

IIC 2143 – Ingeniería de Software

Vistas en Rails

M. Trinidad Vargas
mtvargas1@uc.cl

Hasta ahora ...

Creamos una api, donde cada request devuelve un json

GET <http://localhost:3000/recipes>

```
[
  {
    "id": 1,
    "name": "Pastel de Choclo",
    "duration": 50,
    "created_at": "2025-03-12T04:40:02.557Z",
    "updated_at": "2025-03-12T04:40:02.557Z"
  },
  {
    "id": 2,
    "name": "Empanadas de Queso",
    ...
  }
]
```

Modelo - Vista - Controlador

Modelo: contiene la lógica de negocio y los datos de la aplicación

Vista: es la interfaz de usuario, muestra los datos proporcionados por el modelo en un formato adecuado

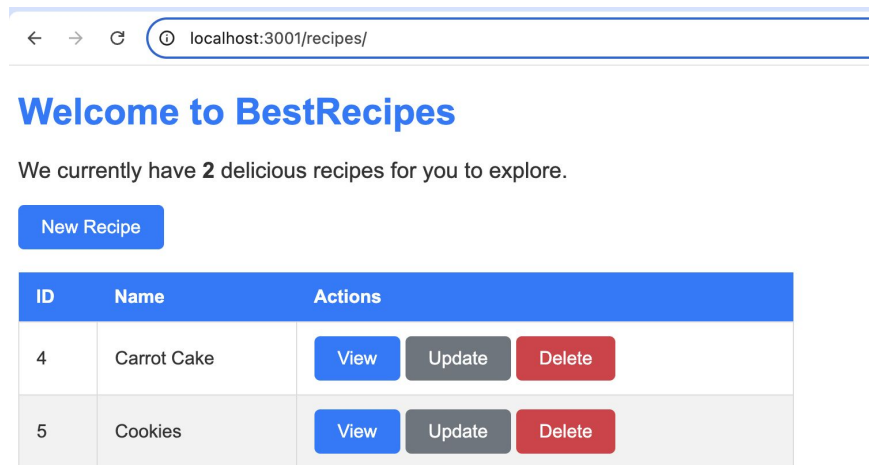
Controlador: actúa como un intermediario entre el modelo y la vista

Ahora ...

Crearemos una aplicación web

GET <http://localhost:3000/recipes>

Response es html



Vistas en Ruby on Rails

- Ruby genera archivos html a partir de archivos Embedded Ruby
- Se procesa el código Ruby antes de mandarselo al navegador
- Son archivos .html.erb localizados en app/views
- Usualmente solo escribimos el body del HTML
- La base del html está en app/views/layouts/application.html.erb

```
app/views/recipes/index.html.erb
```

```
<h1> Welcome to BestRecipes </h1>
<% @recipes.each do |recipe| %>
  <%# Your code here %>
<% end %>
```

Vistas en Ruby on Rails

- El código ruby va entre tags
 - `<% %>` Para realizar operaciones
 - `<%= %>` Para mostrar información
- Las variables de instancia del controlador son accesibles desde la vista

```
<% @recipes.each do |recipe| %>
  <tr>
    <td><%= recipe.name %></td>
    <td><%= recipe.duration %></td>
    <td><%= link_to 'Ver receta', recipe_path(recipe) %></td>
  </tr>
<% end %>
```

Helpers

- Son métodos para abstraer la lógica reutilizable
- Existen helpers de rutas y de vistas
- Puedes definir tus helpers the vista en app/helpers

Rutas:

- <https://guides.rubyonrails.org/routing.html>

Vista

- Vistas: https://guides.rubyonrails.org/action_view_helpers.html
- Formulario: https://guides.rubyonrails.org/form_helpers.html

Path Helpers

Permite generar rutas y asociarlas un método del controlador automáticamente

config/routes.rb

```
Rails.application.routes.draw do
  resources :photos
end
```

HTTP Verb	Path	Controller#Action	Used to
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

Asociar controladores y vistas

- Sigue el principio convención sobre configuración
- Cada operación asociada a una ruta del controlador puede terminar indicando una vista para el usuario
- Existen dos métodos para indicar las vistas desde el controlador
 - **render:** devuelve una vista en el HTTP response
 - **redirect_to:** redirige a la nueva ruta

Asociar controladores y vistas

- Devuelve por defecto **render <método>**
- render busca el archivo con **el mismo nombre** dentro de views/objetos
- No llama (nuevamente) al método del controlador

app/controllers/recipes_controller.rb

```
def index
  @recipes = Recipe.all
  # render: index
end
```

app/views/recipes/index.html.erb

```
<h1> Recipes </h1>
```

Asociar controladores y vistas

- `redirect_to` devuelve una redirección al navegador para que el navegador consulte la nueva ruta
- `:see_other` o 303 es usado para redirección

app/controllers/recipes_controller.rb

```
def destroy
  @recipe = Recipe.find(params[:id])
  @recipe.destroy
  redirect_to action: "index", status: :see_other
end
```

