



Rails

- La necesidad de un framework está ya establecida
 - construir una aplicación Web es complejo
 - un buen framework equivale a partir con un porcentaje significativo de la aplicación ya escrito
 - se genera un código más ordenado y más fácil de modificar
 - características deseables
 - open source, cross platform
 - database abstraction layer
 - promueve arquitectura (MVC)

Características de Rails

- Creado en 2004 por David Heinemeier Hansson
- Para desarrollo en Ruby
- Extraído de desarrollo de aplicación real (Basecamp)
- Actualmente en versión 6.0 (reciente)
- Uso muy efectivo de las convenciones (mucho menos esfuerzo de configuración)
- Rails funciona muy bien con desarrollo ágil

Principios

- DRY (don't repeat yourself)
 - menos esfuerzo y menos errores
- Convention over Configuration
 - se usan fuertemente convenciones de nombres
 - solo si uno no las acepta debe configurar
- MVC - model-view-controller architecture

La fuente de la Verdad

More at rubyonrails.org: [Blog](#) | [Guides](#) | [API](#) | [Ask for help](#) | [Contribute on GitHub](#)



Home [Guides Index](#) ▾ Contribute Credits

Getting Started with Rails

This guide covers getting up and running with Ruby on Rails.

After reading this guide, you will know:

- ✓ How to install Rails, create a new Rails application, and connect your application to a database.
- ✓ The general layout of a Rails application.
- ✓ The basic principles of MVC (Model, View, Controller) and RESTful design.
- ✓ How to quickly generate the starting pieces of a Rails application.

 **Chapters**

1. [Guide Assumptions](#)
2. [What is Rails?](#)
3. [Creating a New Rails Project](#)
 - [Installing Rails](#)
 - [Creating the Blog Application](#)
4. [Hello, Rails!](#)
 - [Starting up the Web Server](#)
 - [Say "Hello", Rails](#)
 - [Setting the Application Home Page](#)
5. [Getting Up and Running](#)
 - [Laying down the groundwork](#)
 - [The first form](#)
 - [Creating articles](#)
 - [Creating the Article model](#)
 - [Running a Migration](#)
 - [Saving data in the controller](#)
 - [Showing Articles](#)
 - [Listing all articles](#)
 - [Adding links](#)

1 Guide Assumptions

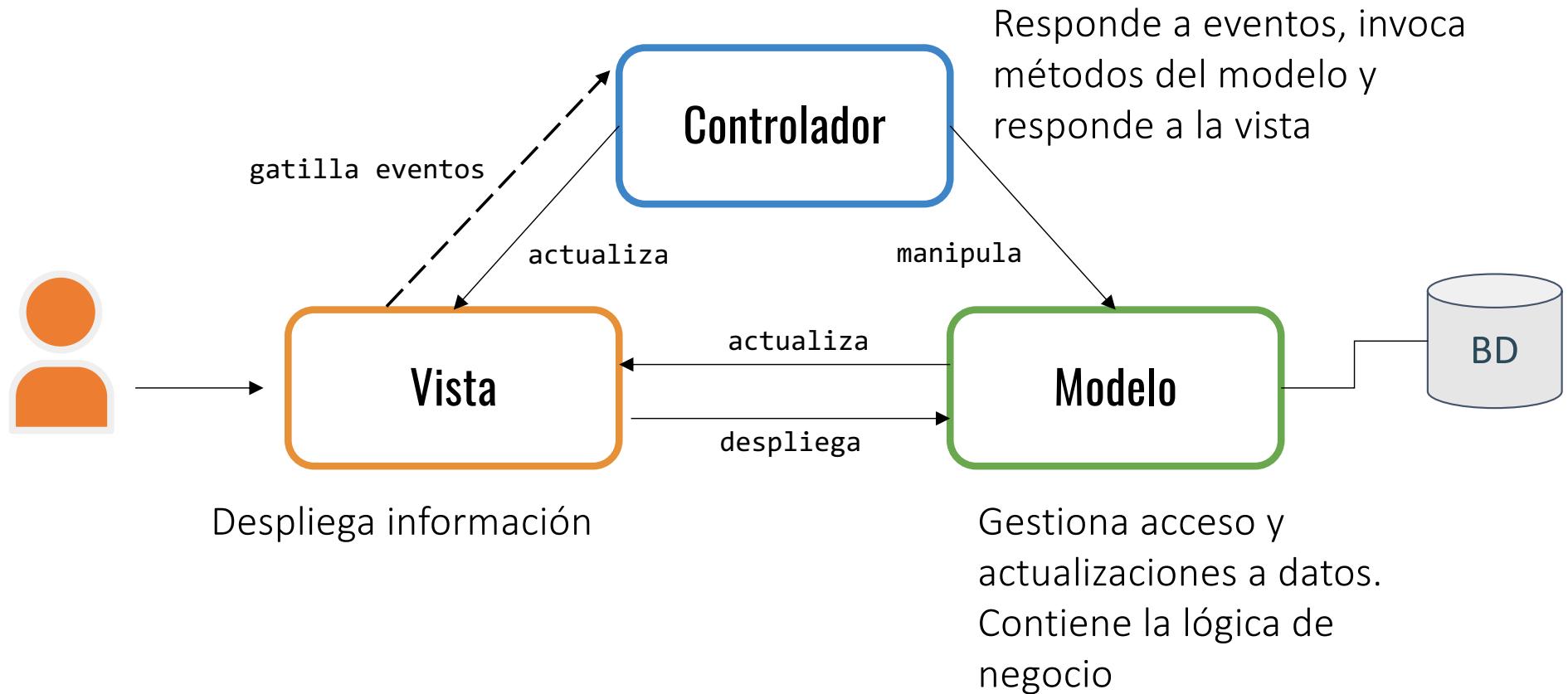
This guide is designed for beginners who want to get started with a Rails application from scratch. It does not assume that you have any prior experience with Rails.

