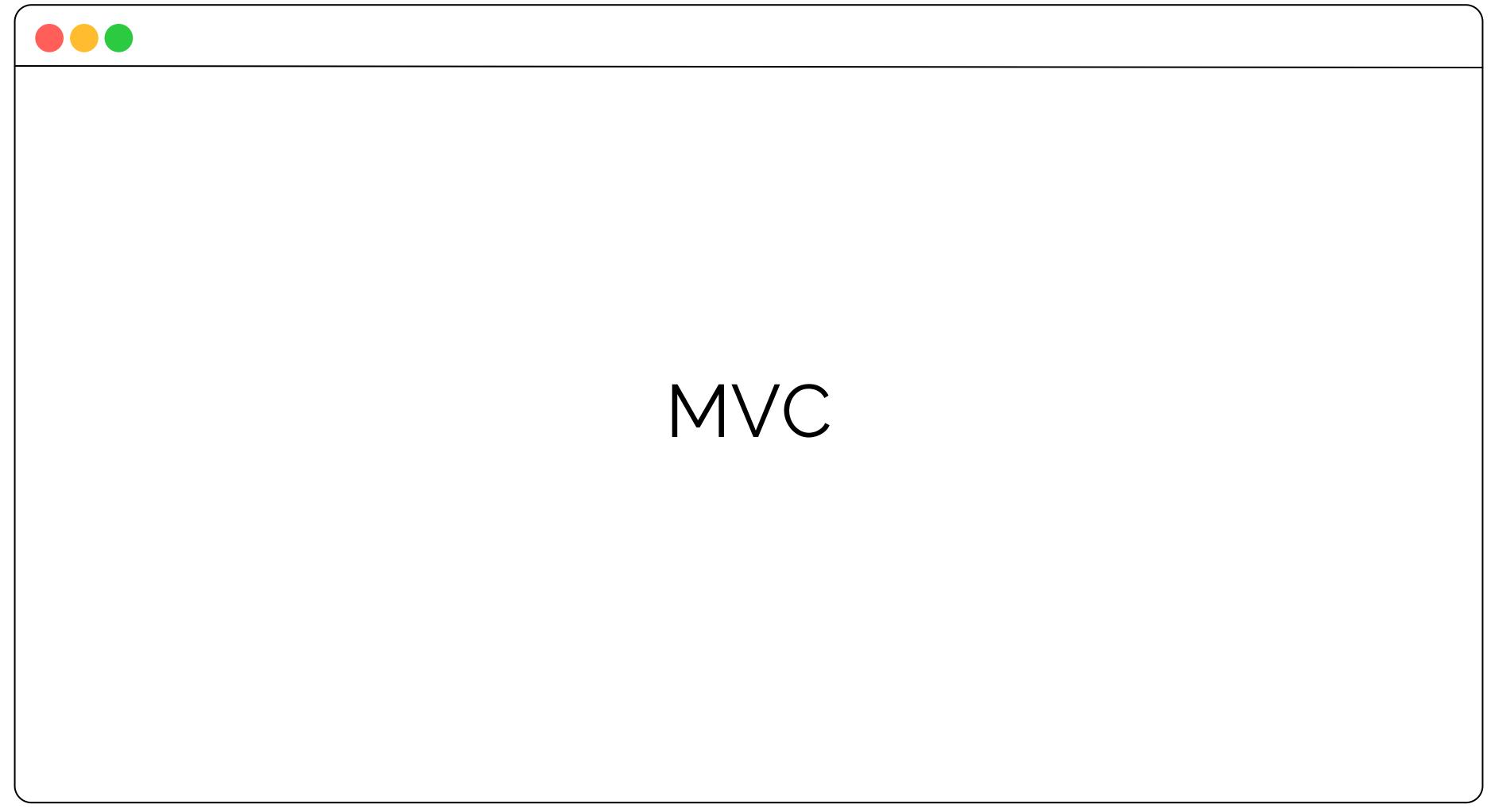




10 - MVC + Routing

IIC2143-3 Josefa España

jpespana@uc.cl





Modelo-Vista-Controlador

Arquitectura que estructura la aplicación, permitiendo organizar la lógica, vistas, etc.

