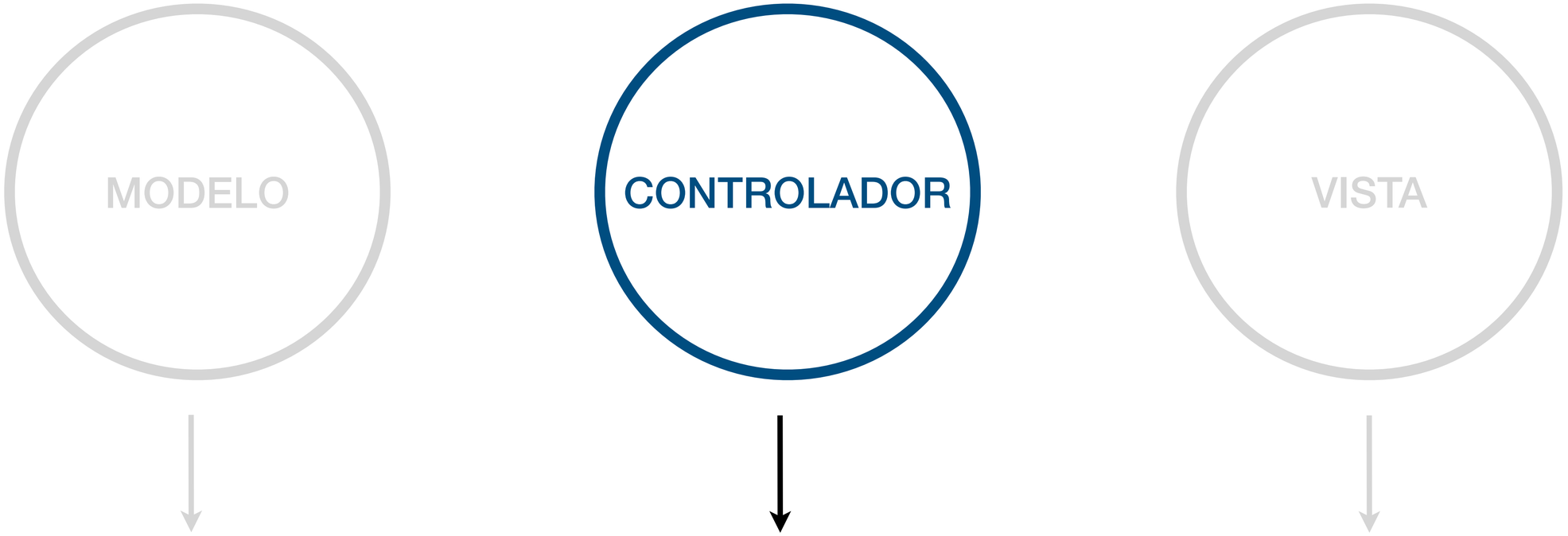
Ingeniería de Software 2021

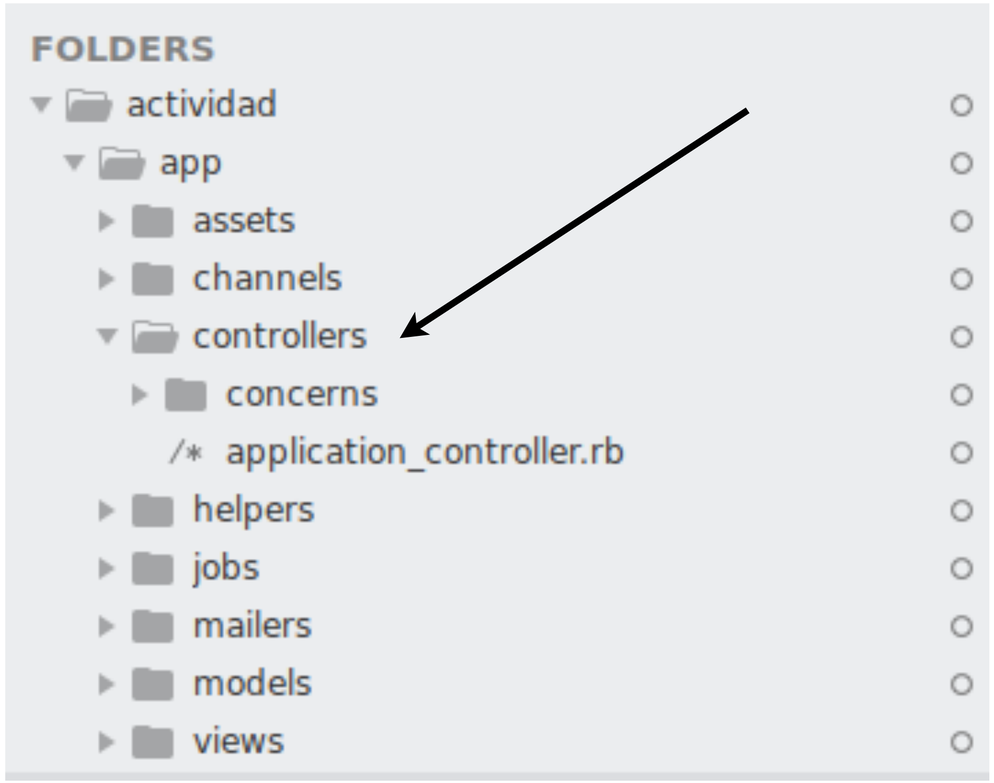
Laboratorio 4 - Controladores

MVC: Model View Controller

Capas de la aplicación



CONTROLADORES EN



Podemos crear el archivo manualmente o usando el comando rails g

El nombre del recurso suele ser en PLURAL

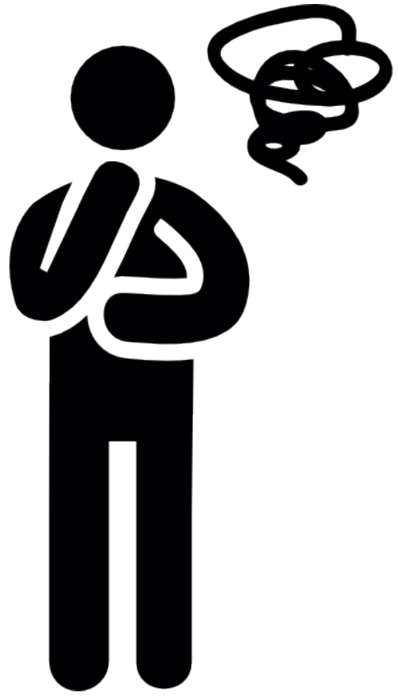
app/controllers/tweets_controller.rb

```
tweets_controller.rb
class TweetsController < ActionController::Base

end
```

tweets_controller.rb

```ruby
class TweetsController < ActionController::Base

  def index
    # Listar tweets
  end

  def show
    # Mostrar datos de un tweet
  end

  def new
    # Mostrar formulario de carga de un tweet
  end

  def create
    # Crear un tweet en la BD
  end

  def edit
    # Mostrar formulario de edición de un tweet
  end

  def update
    # Modificar los datos de un tweet en la BD
  end

  def destroy
    # Eliminar una película
  end

end
```

¿A qué se parece de lo visto hasta ahora?



tweets_controller.rb

```ruby
class TweetsController < ActionController::Base

  def index
    # Listar tweets
  end

  def show
    # Mostrar datos de un tweet
  end

  def new
    # Mostrar formulario de carga de un tweet
  end

  def create
    # Crear un tweet en la BD
  end

  def edit
    # Mostrar formulario de edición de un tweet
  end

  def update
    # Modificar los datos de un tweet en la BD
  end

  def destroy
    # Eliminar una película
  end

end
```

index.html.erb

