RAILS: CRUD + ROUTING

Ingeniería de Software – IIC2143

Rodrigo Alonso
rialonso@uc.cl

Patrón Modelo-Vista-Controlador (MVC)

Patrón Modelo-Vista-Controlador (MVC)

MODELO

Representación de los datos, sus relaciones y mecanismos persistencia.

CONTROLADOR

Intermediario, gestiona el flujo de información entre ambos.

VISTA

Interfaz del usuario, donde ocurre la interacción con él.

Routing

RAILS ROUTER

Encargado de reconocer las URL de una solicitud y redirigirla a la acción del controlador que corresponda

config/routes.rb

```
<MÉTODO HTTP> '<URL>', to: '<CONTROLADOR>#<MÉTODO>'
```

USUARIO

URL

ROUTER

MODELO

Representación de los datos, sus relaciones y mecanismos persistencia.

CONTROLADOR

Intermediario, gestiona el flujo de información entre ambos.

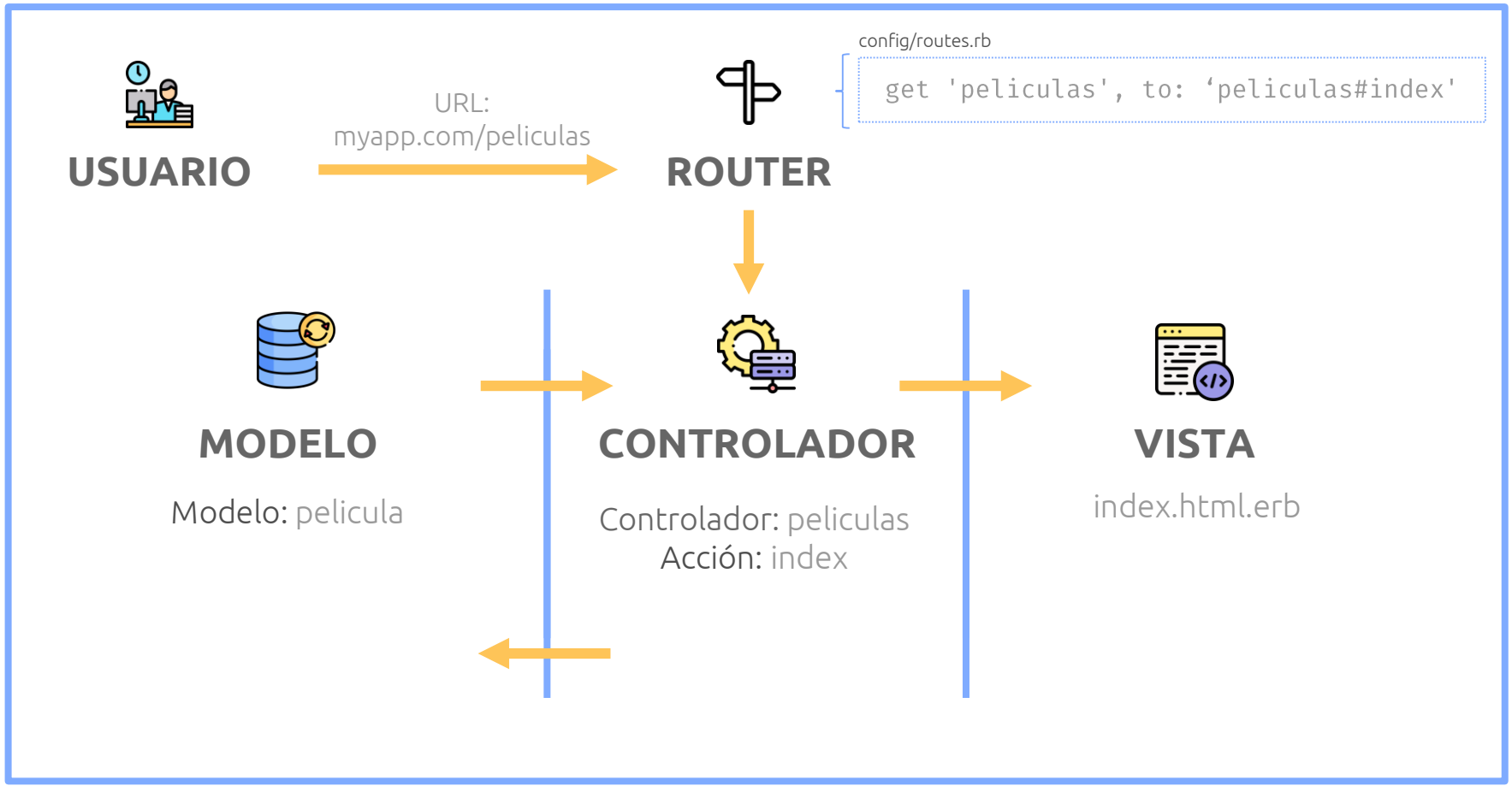
VISTA

Interfaz del usuario, donde ocurre la interacción con él.