- Modelo: Donde haremos todo nuestro manejo de datos con la base de datos.
- Controlador: Maneja la lógica de negocio, requests de usuarios, y el flujo de llevar la data desde el modelo a la vista. (intermediario)
- Vista: Responsable de lo que el usuario visualiza.



Model

Comando usado en consola para crearlo:

1 rails generate model Pokemon name:string pokemon_type:string

Ejemplo del modelo Pokemon, que se encuentra en el archivo app/models/pokemon.rb":

```
1 class Pokemon < ApplicationRecord</pre>
```

2 end

3



Controller

Ejemplo del controlador Pokemon, que se encuentra en el archivo app/controllers/pokemons_controller.rb":

Comando usado en consola para crearlo:

1 rails generate controller Pokemons



Controller

Comando usado en consola para crearlo:

```
1 rails generate controller Pokemons
```

Para cada método, disponibilizamos una ruta en el archivo config/routes.rb:

```
Rails.application.routes.draw do
root to: "pokemons#index"

get "/pokemons", to: "pokemons#index", as: :pokemons

get "/pokemons/new", to: "pokemons#new", as: :new_pokemon

get "/pokemons/:id", to: "pokemons#show", as: :pokemon

end
```

Rails después de ejecutar el método index del controlador PokemonsController, "renderiza" el archivo "pokemons/index.html.erb".

Por defecto busca un archivo con el **mismo** nombre del método, en una carpeta con el **mismo** nombre que el controlador.



Vista

En app/views/pokemons/index.html.erb:

```
<h1>All Pokemons</h1>
   <thead>
     Name
      Type
     </thead>
    <% @pokemons.each do |pokemon| %>
11
12
      <%= pokemon.name %>
13
        <%= pokemon.pokemon_type %>
14
      15
     <% end %>
16
    17
  19
```

i http://localhost:3000

All Pokemons

Name Type

Pikachu Electric

Charmander Fire

Squirtle Water

Bulbasaur Grass

Jigglypuff Normal





Ejemplos

```
1 <h1>
2 Hello <%= name %>
3 </h1>
```

Si la variable name es igual a Loki quedaría así el HTML:

```
1 <h1>
2 Hello Loki
3 </h1>
```