Mostrará listado de tweets

show.html.erb

Mostrará datos de un tweet en particular

new.html.erb

Mostrará un formulario para cargar un tweet

edit.html.erb

Mostrará un formulario para editar un tweet

tweets_controller.rb

```ruby
class TweetsController < ActionController::Base

  def index
    # Listar tweets
  end

  def show
    # Mostrar datos de un tweet
  end

  def new
    # Mostrar formulario de carga de un tweet
  end

  def create
    # Crear un tweet en la BD
  end

  def edit
    # Mostrar formulario de edición de un tweet
  end

  def update
    # Modificar los datos de un tweet en la BD
  end

  def destroy
    # Eliminar una película
  end

end
```

¿Por qué los métodos create, update y destroy no tienen una vista asociada?

index.html.erb

Mostrará listado de tweets

show.html.erb

Mostrará datos de un tweet en particular

new.html.erb

Mostrará un formulario para cargar un tweet

edit.html.erb

Mostrará un formulario para editar un tweet

tweets_controller.rb

```ruby
class TweetsController < ActionController::Base

  def index
    # Listar tweets
  end

  def show
    # Mostrar datos de un tweet
  end

  def new
    # Mostrar formulario de carga de un tweet
  end

  def create
    # Crear un tweet en la BD
  end

  def edit
    # Mostrar formulario de edición de un tweet
  end

  def update
    # Modificar los datos de un tweet en la BD
  end

  def destroy
    # Eliminar una película
  end

end
```

¿Y qué escribimos en los métodos?

