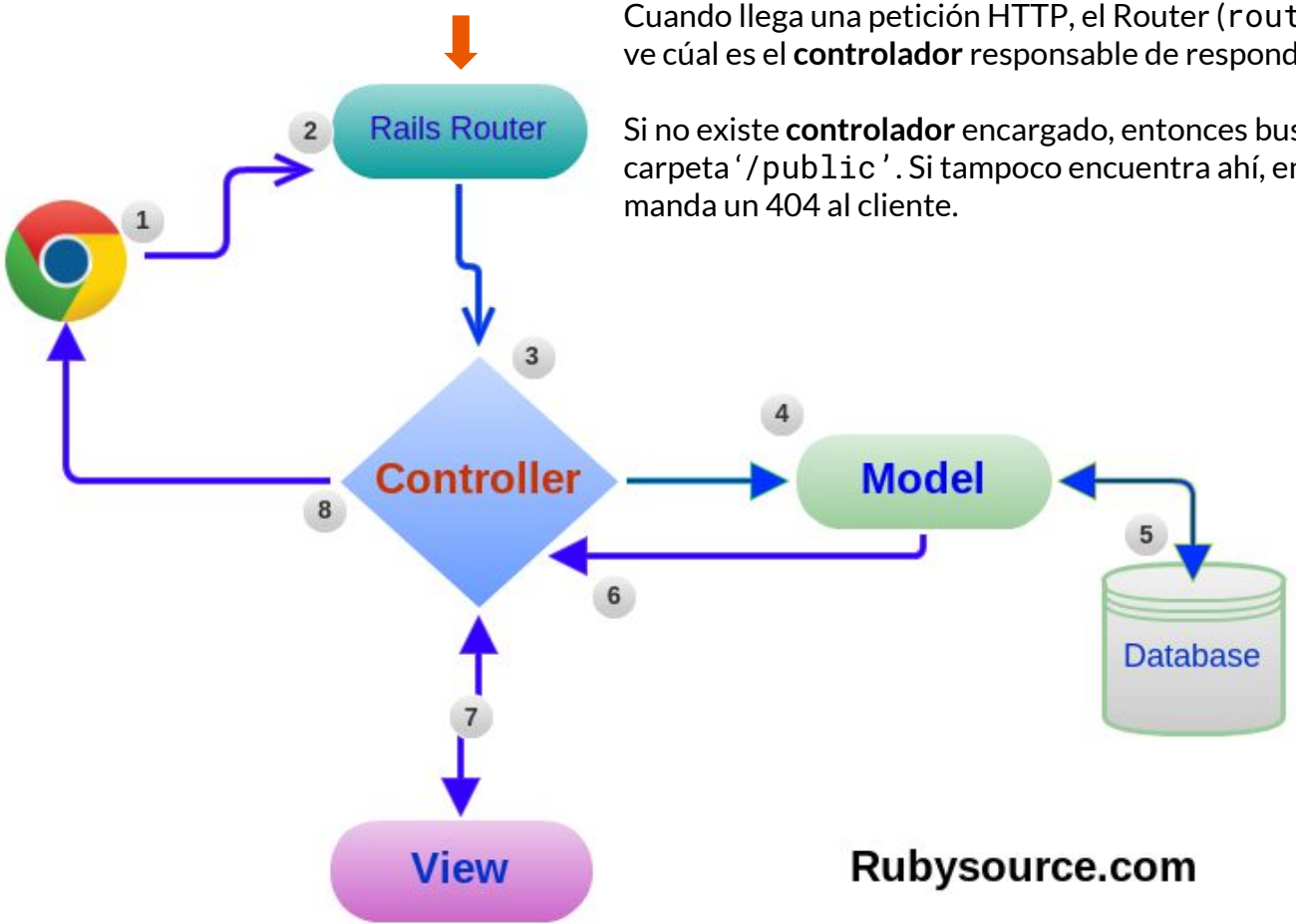
Las partes importantes de Rails

En orden de ejecución:

1. *Router*
2. *Controller* (controlador)
3. *Model* (modelo)
4. *View* (vista)



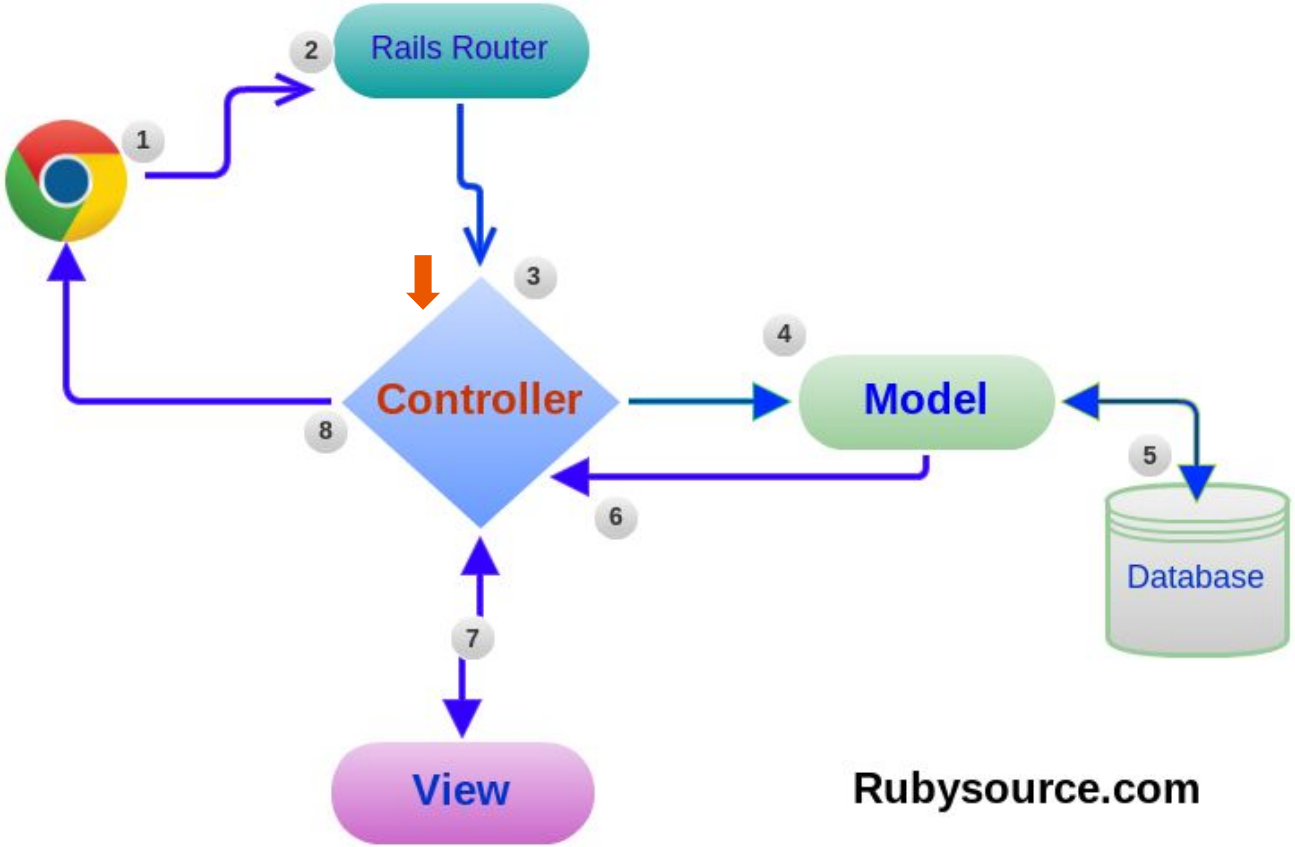

Rails Router
1
2
3
4
5
6
7
8
Controller
Model
Database
View
Rubysource.com



Cuando llega una petición HTTP, el Router (`routes.rb`) ve cúal es el **controlador** responsable de responder.

