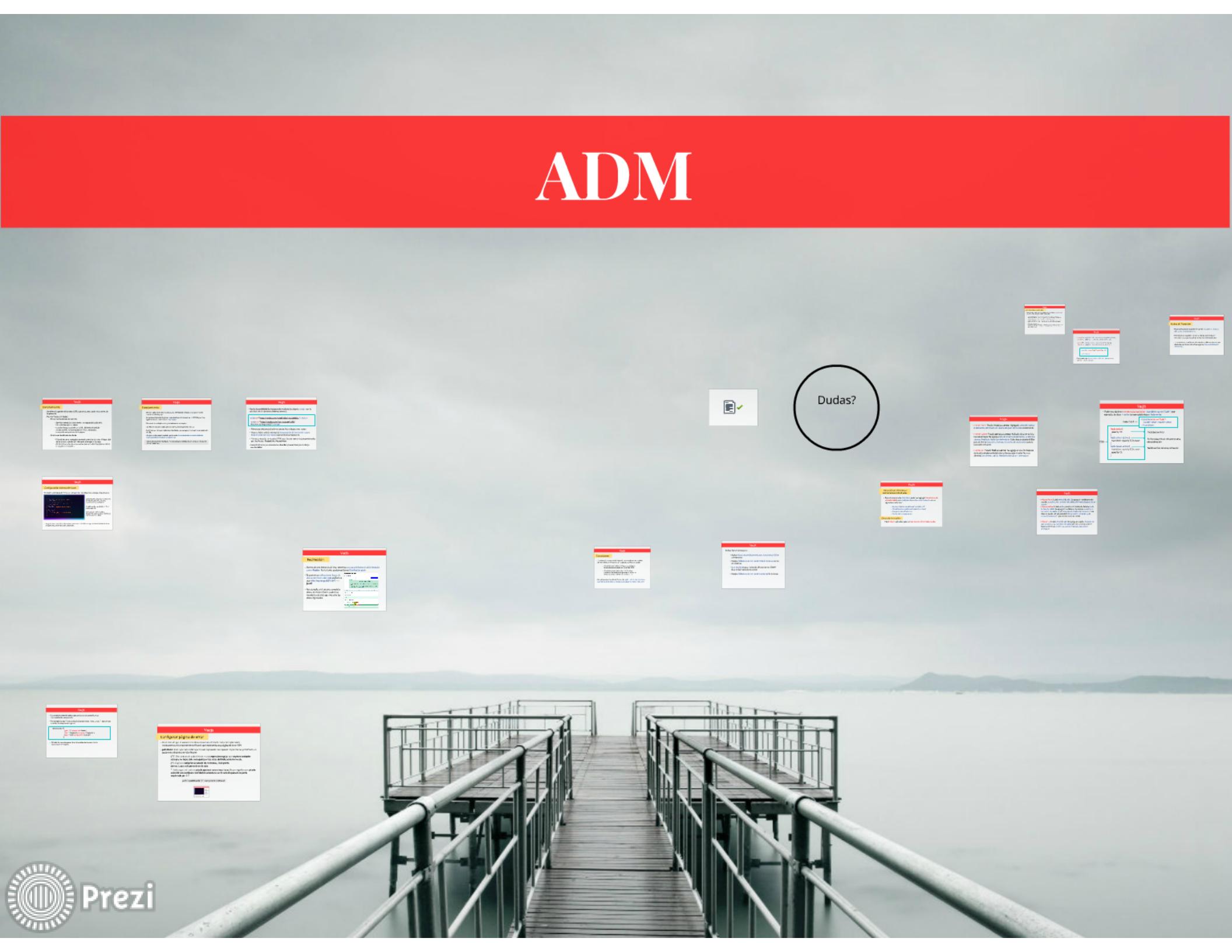


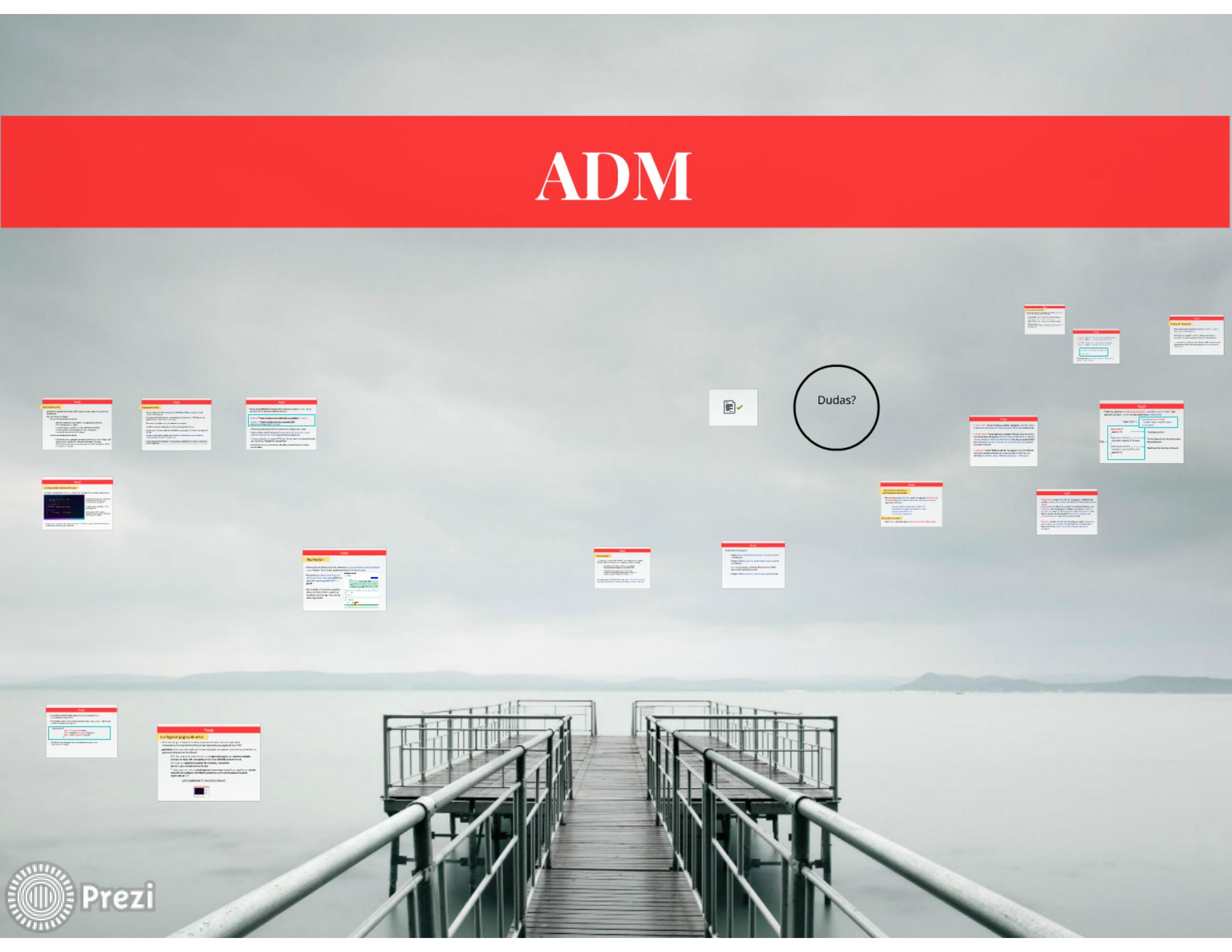
ADM

Dudas?



ADM

Dudas?



Enrutamiento

- Se refiere a la gestión de las rutas o URLs que un usuario puede visitar dentro de la aplicación.
- Hay 2 enfoques principales :
 - Enrutamiento del lado del servidor :
 - Servidor maneja las rutas y envia una respuesta basada en la URL solicitada por el usuario.
 - Cuando el usuario accede a una URL diferente, el servidor genera y envía una nueva página HTML al navegador, recargando completamente la página.
 - Enrutamiento del lado del cliente.
 - El JavaScript en el navegador asume el control de las rutas. Al hacer click en un enlace, el JavaScript intercepta la navegación y carga dinámicamente los datos necesarios para actualizar la página actual sin recargarla por completo.