¡Ya podemos interactuar con el modelo!

Puedo usar funciones como: ***new, save, create, update, destroy, find, etc.***

Recordando el laboratorio anterior...

lab3/app/views/monstruos/index.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Monstruos </title> </head>
  <body>
    <table>
      <tr>
        <th> Monstruo </th>
        <th> Película favorita </th>
      </tr>
      <% Monstruo.all.each do |monstruo| %>
      <tr>
        <td> <%= monstruo.nombre %> </td>
        <td> <%= monstruo.pelicula_favorita %> </td>
      </tr>
      <% end %>
    </table>
  </body>
</html>
```

En esta línea estamos llamando al modelo desde la vista

Recordando el laboratorio anterior...

lab3/app/views/monstruos/index.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Monstruos </title> </head>
  <body>
    <table>
      <tr>
        <th> Monstruo </th>
        <th> Película favorita </th>
      </tr>
      <% Monstruo.all.each do |monstruo| %>
      <tr>
        <td> <%= monstruo.nombre %> </td>
        <td> <%= monstruo.pelicula_favorita %> </td>
      </tr>
      <% end %>
    </table>
  </body>
</html>
```

Funciona? SI

Estamos respetando MVC? NO

Recordando el laboratorio anterior...

lab3/app/views/monstruos/index.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Monstruos </title> </head>
  <body>
    <table>
      <tr>
        <th> Monstruo </th>
        <th> Película favorita </th>
      </tr>
      <% Monstruo.all.each do |monstruo| %>
      <tr>
        <td> <%= monstruo.nombre %> </td>
        <td> <%= monstruo.pelicula_favorita %> </td>
      </tr>
      <% end %>
    </table>
  </body>
</html>
```

Funciona? SI

Estamos respetando MVC? NO

Ahora podemos usar los controladores

```
<!DOCTYPE html>
<html>
  <head> <title> Monstruos </title> </head>
  <body>
    <table>
      <tr>
        <th> Monstruo </th>
        <th> Película favorita </th>
      </tr>
      <% @monstruos.each do |monstruo| %>
      <tr>
        <td> <%= monstruo.nombre %> </td>
        <td> <%= monstruo.pelicula_favorita %> </td>
      </tr>
      <% end %>
    </table>
  </body>
</html>
```

lab4/app/controllers/monstruos_controller.rb

```
def index
  @monstruos = Monstruo.all
end
```

Variables de instancia empiezan con @

Funciona? SI

Estamos respetando MVC? SI

Recordando el laboratorio anterior...

lab3/app/views/tweets/show.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Tweet </title> </head>
  <body>
    <% tweet = Tweet.find(1) %>
    <h1> <%= tweet.estado %> </h1>
    <p> Escrito por <%= tweet.monstruo.nombre %> </p>
  </body>
</html>
```

Recordando el laboratorio anterior...

lab3/app/views/tweets/show.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Tweet </title> </head>
  <body>
    <% tweet = Tweet.find(1) %>
    <h1> <%= @tweet.estado %> </h1>
    <p> Escrito por <%= @tweet.monstruo.nombre %> </p>
  </body>
</html>
```

```
def show
  @tweet = Tweet.find(1)
end
```

¿Cómo hago para que no me muestre siempre el Tweet con ID 1, sino el que recibe de la vista?

PARAMS --> Me devuelve un **hash** que contiene el ID que necesito mostrar

Hashes

Series de pares formadas por clave:valor

```
alumno = {apellido: "Diaz", nombre: "Juan", edad: "20"}
```

alumno[:apellido] --> "Diaz"

alumno[:edad] --> "20"

Símbolo

Hashes

Series de pares formadas por clave:valor

```
alumno = {apellido: "Diaz", nombre: "Juan", edad: "20"}
```

alumno[:apellido] --> "Diaz"

alumno[:edad] --> "20"

Símbolo