Si no existe **controlador** encargado, entonces busca en la carpeta '`/public`'. Si tampoco encuentra ahí, entonces manda un 404 al cliente.

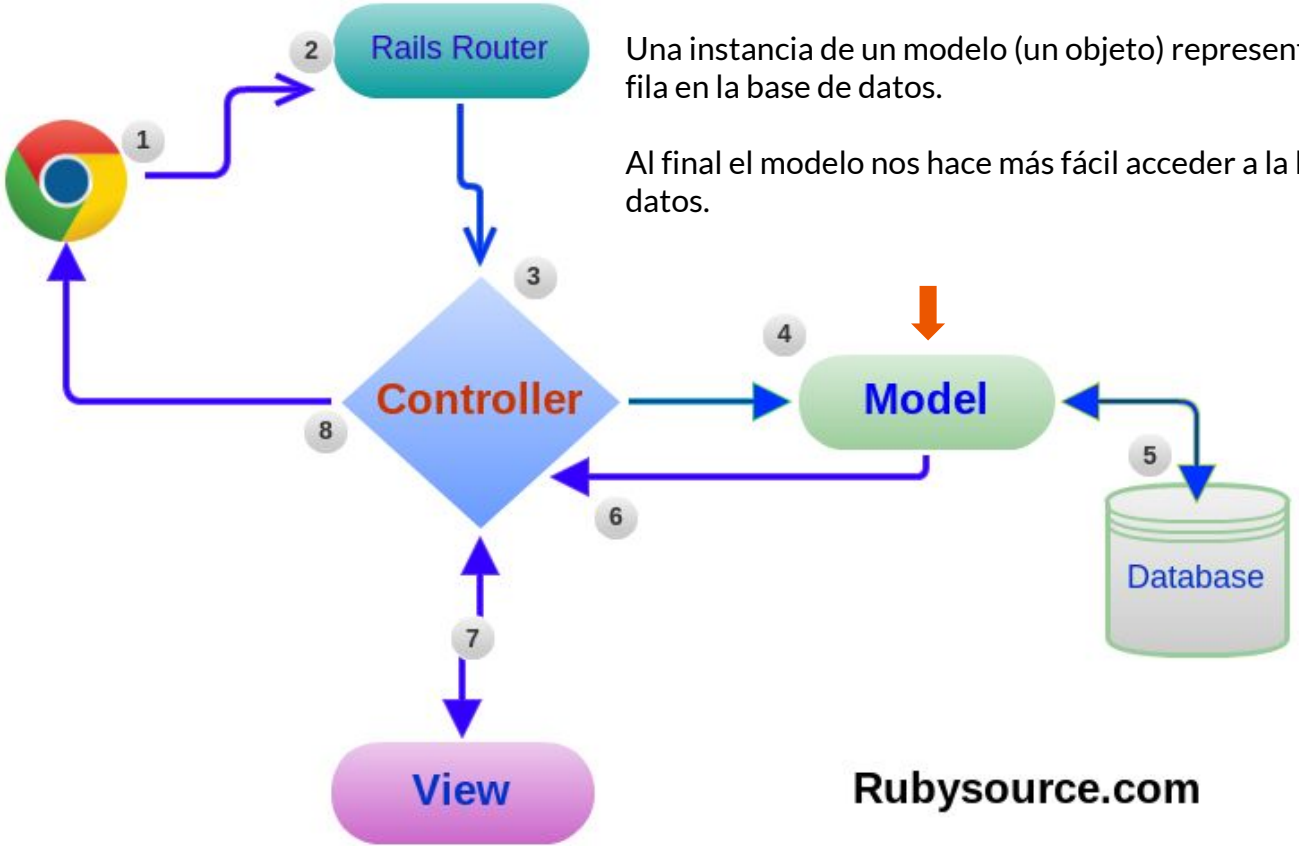
En controlador responsable procesa la petición y carga los datos necesarios desde los **modelos**.