Enrutamiento

- Útil para aplicaciones de una sola página (SPA) donde se busca una experiencia de usuario más fluida y ágil.
- La documentación de Vuejs nos recomienda para la mayoría de las SPA (Single-Page Application), el uso de la librería **vue-router**
- Nos permitirá múltiples vistas y transiciones entre páginas
- Vue Router está construido sobre el sistema de componentes de Vue.
- Configuramos rutas para indicarle a Vue Router qué componentes mostrar para cada ruta de URL.
- Vue.js es un framework modular, por lo tanto **el enrutamiento no está incluído de manera predeterminada en el core de vue.**
 - *Vue.js 3 funciona con Vue Router 4 y necesita que se trabaje con un live Server como el de Visual Studio Code*

VueJS

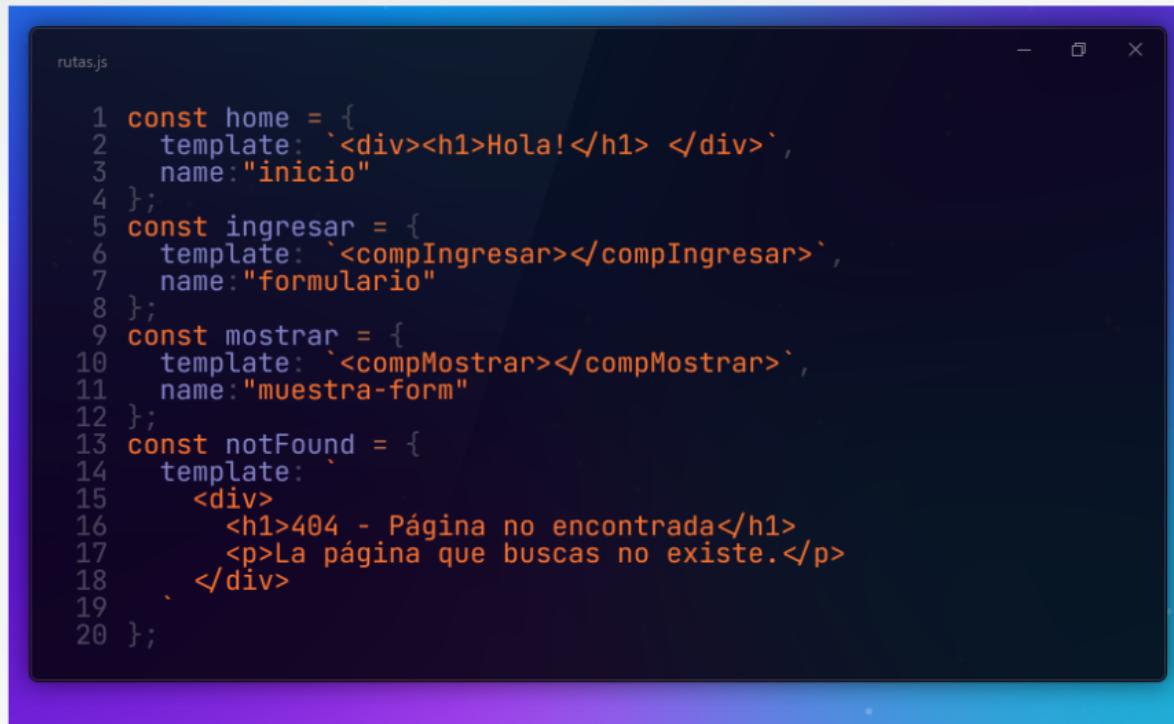
- Nos da la posibilidad de incorporarlo mediante la etiqueta `<script>` con la ruta hacia el cdn (content delivery network) :

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>  
<script src="https://unpkg.com/vue-router@4.0.15/  
dist/vue-router.global.js"></script>
```

- Primero se referencia al archivo core de Vue y luego a vue-router
- Nuestro index tendrá entonces la **incorporación del archivo de vuejs** y luego el **script con vue-router**, caso contrario arrojará error.
- Vamos a necesitar en nuestro HTML usar dos componentes proporcionados por Vue Router, **RouterLink** y **RouterView**.
- RouterLink para que renderice los **vínculos** y RouterView para trabajar con las **vistas**

Configuración sistema de rutas

- En el archivo de **rutas.js**, definimos los componentes que utilizaremos a lo largo de la aplicación.