Rails is a web application framework running on the Ruby programming language. If you have no prior experience with Ruby, you will find a very steep learning curve diving straight into Rails. There are several curated lists of online resources for learning Ruby:

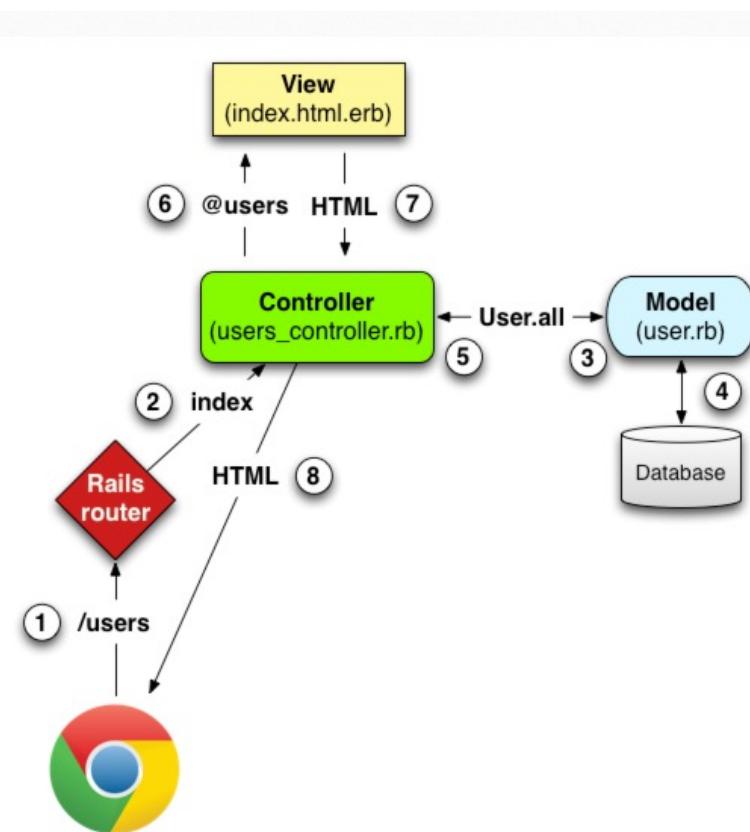
- [Official Ruby Programming Language website](#)
- [List of Free Programming Books](#)

Be aware that some resources, while still excellent, cover versions of Ruby as old as 1.6, and commonly 1.8, and will not include some syntax that you will see in day-to-day development with Rails.