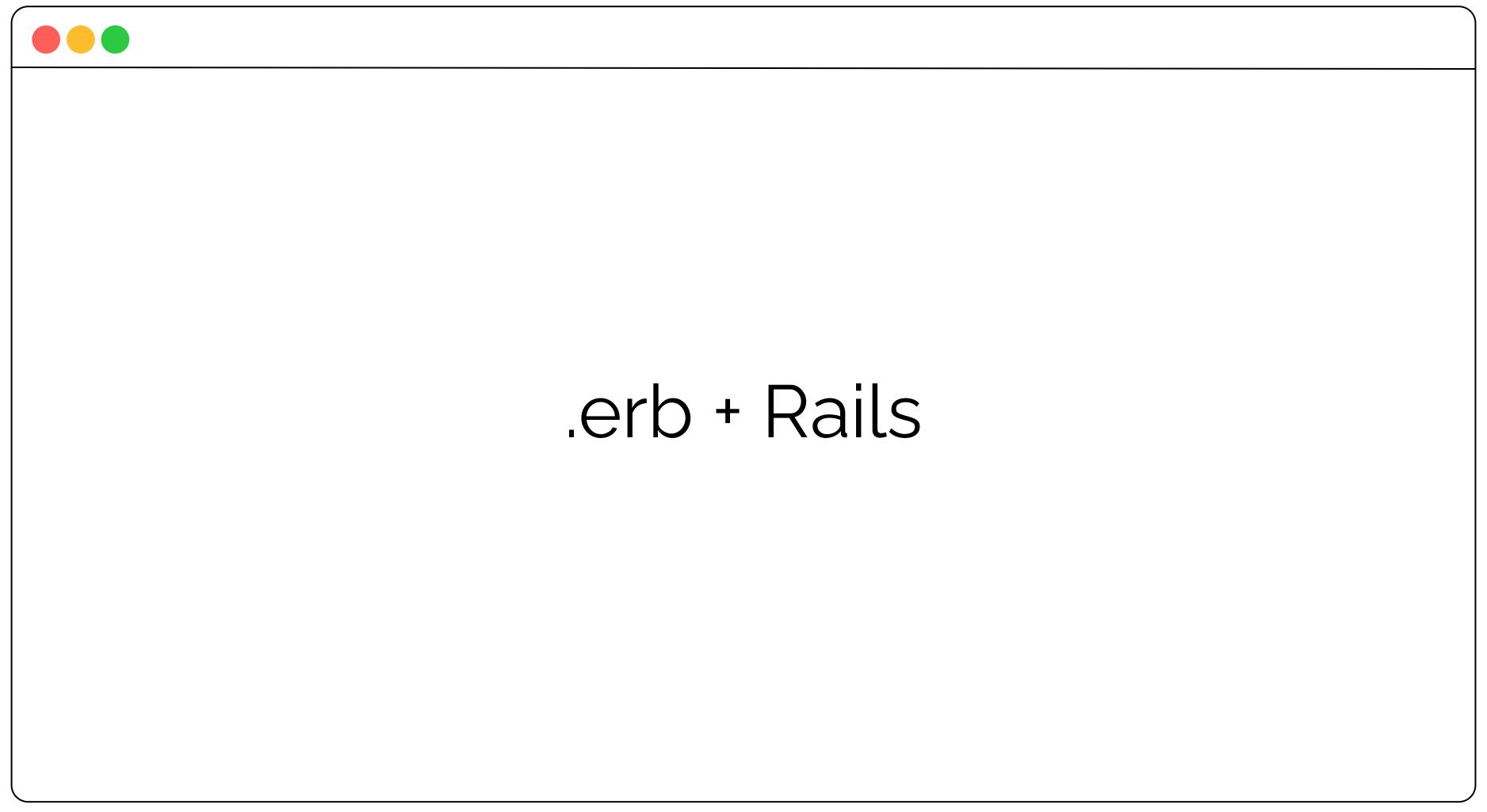
Ejemplos

erb no generará la sección de código de las líneas 1, 3 y 5. En este ejemplo dependiendo de la condición solo mostrará la línea 2 o la línea 4.



Ejemplos

erb generará un HTML con la información de todos los @pokemons, en formato de tabla.





ERB + Rails

• Si en el controlador no usamos el método "render" para devolver un HTTPResponse, Rails buscará un archivo .erb con el mismo nombre de la acción (ej: index), generará el HTML y enviará el HTTPResponse.

app/views/pokemons/index.html.erb



ERB + Rails

La variable @pokemons está definida e inicializada en el método index dentro del controlador PokemonsController.

Rails Form Helpers

Equivalente en HTML:

Rails Form Helpers Checkbox

Equivalente en HTML:

```
1      <input type="checkbox" id="pet_dog" name="pet_dog" value="1" />
2      <label for="pet_dog">I own a dog</label>
3      <input type="checkbox" id="pet_cat" name="pet_cat" value="1" />
4      <label for="pet_cat">I own a cat</label>
```

Rails Form Helpers Otros Ejemplos

```
<%= form.text_area :message, size: "70×5" %>
<%= 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 %>
```





Paths

- 1 Rails.application.routes.draw do
- 2 resources :photos
- 3 end

resources genera las rutas de la foto de manera automática.