```
rutas.js

1 const home = {
2   template: `<div><h1>Hola!</h1> </div>`,
3   name:"inicio"
4 };
5 const ingresar = {
6   template: `<compIngresar></compIngresar>`,
7   name:"formulario"
8 };
9 const mostrar = {
10   template: `<compMostrar></compMostrar>`,
11   name:"muestra-form"
12 };
13 const notFound = {
14   template:
15     <div>
16       <h1>404 - Página no encontrada</h1>
17       <p>La página que buscas no existe.</p>
18     </div>
19   };
20 };
```

Cada **constante** representa un componente de Vue que asociaremos a una ruta específica de nuestra aplicación.

template: contiene la estructura HTML y/o su componente

name asigna un nombre legible al componente, lo cual es útil para identificarlo y referenciarlo en el código

- Luego debemos asociarlos con las rutas de nuestra aplicación. Esto se logra mediante la definición de un **arreglo de rutas dentro de la constante routes**.

VueJS

- Cada objeto dentro de routes representa una ruta específica y su correspondiente componente.
- Por ejemplo, la ruta '/' está asociada al componente home, la ruta '/ingresar' está asociada al componente ingresar

```
const routes = [  
    { path: '/', component: home },  
    { path: '/ingresar', component: ingresar },  
    { path: '/ver', component: mostrar }  
]
```

- El **valor** de cada componente es el nombre de cada constante anteriormente definida.



Vuejs

Configurar página de error

- En el caso de que el usuario acceda a una ruta no definida en nuestra aplicación, mostraremos el componente notFound, que representa una página de error 404.
- **pathMatch**: Esta expresión indica que la ruta capturada corresponde al parámetro pathMatch, un parámetro dinámico en Vue Router.
 - (.*)* : Este patrón de coincidencia es una **expresión regular** que **captura cualquier ruta que no haya sido manejada por las rutas definidas anteriormente**.
 - (.*): Captura **cualquier secuencia de caracteres, incluyendo barras /, que esté presente en la ruta**.
 - * : Indica que este patrón puede aparecer cero o más veces, lo que significa que **puede coincidir con cualquier cantidad de caracteres en la ruta después de la parte capturada por (.*).**