Asociar controladores y vistas

Más formas de usar render

- `render :edit`
- `render action: :edit`
- `render "edit"`
- `render action: "edit"`
- `render "recipes/edit"`
- `render template: "recipes/edit"`

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

Helpers para crear formularios

Son helpers para simplificar la creación de formularios

- El siguiente código .html.erb

```
<%= form_with(model: recipe) do |form| %>
  <div class="field">
    <%= form.label :name%>
    <%= form.text_field :name%>
  </div>
<% end%>
```

- Devuelve el siguiente .html

```
<form action="/recipes" method="post">
  <div class="field">
    <label for="recipe_name">Name</label>
    <input type="text" name="recipe[name]" id="recipe_name" />
  </div>
</form>
```

Helpers para crear formularios

```
<%= form.text_area :message, size: "70x5" %>
<%= form.hidden_field :parent_id, value: "foo" %>
<%= form.password_field :password %>
<%= form.number_field :price, in: 1.0..20.0, step: 0.5 %>
<%= form.range_field :discount, in: 1..100 %>
<%= form.date_field :born_on %>
<%= form.time_field :started_at %>
<%= form.datetime_local_field :graduation_day %>
<%= form.month_field :birthday_month %>
<%= form.week_field :birthday_week %>
<%= form.search_field :name %>
<%= form.email_field :address %>
<%= form.telephone_field :phone %>
<%= form.url_field :homepage %>
<%= form.color_field :favorite_color %>
```

Crear una aplicación web

```
> rails new recipes_app --database=postgresql
```

```
> cd recipes_app
```

```
> rails server
```

<http://localhost:3000>

Crear una página web

1. Definir las rutas: Usa resources
2. Crear un modelo Recipe y migrar
3. Crear un controlador Recipes
4. Implementar las operaciones en el controlador
5. Implementar las vistas
6. Probar en el navegador

Crear el modelo y controlador

```
> rails generate model Recipe name:string description:text  
preparation_time:integer category:string servings:integer  
image:string
```

```
> rails db:create
```

```
> rails db:migrate
```

```
> rails generate controller Recipes
```

Crea una receta en la consola para tener un ejemplo

Definir las rutas

config/routes.rb

```
Rails.application.routes.draw do  
  resources :recipes  
end
```

Crear vista index

app/controllers/recipes_controller.rb

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

Crear vista index

app/views/recipes/index.html.erb

```
<h1> Welcome to BestRecipes </h1>
<p>We currently have <strong><%=
Recipe.count %></strong> delicious
recipes for you to explore.</p>
```



```
<%= link_to "New Recipe",
new_recipe_path, class: 'btn primary' %>
```

```
<table>
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Actions</th>
  </tr>
```

```
<% @recipes.each do |recipe| %>
  <tr>
    <td><%= recipe.id %></td>
    <td><%= recipe.name %></td>
    <td>
      <%= link_to 'View',
recipe_path(recipe), class: 'btn
primary' %> <%= link_to 'Update',
edit_recipe_path(recipe), class: 'btn
secondary' %>
      <%= link_to 'Delete',
recipe_path(recipe), data:
{turbo_method: :delete, turbo_confirm:
"Are you sure?"}, class: 'btn danger'
%> </td>
    </tr>
  <% end %>
</table>
```

Crear vista index

<http://localhost:3000/recipes>

  localhost:3001/recipes/		
<h2>Welcome to BestRecipes</h2>		
<p>We currently have 2 delicious recipes for you to explore.</p>		
New Recipe		
ID	Name	Actions
4	Carrot Cake	View Update Delete
5	Cookies	View Update Delete

Crear vista show

```
class RecipesController < ApplicationController
  # GET /recipes/:id
  def show
    @recipe = Recipe.find(params[:id])
  end
end
```

Crear vista show

app/views/recipes/show.html.erb

```
<h1><%= @recipe.name %></h1>
<p><%= @recipe.description %></p>
<%= link_to 'Update', edit_recipe_path(@recipe), class: 'btn secondary' %>
<%= link_to 'Delete', recipe_path(@recipe), data: {turbo_method: :delete,
turbo_confirm: "Are you sure?"}, class: 'btn danger' %>
<%= link_to "All recipes", "/recipes/", class: 'btn primary' %>
```

Crear vista show



Cookies

The best cookies ever



Crear vista new

```
class RecipesController < ApplicationController
  # GET /recipes/new
  def new
    @recipe = Recipe.new
  end

  # POST /recipes
  def create
    @recipe = Recipe.new(recipe_params)
    if @recipe.save
      redirect_to action: "show", id: @article.id
    else
      render :new, status: :unprocessable_entity
    end
  end
end
```