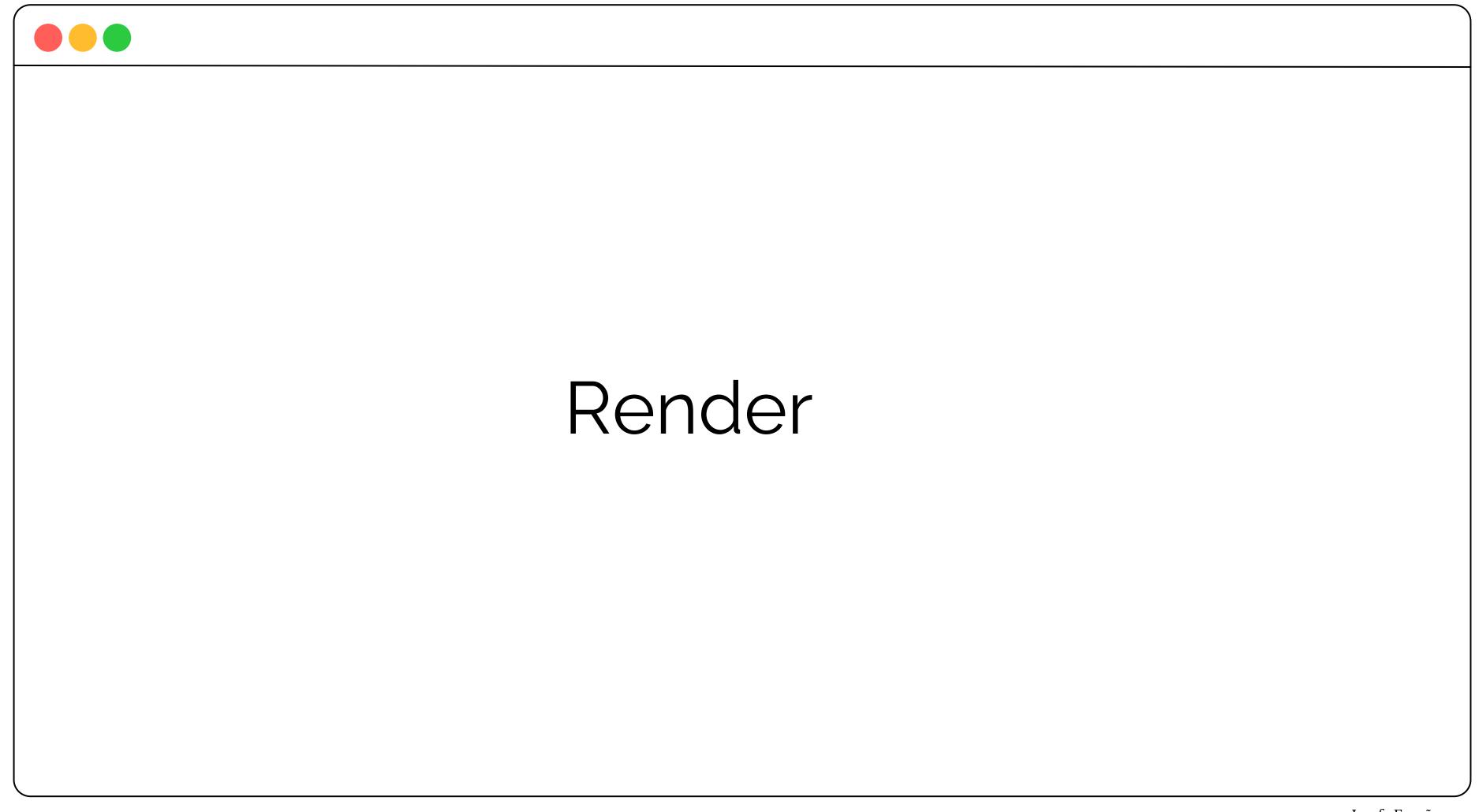
Estas rutas siguen el estándar de la web y de Ruby.

HTTP Verb	Path	Controller#Action	Used for
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



Paths

- photos_path : genera el string "/photos"
- new_photo_path: "/photos/new"
- edit_photo_path(:id) : "/photos/:id/edit"
- photo_path(:id) : "/photos/:id"







Render

```
def update
if @pokemon.update(pokemon_params)
redirect_to @pokemon, notice: 'Pokemon was successfully updated.'
else
render :edit
end
end
```

- Si se quiere renderizar otra vista de nombre diferente al nombre del método, se puede usar el método render.
- En el ejemplo dependiendo de la condición se renderiza "edit.html.erb" o "show.html.erb"

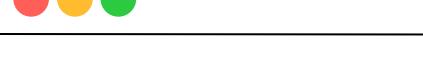


Otras formas de usar Render

```
1 render :edit
2 render action: :edit
3 render "edit"
4 render action: "edit"
5 render "books/edit"
6 render template: "books/edit"
```

• Todas son equivalentes.