MVC : Modelo Vista Controlador



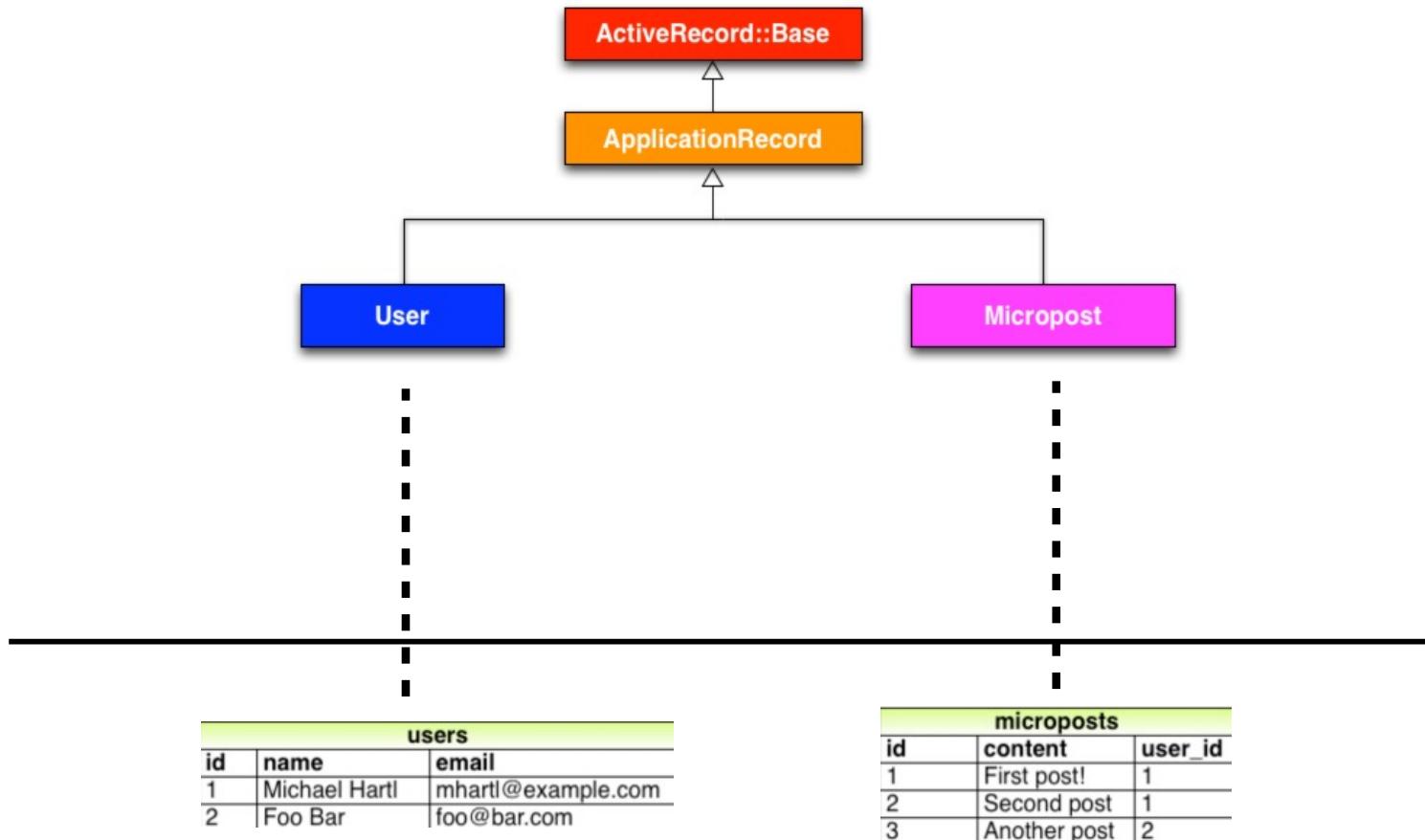
MVC en acción



Modelos

- Los modelos representan a la base de datos
- Cada tabla tiene su contrapartida en una clase del modelo
- Las clases asociadas a tablas heredan de superclase Active Record
 - métodos para agregar, eliminar y buscar instancias
- Incluye reglas de validación y reglas de negocio

ActiveRecord



Vistas

- Representan la parte visible de la aplicación
- Son responsables de presentar visualmente el estado de los objetos del modelo
- A través de templates que contienen mayormente markup (HTML)
- Puede incluir Ruby embebido (erb) para injectar en ella el contenido de los modelos
- No debe haber ninguna interacción con el modelo

Más sobre Vistas

- archivos .html.erb
- principalmente html pero incluyen elementos dinámicos que deben ser resueltos (por eso erb) antes de ser devuelto al browser
 - contenido de variables de instancia @var
 - aplicación de helpers (<%=>
 - built-in (rails)
 - custom