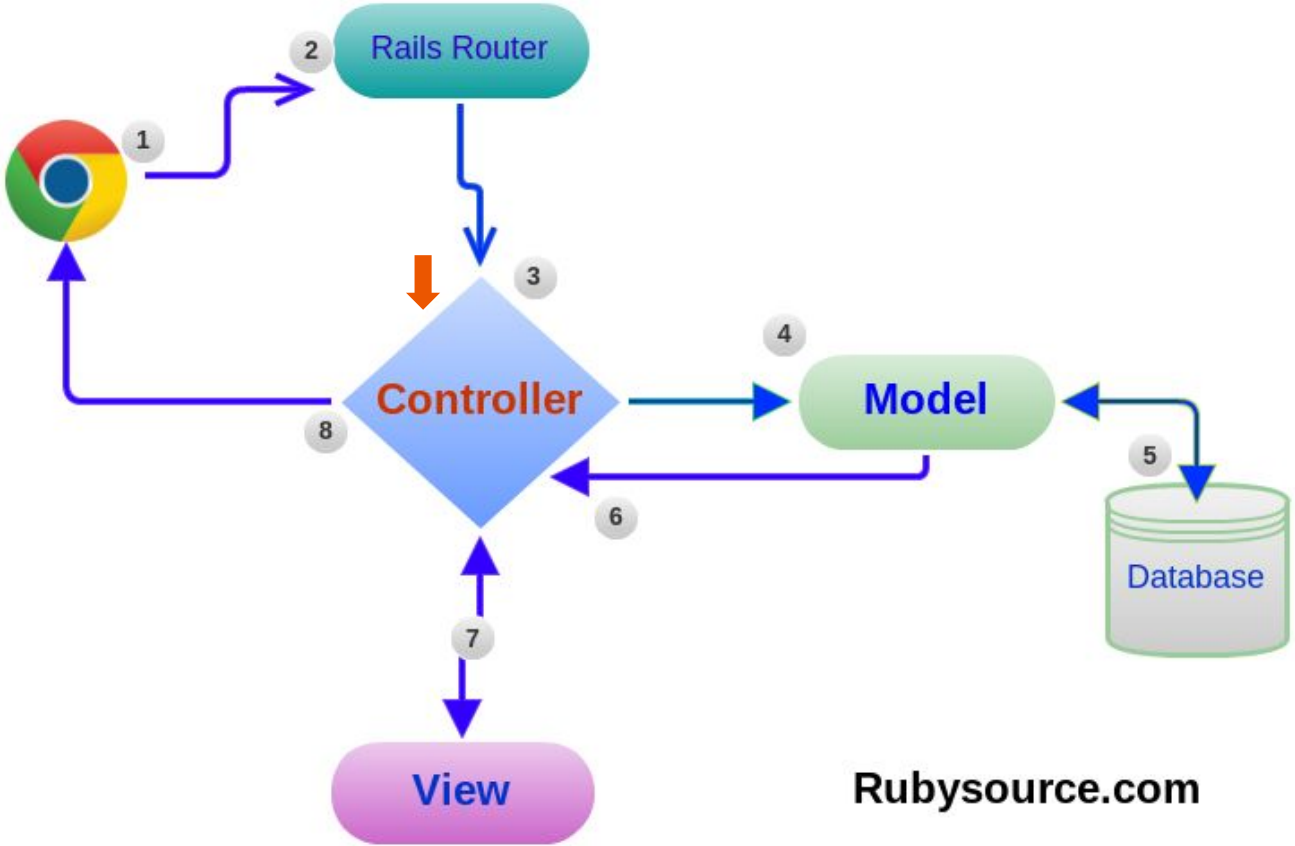
Los **modelos** definen la estructura de la información en la base de datos.