Render



redirect_to

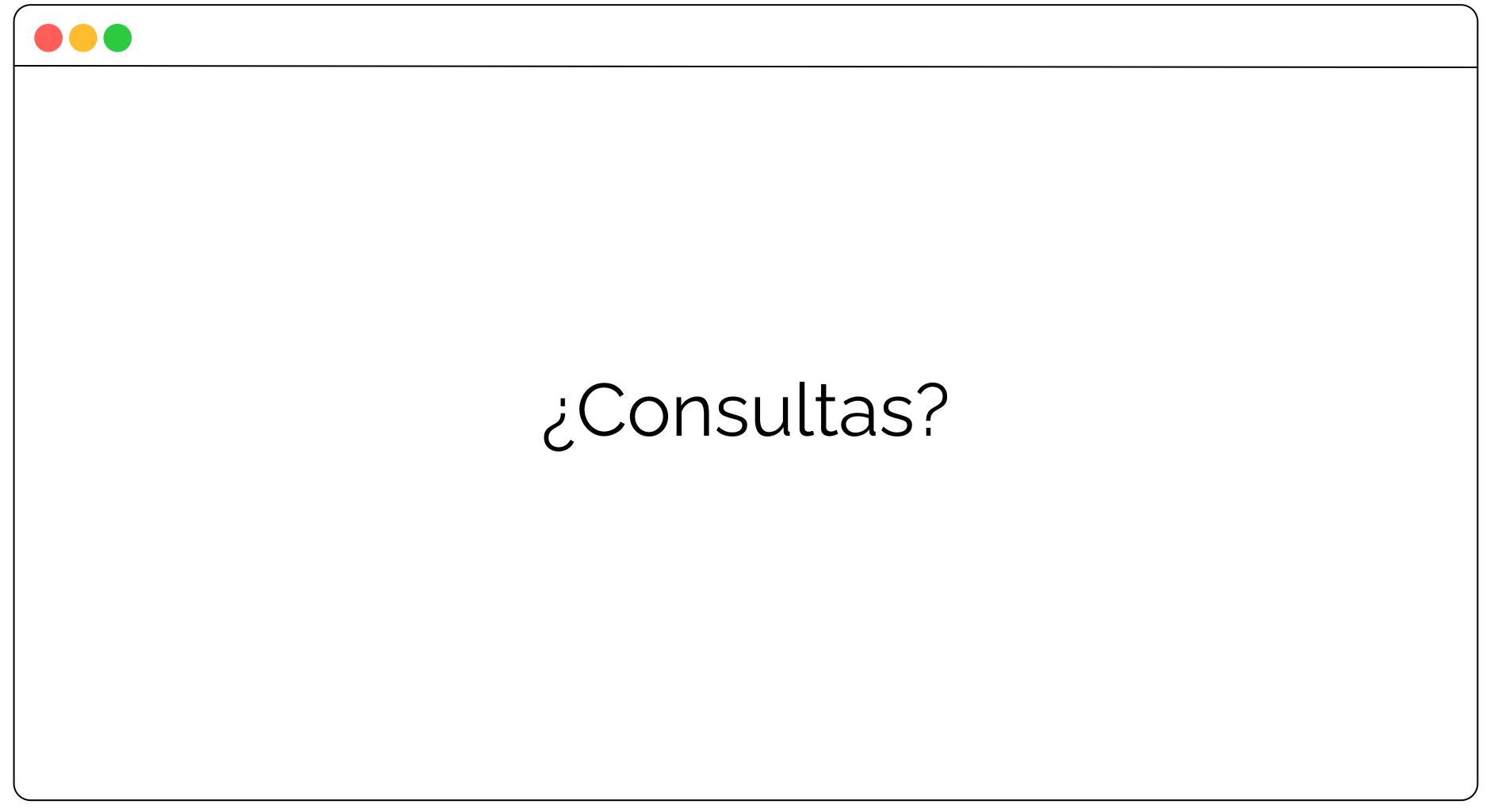
```
def update
if @pokemon.update(pokemon_params)
redirect_to @pokemon, notice: 'Pokemon was successfully updated.'
else
render :edit
end
end
```

- redirect_to no solo renderiza la vista "show.html.erb", ya que para renderizarla es probable que se necesite la variable @pokemon.
- El redirect_to lo que hace es un refresh del browser en la página "show.html.erb" usando @pokemon, por lo que se envía otro HTTPRequest.



Más ejemplos y documentación

- https://guides.rubyonrails.org/form-helpers.html
- https://guides.rubyonrails.org/layouts and rendering.html
- https://guides.rubyonrails.org/routing.html







10 - MVC + Routing

IIC2143-3 Josefa España

jpespana@uc.cl