Controladores

- Son los responsables de producir la vista que será enviada al browser
- Aceptan solicitudes (http), procesan , interactúan con modelo y finalmente renderean la vista que será enviada de regreso
- Un controlador por cada área de la aplicación
- Por ejemplo un controller para recetas ...

```
class RecipesController < ApplicationController
  def index
    # logic to list all recipes
  end

  def show
    # logic to show a particular recipe
  end

  def create
    # logic to create a new recipe
  end

  def update
    # logic to update a particular recipe
  end

  def destroy
    # logic to delete a particular recipe
  end
```

Facilidades para construir en forma automática

- Construir una aplicación
- Construir el modelo
- Construir controladores

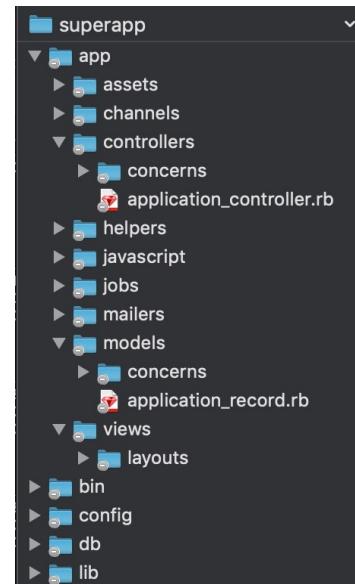
El comando rails permite ...

- crear la estructura completa de la aplicación
- crear controladores, modelos o vistas
- crear scaffolds de recursos (CRUDs completos)
- levantar un server
- etc.

Creación de la Aplicación

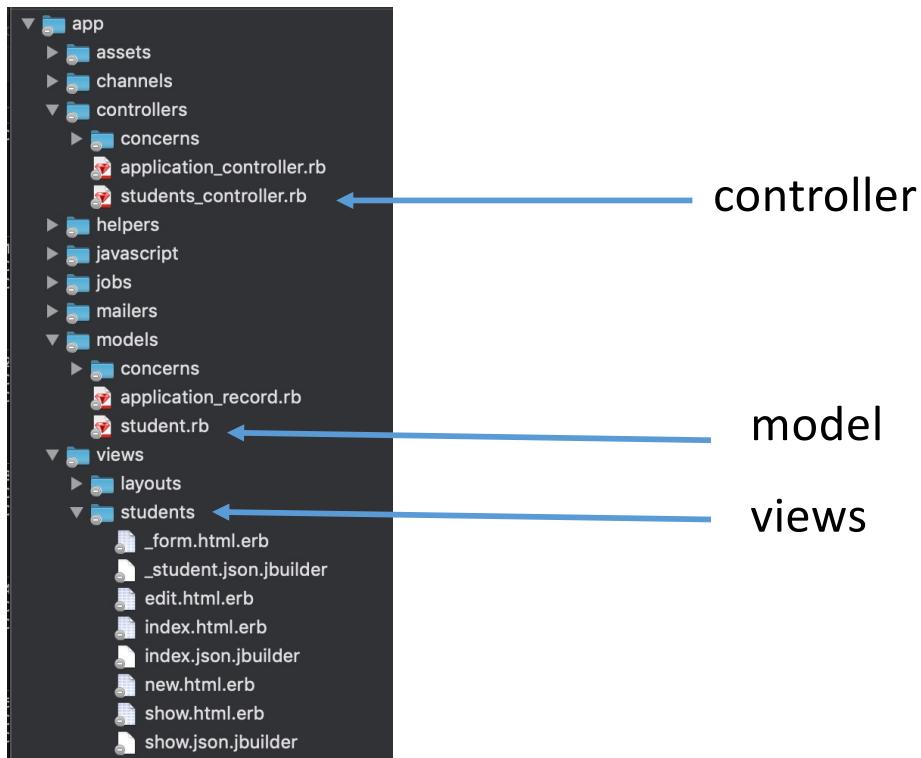
```
$ rails new superapp  
$ cd superapp  
$ ls -l
```

```
total 680  
-rw-r--r-- 1 jnavon staff 1963 Sep 4 22:09 Gemfile  
-rw-r--r-- 1 jnavon staff 5537 Sep 4 22:12 Gemfile.lock  
-rw-r--r-- 1 jnavon staff 374 Sep 4 22:09 README.md  
-rw-r--r-- 1 jnavon staff 227 Sep 4 22:09 Rakefile  
drwxr-xr-x 11 jnavon staff 352 Sep 4 22:09 app  
-rw-r--r-- 1 jnavon staff 1876 Sep 4 22:12 babel.config.js  
drwxr-xr-x 10 jnavon staff 320 Sep 4 22:12 bin  
drwxr-xr-x 18 jnavon staff 576 Sep 4 22:12 config  
-rw-r--r-- 1 jnavon staff 130 Sep 4 22:09 config.ru  
drwxr-xr-x 3 jnavon staff 96 Sep 4 22:09 db  
drwxr-xr-x 4 jnavon staff 128 Sep 4 22:09 lib  
drwxr-xr-x 4 jnavon staff 128 Sep 4 22:12 log  
drwxr-xr-x 777 jnavon staff 24864 Sep 4 22:12 node_modules  
-rw-r--r-- 1 jnavon staff 335 Sep 4 22:12 package.json  
-rw-r--r-- 1 jnavon staff 224 Sep 4 22:12 postcss.config.js  
drwxr-xr-x 9 jnavon staff 288 Sep 4 22:09 public  
drwxr-xr-x 3 jnavon staff 96 Sep 4 22:09 storage  
drwxr-xr-x 12 jnavon staff 384 Sep 4 22:09 test  
drwxr-xr-x 6 jnavon staff 192 Sep 4 22:12 tmp  
drwxr-xr-x 4 jnavon staff 128 Sep 4 22:09 vendor  
-rw-r--r-- 1 jnavon staff 310086 Sep 4 22:12 yarn.lock
```