```
path: '/:pathMatch(.)*', component: notFound
```



Vuejs

```
rutas.js

1 // Instanciamos el Router con las routes.
2 // Crear enrutador
3 const router = VueRouter.createRouter({
4   history: VueRouter.createWebHistory(),
5   routes
6 });
7
8 // Crear la aplicación Vue
9 const app = Vue.createApp({});
10
11 // Montar el enrutador en la aplicación Vue
12 app.use(router);
13
14 app.component('compIngresar', compIngresar); // Registra el
componente CompIngresar
15 app.component('compMostrar', compMostrar); // Registra el
componente CompMostrar
16
17 // Montar la aplicación en el elemento con id="app"
18 app.mount('.contenedor');
```

- La const router nos permite crear el enrutador:
VueRouter.createRouter. Es una función de Vue Router y nos permite configurar las rutas y el comportamiento de la navegación en nuestra aplicación.
- Modo historial:
createWebHistory() : Método para configurar el modo de historial en Vue Router. Este método configura el enrutador para utilizar la API de historial HTML5,

app.use(router). Esto permite utilizar las rutas definidas en el enrutador y renderice los componentes correspondientes según la URL actual.

- Luego registramos los componentes y por último montamos la aplicación

Redirección

- Dentro de una instancia de Vue, tenemos acceso a la instancia del enrutador como `$router`. Por lo tanto, podemos llamar `this.$router.push`.
- Si queremos redireccionar luego de una acción hacia otra vista, podemos usar `this.$router.push("/ver") --> (path)`
- Por ejemplo, si el usuario completa datos, al enviar el form, podemos mandarlo a la vista que muestra los datos ingresados.

Ejemplo Router

Ruta actual: /ver

Home

Ingresar

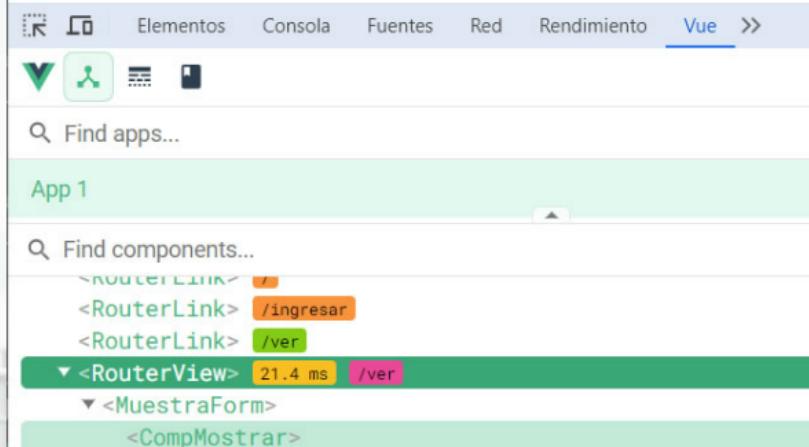
Ver

<compMostrar> 814.22 × 188.18

Datos guardados

Película: It, Comentario: Excelente película : Temática: - Terror- Ciencia Ficción, 1990

Película: El Conjuro, Comentario: Mucho suspenso, muy interesante : Temática: Ciencia Ficción- Otro, Estreno: 2013



Transiciones

- Vue ofrece dos componentes integrados que pueden ayudar a trabajar con transiciones y animaciones en respuesta a cambios de estado:
 - **<transition>** para aplicar animaciones cuando un elemento o componente entra y sale del DOM
 - **<transition-group>** para aplicar animaciones cuando un elemento o componente se inserta, se elimina o se mueve dentro de una lista v-for.
- Vue ofrece una variedad de formas de aplicar efectos de transición cuando los elementos se insertan, actualizan o eliminan del DOM.

VueJS

Incluye herramientas para:

- Aplicar **clases automáticamente para transiciones CSS** y animaciones
- Integrar **bibliotecas de animación CSS** de terceros, como Animate.css
- **Usar JavaScript** para manipular directamente el DOM durante los hooks de transición
- Integrar **bibliotecas de animación JavaScript** de terceros,

VueJS

Transición de elementos / componentes individuales

- Con el componente **transition**, podemos agregar **transiciones de entrada/salida** para cualquier elemento o componente en los siguientes contextos:
 - *Representación condicional (usando v-if)*
 - *Visualización condicional (usando v-show)*
 - *Componentes dinámicos*
 - *Nodos raíz componentes*

Clases de Transición:

Hay **6 clases** aplicadas para las **transiciones de entrada/salida**.