Ejemplo

Página que muestre un listado
de películas en la URL:

myapp.com/peliculas


USUARIO
URL:
myapp.com/peliculas
ROUTER
config/routes.rb
get 'peliculas', to: ‘peliculas#index'
MODELO
Modelo: pelicula
CONTROLADOR
Controlador: peliculas
Acción: index
VISTA
index.html.erb

CRUD

CRUD

Create, Read, Update, Delete
(Crear, Leer, Actualizar, Eliminar)

- Cuatro operaciones básicas llevadas a cabo en una base de datos.
- Son ejecutadas sobre instancias de las entidades de la aplicación.

Implementación de un CRUD

Involucra a todos los actores del modelo MVC, es decir, es necesario crear:

- Modelo
- Controlador
- Vistas

Del CRUD en cuestión

CRUD: Modelo

Especificar:

- Nombre del modelo
- Nombres de atributos
- Tipos de atributos

```
rails g model Pelicula titulo:string ano:integer director:string
```

*Crea el modelo y la migración asociada e él

CRUD: Controlador y Vistas

Especificar:

- Nombre del controlador
- Acciones del controlador

```
rails g controller Peliculas new index show edit
```

*Crea el controlador, con las vistas y acciones especificadas.

CRUD: Acciones del Controlador

Create:

- **new**: hace el render del formulario para crear la instancia del modelo.
- **create**: se ejecuta al hacer “submit” del formulario. Crea instancia en base de datos.

Read:

- **index**: muestra la lista con todas las instancias del modelo.
- **show**: muestra información de una instancia específica.

CRUD: Acciones del Controlador

Update:

- **edit**: hace el render del formulario para editar la instancia del modelo.
- **update**: se ejecuta al hacer “submit” del formulario. Actualiza instancia en base de datos.

Delete:

- **destroy**: elimina la instancia de la base de datos.

Actividad Práctica

CRUD de películas con los atributos de:

- Título
- Año
- Director

Rutas del CRUD

rails routes

Path/URL Helpers

Forma de las URL

```
        Prefix Verb   URI Pattern                       Controller#Action
               GET    /                                 pages#main
          root GET    /                                 pages#main
          info GET    /info(.:format)                   pages#info
 peliculas_new GET    /peliculas/new(.:format)          peliculas#new
     peliculas POST   /peliculas(.:format)              peliculas#create
               GET    /peliculas(.:format)              peliculas#index
      pelicula GET    /peliculas/:id(.:format)          peliculas#show
peliculas_edit GET    /peliculas/:id/edit(.:format)     peliculas#edit
               PATCH  /peliculas/:id(.:format)          peliculas#update
               PUT    /peliculas/:id(.:format)          peliculas#update
               DELETE /peliculas/:id(.:format)          peliculas#destroy
```

Método HTTP

Controlador y su acción asociada

Path/URL Helpers

```
<%= link_to 'Ver película', '/peliculas/ + pelicula.id.to_s'  %>
```

```
<%= link_to 'Ver película', pelicula_path(pelicula.id)  %>
```

Scaffold

```
rails g scaffold Pelicula titulo:string ano:integer director:string
```

*Crea el modelo, su migración asociada y el controlador con todas las acciones y vistas del CRUD.

Scaffold

```
rails g scaffold Pelicula titulo:string ano:integer director:string
```

*Crea el modelo, su migración asociada y el controlador con todas las acciones y vistas del CRUD.

Ventajas:

- Rápido.
- Útil para algunas entidades del modelo.

Desventajas:

- Puede generar archivos o funciones que puede que nunca se utilicen.
- Sin una comprensión de cómo funciona un CRUD por “detrás”, luego puede ser difícil adaptar uno a las necesidades de mi aplicación.