Scaffolds

```
$ rails generate scaffold Student rut:string name:string
```



Generación de las tablas de la BD

```
$ rails db:migrate  
(base) Tatooine:superapp jnavon$ pwd  
/Users/jnavon/Develop/rails/superapp  
(base) Tatooine:superapp jnavon$ rails db:migrate  
== 20190905022738 CreateStudents: migrating =====  
-- create_table(:students)  
  -> 0.004s  
== 20190905022738 CreateStudents: migrated (0.0047s) =====  
  
(base) Tatooine:superapp jnavon$ █
```

Levantando la App

```
$ rails server  
(base) Tatooine:superapp jnavon$ rails server  
=> Booting Puma  
=> Rails 6.0.0 application starting in development  
=> Run `rails server --help` for more startup options  
Puma starting in single mode...  
* Version 3.12.1 (ruby 2.6.3-p62), codename: Llamas in Pajamas  
* Min threads: 5, max threads: 5  
* Environment: development  
* Listening on tcp://localhost:3000  
Use Ctrl-C to stop
```



Conectando con la App

- localhost:3000/students
- localhost:3000/students/1

```
Started GET "/students/1" for ::1 at 2019-09-04 22:39:37 -0400
Processing by StudentsController#show as HTML
Parameters: {"id"=>"1"}
Student Load (0.3ms) SELECT "students".* FROM "students" WHERE "students"."id" = ? LIMIT ? [[{"id": 1}, {"LIMIT": 1}]]  
↳ app/controllers/students_controller.rb:67:in `set_student'
Rendering students/show.html.erb within layouts/application
Rendered students/show.html.erb within layouts/application (Duration: 1.6ms | Allocations: 278)
Completed 200 OK in 32ms (Views: 26.3ms | ActiveRecord: 0.3ms | Allocations: 7219)
```

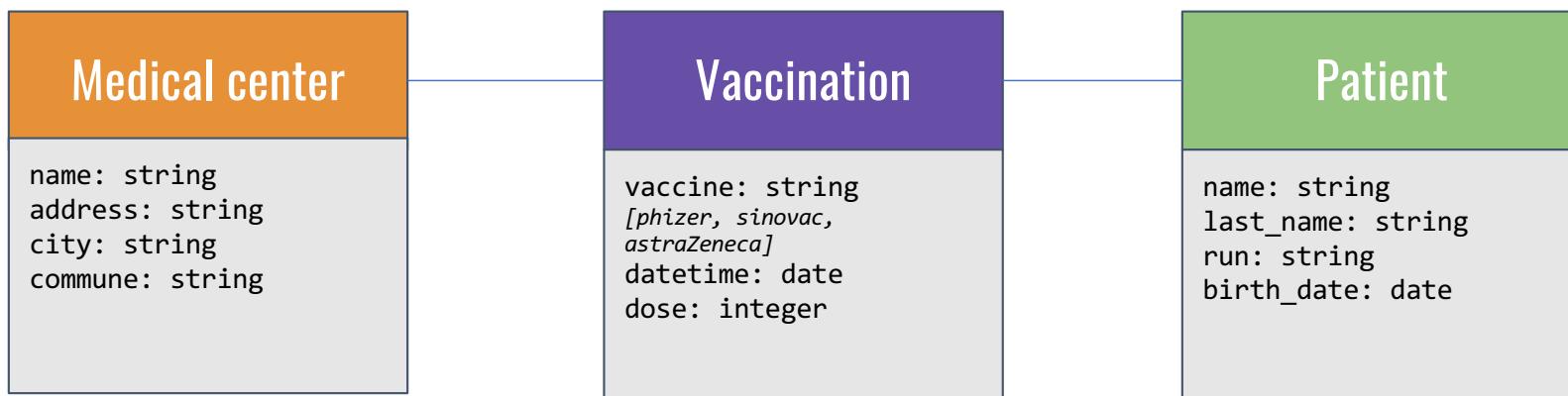
The image displays four sequential screenshots of a web application interface:

- Screenshot 1:** A sidebar titled "Students" contains a "Rut Name" input field and a "New Student" button.
- Screenshot 2:** A modal titled "New Student" shows two input fields: "Rut" (containing "14345765-3") and "Name" (containing "Juan Perez"). Below the inputs are "Create Student" and "Back" buttons.
- Screenshot 3:** A success message "Student was successfully created." is displayed above the student details: "Rut: 14345765-3" and "Name: Juan Perez". There are "Edit" and "Back" links at the bottom.
- Screenshot 4:** The student details are shown again: "Rut: 14345765-3" and "Name: Juan Perez", with "Edit" and "Back" links.

Modelo, migraciones y tablas

- Una de las cosas en que Rails brilla !!
- Modelo (clases Ruby) está íntimamente asociado a las tablas de la BD
- No es necesario manipular las tablas, todo se hace a través de "migraciones"
- Una migración es código Ruby que
 - crea las tablas de la BD
 - actualiza las tablas cuando hay cambios en el modelo

Un ejemplo: Vacunas Covid



Construir el modelo Patient

1. Creamos el modelo Patient desde la terminal

```
/my_app$ rails g model Patient name:string last_name:string run:string birth_date:date
```

2. Se generó el modelo y la migración

La migración permite hacer cambios controlados a la base de datos. Agregar Tablas, editar o eliminar columnas, todo se hace mediante Migraciones. El schema no se modifica, sólo nos permite ver cómo es nuestra base de datos

```
class CreatePatients < ActiveRecord::Migration[5.2]
  def change
    create_table :patients do |t|
      t.string :name
      t.string :last_name
      t.string :run
      t.date :birth_date

      t.timestamps
    end
  end
end
```

Se genera la tabla patients con las columnas name, last_name, run y birth_date del tipo definido.

Generará las columnas created_at y updated_at

Observaciones

```
/my_app$ rails g model Patient
```

```
class Patient < ApplicationRecord  
end
```

Cuando hablamos de un modelo
hablamos en singular

```
ActiveRecord::Schema.define(version: 2021_03_28_223748) do  
  
  create_table "patients", force: :cascade do |t|  
    t.string "name"  
    t.string "last_name"  
    t.string "run"  
    t.date "birth_date"  
    t.datetime "created_at", null: false  
    t.datetime "updated_at", null: false  
  end  
  
end
```

Cuando hablamos de la tabla en
la base de datos hablamos en
plural

Crear la tabla y agregar primeros pacientes

Corremos la migración desde la terminal

```
/my_app$ rails db:migrate
```

Creamos nuestros primeros Pacientes desde la consola de rails

```
/my_app$ rails c
```



```
2.6.6 :001 > Patient.create(name: 'Benjamin', last_name: 'Ibarra', birth_date: Date.parse('20-04-1989'), run: '11111111-1')
 (0.1ms) begin transaction
 Patient Create (0.4ms) INSERT INTO "patients" ("name", "last_name", "run", "birth_date", "created_at", "updated_at") VALUES (?, ?, ?, ?, ?, ?)
 [me, "Benjamin"], ["last_name", "Ibarra"], ["run", "11111111-1"], ["birth_date", "1989-04-20"], ["created_at", "2021-03-28 22:47:59"], ["updated_at", "2021-03-28 22:47:59.145812"]]
 (8.1ms) commit transaction
 => #<Patient id: 1, name: "Benjamin", last_name: "Ibarra", run: "11111111-1", birth_date: "1989-04-20", created_at: "2021-03-28 22:47:59", updated_at: "2021-03-28 22:47:59">
```

Creamos un controlador desde la terminal

```
/my_app$ rails g controller Patients
```



Cuando hablamos de controladores, hablamos en plural

Un poco mas ...