Una instancia de un modelo (un objeto) representa una fila en la base de datos.

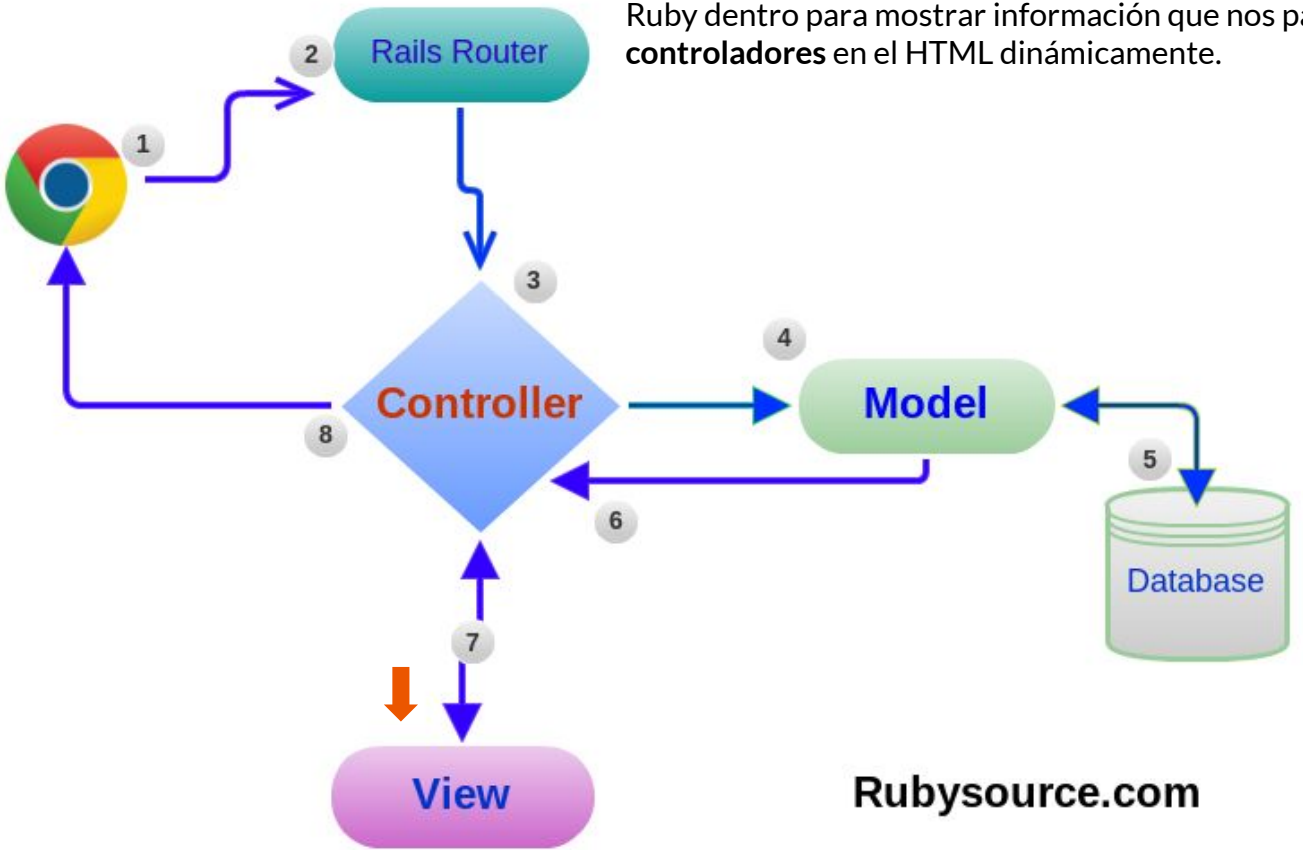
Al final el modelo nos hace más fácil acceder a la base de datos.



