Desarrollo ágil de Software

Actividad: Sinatra (estimado 20 min)

Objetivo

- Crear una nueva aplicación Sinatra compartida con mi equipo a partir de un template
- Aprender algunos concepts mínimos en Sinatra http://sinatrarb.com/intro.html

Prerequisitos

Tener entorno VS Code+docker y usuario en github

Pasos a seguir

1. Tomar el template (en docker, **bash-5.0#**)

```
git clone https://github.com/kleer-la/template2_6_5 kbotapp
cd kbotapp
```

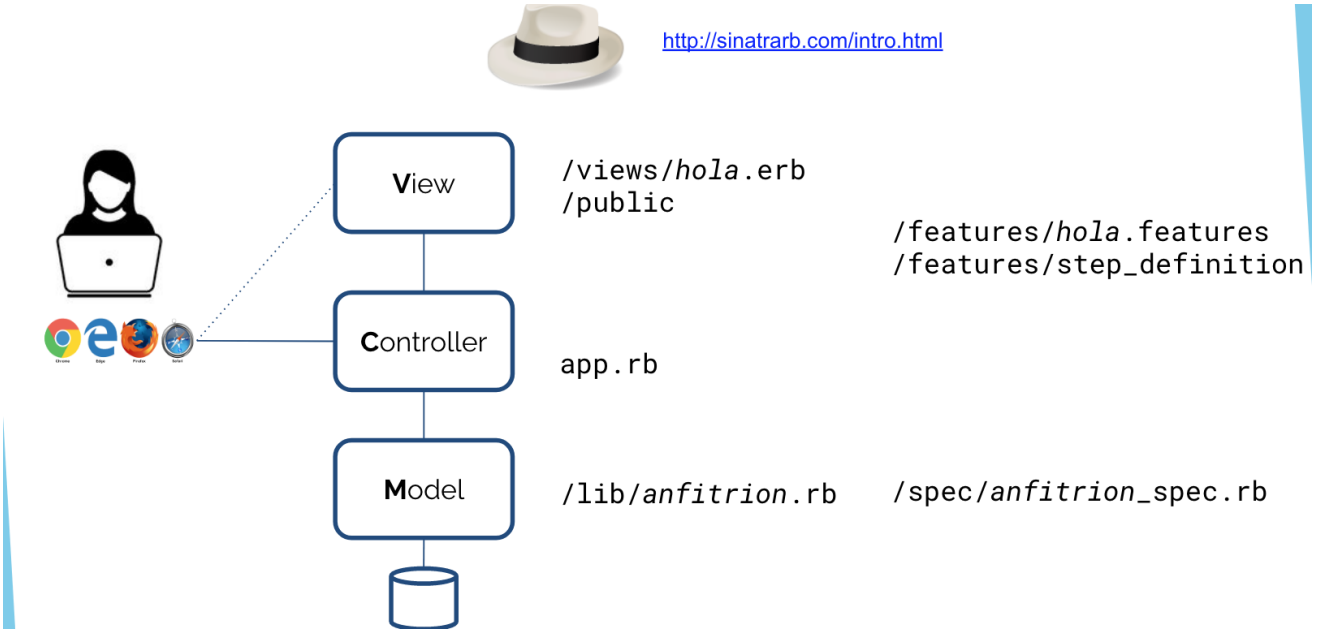
2. Ejecutar la app

```
ruby app.rb
```

   Iniciará un servidor web local, accesible desde tu browser en el puerto localhost:4567

3. Acceder con el navegador `localhost:4567`
   Verás el mensaje **hola mundo**
4. Detener el servidor web (Ctrl+C)
5. Editar el archivo app.rb, cambiando para que salude a otra persona. Volver a ejecutar el servidor y ver el cambio en el navegador

6. Sinatra propone una arquitectura Model-View-Controller.



7. En este momento, el saludo está en el Controller, tenemos que moverlo a la view. Haremos lo mismo.
   En app.rb debe quedar:

```
require 'sinatra'
require './config'

get '/' do
   erb :index
end
```

   `erb` (Embedded RuBy) transforma un template en texto. Es estándar de Ruby. En este caso, Sintatra tiene la convención que las las vistas están en `/views`, por lo que busca el template `/views/index.erb`

```
hola, mundo
```

   Ejecutar la app

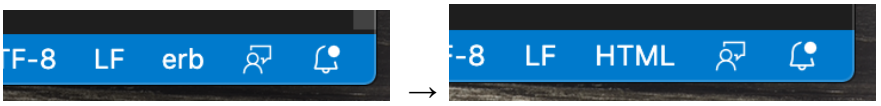
```
ruby app.rb
```

   Acceder con el navegador localhost:4567
   Verás el mensaje **hola mundo**
8. Haciendo un template como html

```
<!DOCTYPE html>
```

```
<html lang="en">
   <head>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <title>Anfitrión</title>
   </head>
   <body>
            hola, mundo
</body>
</html>
```

Tip: Puedes cambiar el tipo de archivo de erb a html, para aprovechar los snippets de código



Luego, en el editor, al poner html te dará la opción html:5 que completa la estructura (y otras etiquetas html)

9. Ahora vamos a recibir el nombre, para saludar personalizado.
   Entonces vamos a agregar un formulario (dejando el resto sin modificar). Usamos el verbo HTTP POST porque pensamos modificar el estado de la aplicación (estilo RESTful).

```
       hola, mundo
       <form action="/" method="post">
           <input type="text" name="nombre" id="nombre">
           <input type="submit" value="Saludar">
       </form>
```

10. Probamos accediendo a `localhost:4567`
    Completamos el campo de nombre y apretamos el botón saludar.
11. ¡Falla!
    No implementamos aún el código que maneja el pedido (request) correspondiente al POST en la ruta `/`
    Por suerte Sinatra me da una pista de cómo implementarlo

    Try this:

    ```
    post '/' do
      "Hello World"
    end
    ```

12. Manejar un dato de formulario
    Cambiaremos `app.rb`

```
require 'sinatra'
```

```ruby
require './config'

get '/' do
   erb :index
end
post '/' do
   @nombre= params['nombre']
   erb :index
end
```

Notar que:
- Tenemos dos manejos, una por cada verbo HTTP, de la ruta `/`
- Reusamos la misma vista `index`

13. Mostramos en la vista un dato que viene del controller

```erb
   <body>
       hola, <%= @nombre%>
       <form action="/" method="post">
```

Notar que:
- La variable definida en `app.rb` que empieza con @ tiene visibilidad dentro de template.
- `<%= %>` es la forma de mostrar un valor en la página.

14. Probamos: Detener el servidor (Ctrl+C), iniciarlo (`ruby app.rb`) y acceder a `localhost:4567`
Los cambios en archivos `.rb` requieren reiniciar el servidor