Crear vista new

app/views/recipes/new.html.erb

```
<%= form_with(model: recipe, local: true) do |form| %>
<div class="field">
  <%= form.label :name%>
  <%= form.text_field :name%>
</div>

<div class="field">
  <%= form.label :description%>
  <%= form.text_area :description%>
</div>

<div class="actions">
  <%= form.submit %>
</div>
```

Crear vista new

Agregamos mensajes de error de validación

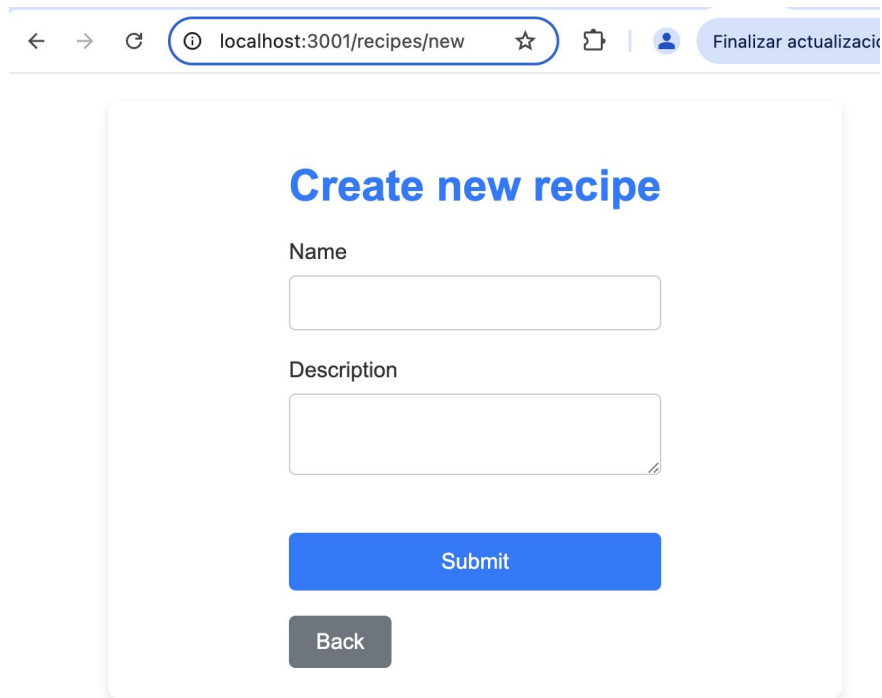
app/models/recipe.rb

```
class Recipe < ApplicationRecord
  validates :name, :description, presence: true
end
```

app/views/recipes/new.html.erb

```
<% if recipe.errors.any? %>
  <p><%= Error saving:</p>
  <ul>
    <% recipe.errors.full_messages.each do |msg| %>
      <li><%= msg %></li>
    <% end %>
  </ul>
<% end %>
```

Crear vista new



The image shows a web browser window with the address bar displaying 'localhost:3001/recipes/new'. The page content is a form titled 'Create new recipe' in blue text. The form has two input fields: 'Name' and 'Description'. Below the 'Description' field is a blue 'Submit' button and a grey 'Back' button. The browser's top bar includes navigation icons, a star icon, and a user profile icon with the text 'Finalizar actualizaci'.

← → ↻ ⓘ localhost:3001/recipes/new ☆ 📄 👤 Finalizar actualizaci

Create new recipe

Name

Description

Submit

Back

Crear vista edit

```
class RecipesController < ApplicationController
  # GET /recipes/:id/edit
  def edit
    @recipe = Recipe.find(params[:id])
  end

  def update
    @recipe = Recipe.find(params[:id])
    if @recipe.update(recipe_params)
      redirect_to action: "show", id: @recipe.id
    else
      render :edit, status: :unprocessable_entity
    end
  end
end
```

Crear vista edit

¿Cómo reutilizar el formulario de new.html.erb?