- Agregamos el método index vacío al controlador

```
class PatientsController < ApplicationController
  def index
  end
end
```

- Agregamos la vista para el método index en la carpeta de patients con nombre index.html.erb

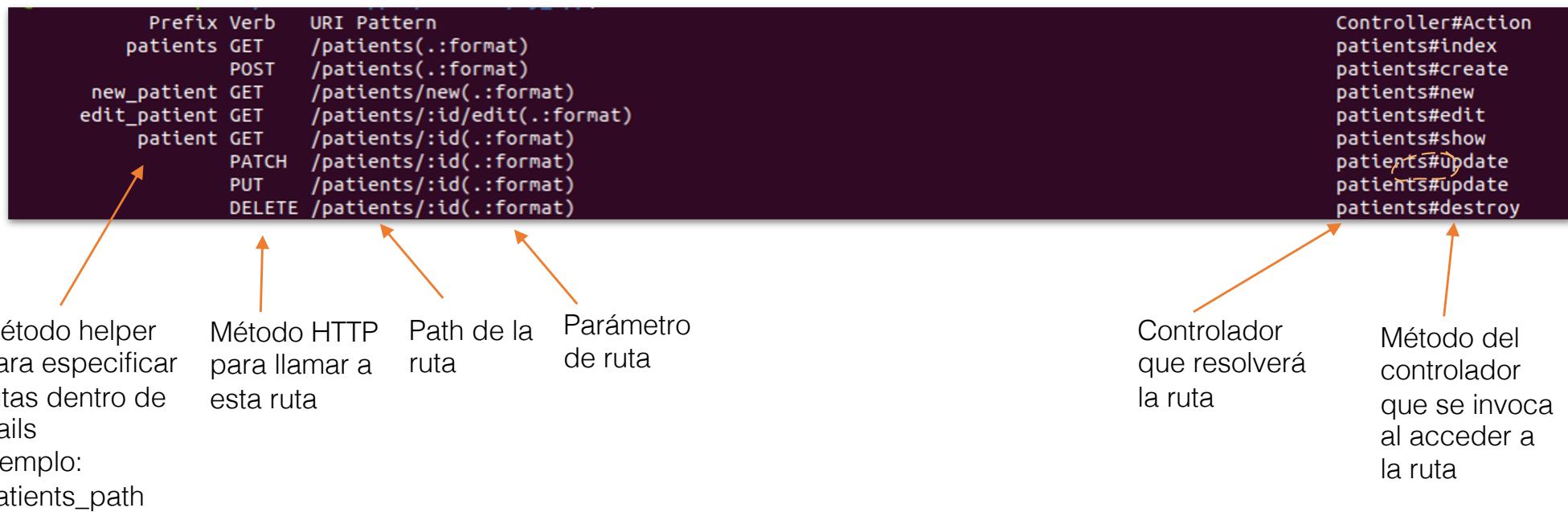
```
<h1> All patients </h1>
```

- Agregamos la ruta para nuestro método index

```
Rails.application.routes.draw do
  # For details on the DSL available within this file,
  #
  # get 'patients/index'
  # get '/patients', to: 'patients#index'
  resources :patients, only: :index
end
```

Rutas

- Podemos revisar las rutas existentes con el comando `rake routes` en la terminal



últimos detalles

- Agregamos una variable de instancia con todos los pacientes. Todas las variables de instancia declaradas en el controlador estarán disponibles en la vista

```
def index
  @patients = Patient.all
end
```

- Iteramos sobre la variable y mostramos los distintos pacientes en la vista

```
<div>
  <% @patients.each do |patient| %>
    <div style="border: solid 1px; width: 200px; padding: 10px; align-text: center; margin: 5px;">
      <h3><%= "#{$patient.name} ${patient.last_name}" %></h3>
      <p><%= patient.run %></p>
    </div>
  <% end %>
<div>
```

<% Código ruby embebido

<%= Expresión ruby que es evaluada y reemplazada in situ

Ahora podemos levantar el servidor y correr la aplicación

```
/my_app$ rails s
```

<http://localhost:3000/patients>

All patients

Benjamin Ibarra
11111111-1
Laura Luna
12111111-1

Operaciones CRUD - Create, Read, Update and Delete

Operaciones CRUD

			métodos en el controlador	Tipo de request
C	Create	crear un recurso	new create	GET POST
R	Read	listar recursos ver un recurso	index show	GET GET
U	Update	editar un recurso	edit update	GET PUT / PATCH
D	Delete	eliminar un recurso	destroy	DELETE