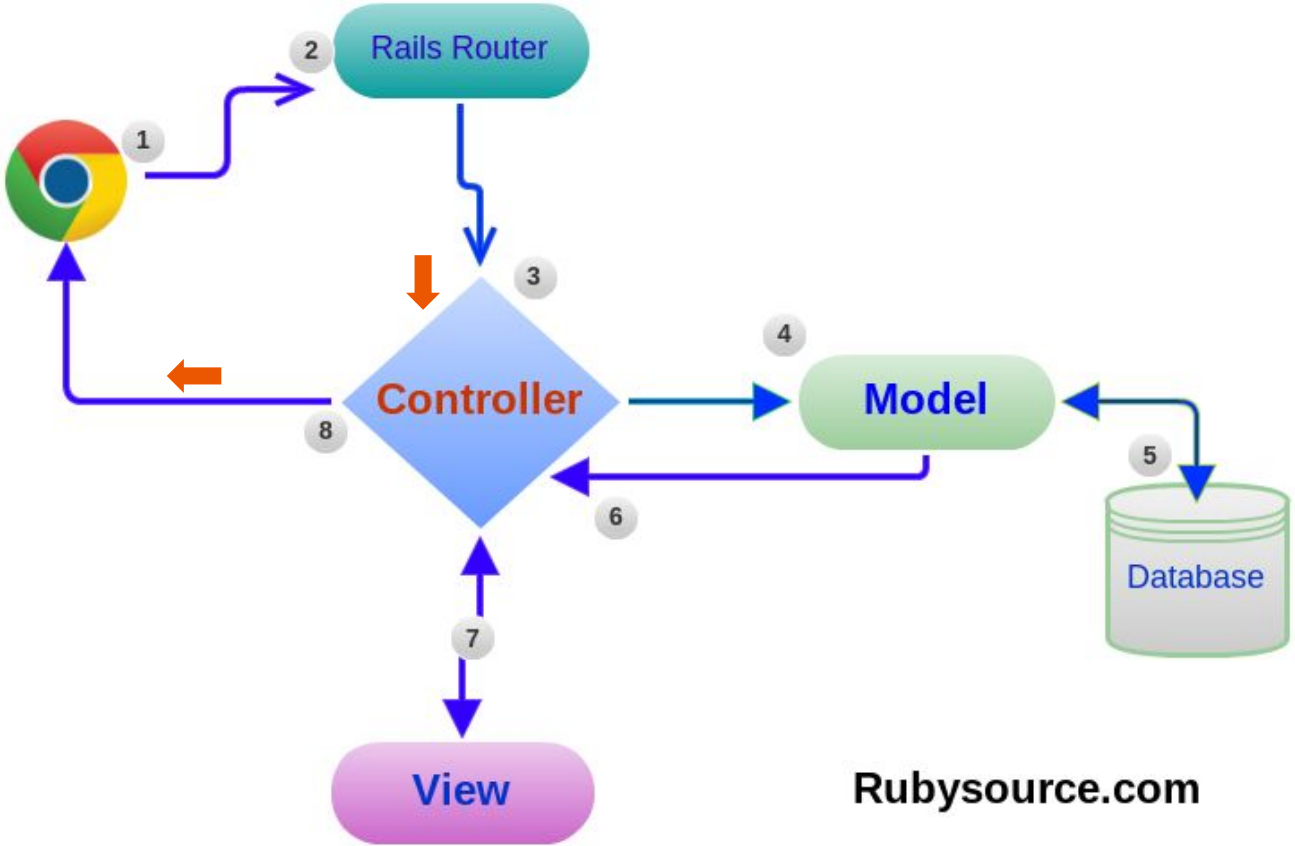
Una vez que el **controlador** tiene los datos cargados desde el **modelo**, los procesa y genera la **vista** (View).