Usando **render** podemos crear un archivo _form.html.erb

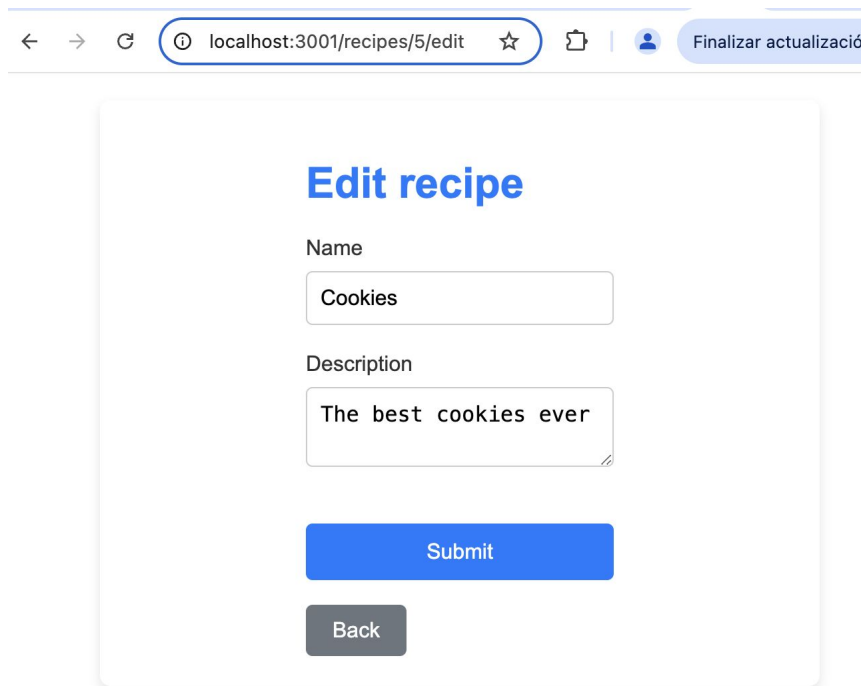
- app/views/recipes/edit.html.erb

```
<%= render "form", recipe: @recipe %>
```

```
<%= link_to "Back", recipes_path %>
```

- Agregar el formulario de new.html.erb en misma carpeta como _form.html.erb y hacer lo mismo en new.html.erb
- Usamos recipe en vez de @recipe

Crear vista edit



The screenshot shows a web browser window with the address bar displaying 'localhost:3001/recipes/5/edit'. The page content is a form titled 'Edit recipe' in blue. It contains two input fields: 'Name' with the value 'Cookies' and 'Description' with the value 'The best cookies ever'. Below the fields are two buttons: a blue 'Submit' button and a grey 'Back' button. The browser's top bar includes navigation icons, a user profile icon, and a button labeled 'Finalizar actualización'.

← → ↻ ⓘ localhost:3001/recipes/5/edit ☆ 📄 | 👤 Finalizar actualización

Edit recipe

Name

Description

Submit

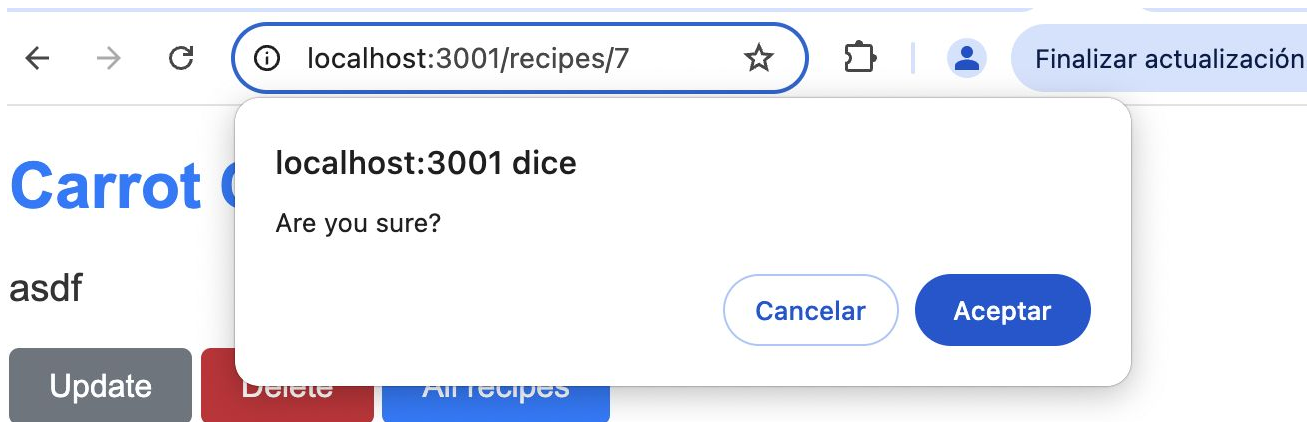
Back

Crear controlador destroy

```
class RecipesController < ApplicationController
# DELETE /recipes/:id
  def destroy
    @recipe = Recipe.find(params[:id])
    @recipe.destroy

    redirect_to action: "index", status: :see_other
  end
end
```


Crear controlador destroy



Agregar estilo CSS

- Crear archivos .css en app/assets/stylesheets
- Agregar en app/assets/stylesheets/application.css

`*= require filename`

Ejemplo

crear recipesapp.css

app/assets/stylesheets/application.css

`*= require recipesapp`

Ejercicio con décima

En grupo de hasta 3 personas

- Crear un modelo cualquiera (User, Article ...)
- Crear las vistas index con una tabla, la vista del update y destroy
- Entregar los archivos controller, las vistas html.erb y un archivo pdf con capturas de cada vista
- Se dará la décima solo si incluye todos los elementos

Entregar hasta miércoles 16 de abril