Listar recursos

index: Listar todos los pacientes

patients_controller.rb

```
def index
  @patients = Patient.all
end
```

views/patients/index.html.erb

```
<h1> All patients </h1>

<div style="display: flex; flex-flow: row wrap; ">
  <% @patients.each do |patient| %>
    <div style="border: solid 1px; width: 200px; padding: 10px">
      <h3><%= patient.full_name%></h3>
      <p><%= patient.run %></p>
    </div>
  <% end %>
</div>
```

localhost:3000/patients

All patients

Gertrud Purdy 28795934-6	Ester Trantow 32104132-9	Danae Kemmer 96589731-3
Cornell Kilback 28923997-9	Jonell Harris 53492820-3	Qiana Frami 40085307-k
Elvin Beahan 400264-4	Irena Beer 51413506-1	

Mostrar un recurso

show: Mostrar información de un paciente

patients_controller.rb

```
def show
| @patient = Patient.find_by(id: params[:id])
end
```

views/patients/show.html.erb

```
<h2>Patient: <%= @patient.full_name %></h2>

<div>
| <h3>Nombre</h3>
| <p><%= @patient.name %></p>
</div>

<div>
| <h3>Apellidos</h3>
| <p><%= @patient.last_name %></p>
</div>

<div>
| <h3>Run</h3>
| <p><%= @patient.run %></p>
</div>
```

localhost:3000/patients/2

Patient: Gertrud Purdy

Nombre

Gertrud

Apellidos

Purdy

Run

28795934-6

Fecha de Nacimiento

25/05/85

Crear un recurso

localhost:3000/patients/new

new/create: Crear un nuevo paciente

GET patients/new

patients#new

El método new resuelve un request de tipo GET y permite mostrar el formulario. Una vez se llene el formulario y se haga submit se hará una llamada POST a la ruta /patients con los datos del formulario en el body de la request

POST patients

patients#create

Los datos del formularios serán accesibles en Rails a través del hash params.

Add new Patient

Nombre:

Apellidos:

Rut

Fecha de Nacimiento:

 mm/dd/yyyy

[Create Patient](#)

[Back](#)

Editar un recurso

edit/update: Editar paciente

views/patients/show.html.erb

```
<%= link_to 'Editar', edit_patient_path(@patient) %>
```

patients_controller.rb

```
def edit
  @patient = Patient.find_by(id: params[:id])
end

def update
  @patient = Patient.find_by(id: params[:id])
  if @patient.update(patient_params)
    redirect_to @patient
  else
    render 'edit'
  end
end
```

localhost:3000/patients/2

Patient: Gertrud Purdy

Nombre

Gertrud

Apellidos

Purdy

Rut

28795934-6

Fecha de Nacimiento

25/05/85

Edad

35

[Editar](#) |
[Volver](#)

Eliminar un paciente

destroy: Eliminar un paciente

views/patients/show.html.erb

```
<div>
  <%= link_to 'Editar', edit_patient_path(@patient) %>
  |
  <%= link_to 'Eliminar', patient_path(@patient), method: :delete %>
</div>
```

patients_controller.rb

```
def destroy
  @patient = Patient.find_by(id: params[:id])
  @patient.destroy

  redirect_to patients_path
end
```

localhost:3000/patients/2

Patient: Gertrud Purdy

Nombre

Gertrud

Apellidos

Purdy

Run

28795934-6

Fecha de Nacimiento

25/05/85

Edad

35

[Editar](#) | [Eliminar](#)
[Volver](#)

DRY!!! usar partials

views/patients/_form.html.erb

```
<%= form_with model: @patient, local: true do |form| %>
<p>
  Nombre:
  <br>
  <%= form.text_field :name %>
</p>

<p>
  Apellidos:
  <br>
  <%= form.text_field :last_name %>
</p>

<p>
  Run
  <br>
  <%= form.text_field :run %>
</p>

<p>
  Fecha de Nacimiento:
  <br>
  <%= form.date_field :birth_date, max: Time.now %>
</p>

<p>
  <%= form.submit %>
</p>

<% end %>
```

views/patients/new.html.erb

```
<h2>Add new Patient</h2>

<%= render 'form' %>

<%= link_to 'Back', patients_path %>
```

views/patients/edit.html.erb

```
<h2>Edit Patient</h2>

<%= render 'form' %>

<%= link_to 'Back', patient_path(@patient) %>
```