```
alumnos = {alumno1: {apellido: "Diaz",
                     nombre: "Juan",
                     edad: "20"},
           alumno2: {apellido: "Perez",
                     nombre: "Florencia",
                     edad: "25"}
          }
```

alumnos[:alumno1][:nombre] --> "Juan"

alumnos[:alumno2][:apellido] --> "Perez"

.../tweet/5

```
<!DOCTYPE html>
<html>
  <head> <title> Tweet </title> </head>
  <body>
    <% tweet = Tweet.find(1) %>
    <h1> <%= @tweet.estado %> </h1>
    <p> Escrito por <%= @tweet.monstruo.nombre %> </p>
  </body>
</html>
```

Recordando el laboratorio anterior...

lab3/app/views/tweets/show.html.erb

```
def show
  @tweet = Tweet.find(1)
end
```

¿Cómo hago para que no me muestre siempre el Tweet con ID 1, sino el que recibe de la vista?

PARAMS --> Me devuelve un **hash** que contiene el ID que necesito mostrar

params = {"controller" => "tweets", "action" => "show", "id" => "5"}

.../tweet/5

Recordando el laboratorio anterior...

lab3/app/views/tweets/show.html.erb

```
<!DOCTYPE html>
<html>
  <head> <title> Tweet </title> </head>
  <body>
    <% tweet = Tweet.find(1) %>
    <h1> <%= @tweet.estado %> </h1>
    <p> Escrito por <%= @tweet.monstruo.nombre %> </p>
  </body>
</html>
```

```
def show
  @tweet = Tweet.find(params[:id])
end
```

¿Cómo hago para que no me muestre siempre el Tweet con ID 1, sino el que recibe de la vista?

PARAMS --> Me devuelve un **hash** que contiene el ID que necesito mostrar

params = {"controller" => "tweets", "action" => "show", "id" => "5"}

Crear un monstruo

.../monstruos?monstruo[nombre]=Nahuelito&monstruo[edad]=26

Al hacer este requerimiento en el navegador, al controlador le va a llegar:

```
params = { monstruo: {nombre: "Nahuelito", edad: "26"} }
```

Entonces, podríamos crear un monstruo de la siguiente forma:

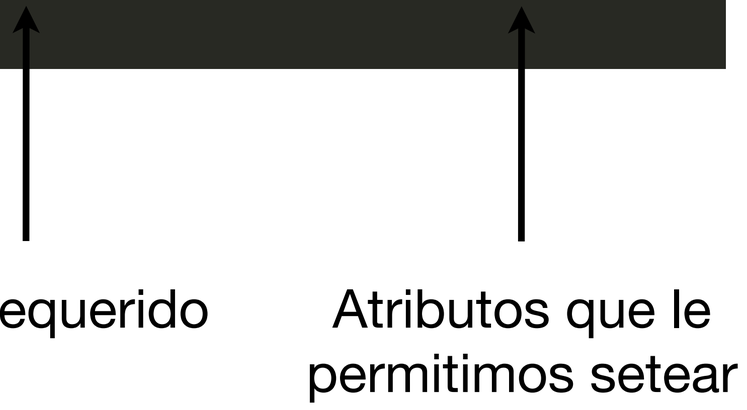
```
def create
  @monstruo = Monstruo.create(nombre: params[:monstruo][:nombre],
                              edad: params[:monstruo][:edad])
end
```

Crear un monstruo

```ruby
def create
  @monstruo = Monstruo.create(nombre: params[:monstruo][:nombre],
                              edad: params[:monstruo][:edad])
end
```

Lo podemos hacer más legible de la siguiente forma:

```ruby
def create
  @monstruo = Monstruo.create(params.require(:monstruo).permit(:nombre, :edad))
end
```



Crear un monstruo

Y lo podemos mejorar aún más...

```
def create
  @monstruo = Monstruo.new(params.require(:monstruo).permit(:nombre, :edad))

  if @monstruo.save
    redirect_to monstruos_path, notice: "¡El monstruo se ha cargado exitosamente!"
  else
    flash[:error] = "Hubo un error al cargar el monstruo"
    render :new
  end
end
```

redirect_to --> Redirecciona a la URL indicada
notice --> Muestra un mensaje en la redirección
flash --> Carga un mensaje en flash que luego es mostrado en la vista correspodiente. Hay de varios tipos.
render --> Vuelve al formulario de new sin perder la información cargada