VueJS

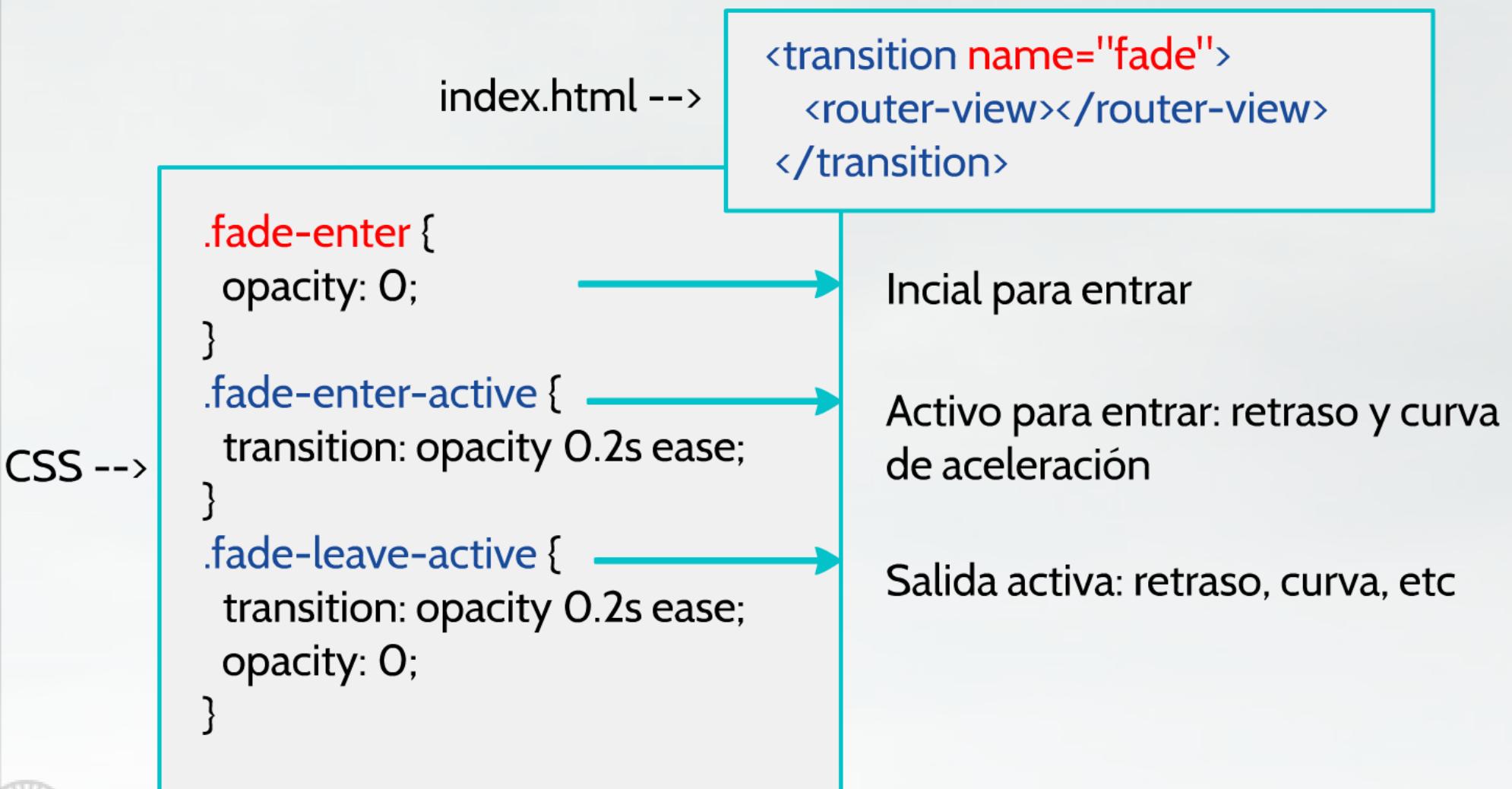
- **v-enter-from**: Estado **inicial** para **entrar**. Agregado **antes de insertar el elemento, eliminado un cuadro después de insertar** el elemento.
- **v-enter-active**: Estado **activo** para **entrar**. Aplicado durante toda la fase de entrada. Se agrega **antes de insertar el elemento, se elimina cuando finaliza la transición/animación**. Esta clase se puede utilizar para definir la **duración, el retraso y la curva de aceleración** para la transición entrante.
- **v-enter-to**: Estado **final** para **entrar**. Se agrega un cuadro después de insertar el elemento (al mismo tiempo que v-enter from se elimina), **se elimina cuando finaliza la transición / animación**.

VueJS

- **v-leave-from:** Estado inicial de salir. Se agrega inmediatamente cuando se activa una transición de salida, eliminada después de un cuadro.
- **v-leave-active:** Estado activo para la salir. Aplicado durante toda la fase de salida. Se agrega inmediatamente cuando se activa la transición de salida, se elimina cuando finaliza la transición. Esta clase se puede utilizar para definir la duración, el retraso y la curva de aceleración para la transición de salida.
- **v-leave-to:** Estado final de salir. Se agrega un cuadro después de que se activa una transición de salida (al mismo tiempo que v-leave se elimina), se elimina cuando finaliza la transición/animación.

VueJS

- Podemos darle un **nombre a la transición** `<transition name="fade">` por ejemplo, la clase `v-enter` se reemplazaría por `.fade-enter`.



Modos de Transición

- El comportamiento predeterminado de `<transition>`: entrar y salir ocurre simultáneamente.
- Esto funciona muy bien, como cuando los elementos en transición se colocan absolutamente uno encima del otro:
- Las transiciones simultáneas de entrada y salida no siempre son deseables, por lo que Vue ofrece algunos modos de transición alternativos :

VueJS

- **in-out:** Las transiciones de elementos nuevos primero, luego, cuando se completa, el elemento actual se transiciona.
- **out-in:** El elemento actual se transfiere primero, luego, cuando se completa, el nuevo elemento se trasiciona.

```
<transition name="fade" mode="out-in">  
  </transition>
```

- Esto permite que la transicion se realice sin saltos entre los elementos transicionados.

VueJS

Aplicar transition a router-view

- Esto permite que el usuario comprenda el comportamiento del sistema de manera más predecible y evita sorpresas con transiciones inesperadas.
 - **v-slot="{ Component }"** : Es una directiva de Vue que permite acceder al componente que se está renderizando en ese momento dentro de <router-view>. Al usar { Component }, estamos extrayendo el componente actual que se está mostrando en la ruta.
 - **<transition name="rutas"></transition><transition>** : Envuelve al componente que se está mostrando en la ruta.name="rutas" : Indica que se aplicará un efecto de transición llamado "rutas".
 - **<component :is="Component" />**: Se usa el componente dinámico de Vue para renderizar dinámicamente el componente actual de la ruta. :is="Component" : Indica que el componente a renderizar será el que esté almacenado en la variable Component.

Dudas?



ADM

Dudas?