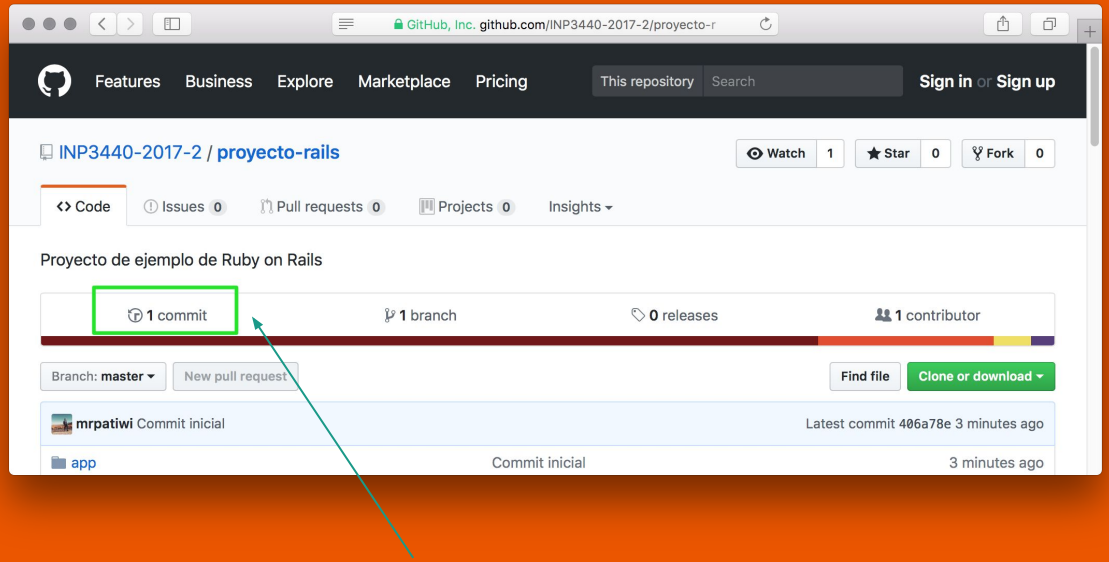
La vista son HTML a los que ya estamos acostumbrados, pero estos **tienen extensión `.erb`** que nos permite usar Ruby dentro para mostrar información que nos pasó los **controladores** en el HTML dinámicamente.



Una vez que la vista está lista (HTML), en controlador la devuelve al cliente para mostrarla al usuario.



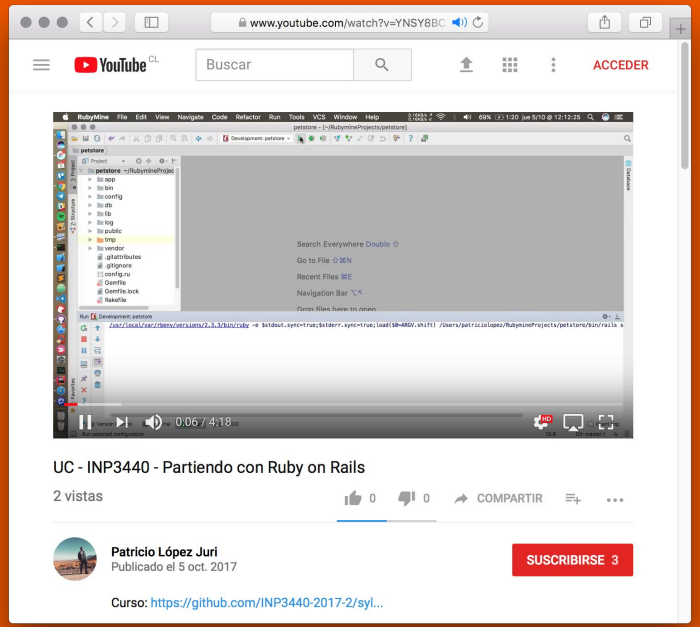
Hice un proyecto de ejemplo en:
https://github.com/INP3440-2017-2/proyecto-rails



Pueden ver paso a paso el código aquí.

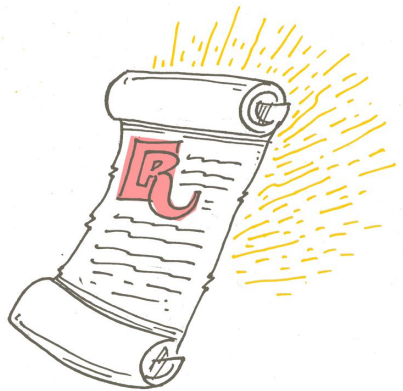
Subí videos a YouTube

https://www.youtube.com/watch?v=YNSY8BC3iG4



Ruby on Rails usa “convención por sobre configuración”

Esto significa que **en vez de definir configuraciones personalizadas para cada cosa**, mejor definamos ciertas convenciones o reglas a seguir.



RAILS se parece al patrón: **MVC: Modelo Vista Controlador**

Un patrón es un diseño que soluciona problemas que se repiten mucho, entonces se estandariza su solución como un “patrón”.

En este caso el patrón MVC ayuda a que la aplicación sea fácil de mantener y establece funciones específicas al modelo, vista y controlador; en vez de mezclar todo el código en un solo gran archivo y crear un monstruo.

No sigue estrictamente el MVC, pero a al gente que no conoce Rails les aclara mucho cuando uno le dice esto.

Tener un patrón es importante.

Muchas aplicaciones no tienen una arquitectura por detrás y se convierten en código que nadie entiende ni quieren modificar por miedo a romperlo.

Empezemos!

Creemos la página de inicio

Los pasos son:

1. Crear una entrada en `routes.rb`
2. Crear un controlador
3. Crear las vistas

Para este caso no necesitamos modelos (todavía).

VER VIDEO EN YOUTUBE

https://www.youtube.com/watch?v=YNSY8BC3iG4

Router, el archivo `config/routes.rb`

Este define las reglas de las request HTTP que recibe la app en Rails.

```ruby
Rails.application.routes.draw do
  get '/', to: 'welcome#index'
end
```

El controlador:

`app/controllers/welcome_controller.rb`

Responde a la request HTTP

```
class WelcomeController < ApplicationController
 def index
   @time = Time.now
 end
end
```

Vista:

app/views/welcome/index.html.erb

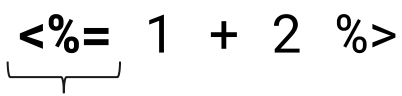
Generamos HTML y con contenido dinámico usando Ruby y ERB. ERB es para generar “templates”.

```
<div>
  <h1>
    Bienvenido!
  </h1>
  <p>
    La hora actual es: <%= @time %>
  </p>
</div>
```

.html.erb

Permite usar lo que ya sabemos de HTML pero ahora lo podemos mezclar con Ruby.

Se usa la siguiente marcación para incluir Ruby:



El resultado es “impreso” en el HTML

El resultado no aparece

¿Entonces para qué sirve el <% %>?

Para poder código de control de flujo:

En este ejemplo mostrará el nombre del usuario si existe, en caso contrario avisará que no ha iniciado sesión.

```
<div>
  <h1>
    Bienvenido!
  </h1>
  <% @user.present? %>
    <h2>Hola <%= @user.name %></h2>
  <% else %>
    <h2>No has iniciado sesión</h2>
  <% end %>
</div>
```

También podemos iterar sobre listas

```
<div>
  <h1>
    Noticias:
  </h1>
  <% @noticias.each do |n| %>
    <li>Título: <%= n.titulo %></li>
  <% end %>
</div>
```

Rails nos ayuda con “generadores”

Estos nos generan código que uno suele repetir una y otra vez.

```
rails g scaffold Pet name:string age:integer
```