

ADM



Dudas?

ADM



Dudas?



Prezi

VueJS

Instalación Vue-Router

- Desde la consola vamos a crear un proyecto que tenga `vue-router`
- Sobre la partición o ruta que seleccionen en sus pc, crearemos el proyecto llamado "enrutamiento"

```
c:\tucarpeta\npm create vue@latest
```

- Configuramos las características que tendrá (Eslint- Prettier - Vue-Router) marcando **Yes**:

```
Vue.js - The Progressive JavaScript Framework

✓ Project name: ... enrutamiento
✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes ✓
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? ... No / Yes
✓ Add Prettier for code formatting? ... No / Yes
✓ Add Vue DevTools 7 extension for debugging? (experimental) ... No / Yes
```

VueJS

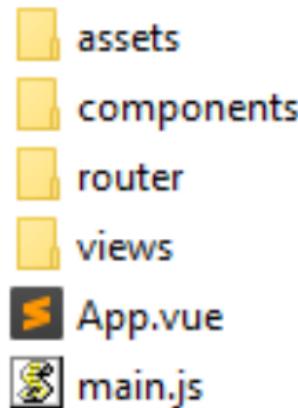
- Luego nos indicará que comandos tenemos que escribir para poder correr este nuevo proyecto:

```
Done. Now run:  
  
cd enrutamiento  
npm install  
npm run format  
npm run dev
```

- Una vez ejecutados los comandos, el último comando nos muestra la dirección url que debemos colocar en el navegador para tener nuestro proyecto en el servidor de desarrollo.
- Por defecto: ***localhost://5173***
- El proyecto base que tenemos tiene agregado en la interfaz una navegación: **Home** y **About**

VueJS

- Con el comando `dir`, podremos ver la estructura de archivos generada.
- Si examinamos la estructura de archivos veremos que se crearon 2 carpetas la carpeta `views` y `router`



- Dentro de `views` crearemos *las vistas* que representan los elementos visuales. Usando Templates y componentes
 - Son responsables de mostrar datos y responder a las interacciones del usuario.
 - Cada **vista** suele estar asociada a *una ruta específica y puede contener componentes, datos y lógica* relacionada con esa parte de la interfaz.
-
- Dentro de la carpeta `router`, en el archivo `index.js` configuraremos las rutas

Enrutamiento (Parte II)

- Vuejs, al igual que otros frameworks nos permite **pasar parámetros cuando definimos una ruta**
- Con vue-router podemos usar un **segmento dinámico** en el camino para lograr eso:
- La configuración de la ruta la tenemos que hacer en el archivo index.js

```
const routes=[  
  {  
    path: '/peliculas/:id/:estreno',  
    name: 'Peliculas',  
    component: PeliculasComponent  
  },  
]
```

VueJS

- Un segmento dinámico se denota con dos puntos :
- Cuando se hace coincidir una ruta, el valor de los segmentos dinámicos se expondrá como **this.\$route.params** en cada componente.

Enrutamiento con Parámetros



Top 3 Películas super taquilleras

The Avengers [Ver Detalle](#)

Los juegos del hambre [Ver Detalle](#)

Avatar [Ver Detalle](#)

- Mas allá de la navegación principal, podemos tener una navegación anidada dentro de nuestra aplicación.

El componente App se renderizará mostrando el contenido de Detalles.

VueJS

```
export default {
  name: 'PeliculasComponent',
  data:function(){
    return{
      pelis:[
        {id:1, titulo: "The Avengers", estreno:2012},
        {id:2, titulo: "Los juegos del hambre", estreno:2012},
        {id:3, titulo: "Avatar",estreno:2009}
      ]
    }
  },
}
```

- De toda la información que posee el **array peliculas**, al usuario solo le mostraremos la información del **titulo**
- El **template del component** tendrá un **router-link** por el cual le **pasaremos los datos hacia otra vista**.



VueJS

```
<template>
  <div class="detalles">
    <ol reversed>
      <li v-for="item in pelis" :key="item.id">{{item.titulo}}
        <a href="/peliculas/" + item.id + '/' + item.estreno" > Ver Detalle </a>
      </li>
    </ol>
  </div>
</template>
```

- Cuando el usuario haga click en **Ver Detalle**, pasaremos **por parámetro las propiedades** que tenga o que queramos mostrarle al usuario

VueJS

Componente Peliculas

The screenshot shows a web browser window with the URL `localhost:8080/peliculas/1/2012`. The page displays a movie detail for "The Avengers". At the top, there are three pink buttons labeled "Home", "Ingresar", and "Ver". Below them, the text "1 - 1" and the movie title "The Avengers" are shown. A large circular image of the movie's cast is on the left. To the right of the image is a detailed plot summary and cast information. At the bottom, it says "Fecha de Estreno: 2012" and has a "Volver" button.

1 - 1

The Avengers

Cuando un enemigo inesperado amenaza la seguridad del planeta y de sus habitantes, Nick Fury (Samuel L. Jackson), director de SHIELD, monta un dispositivo con todos los hombres capaces de preservar a la humanidad del caos. El enemigo es tan poderoso, que necesita que todos los superhéroes luchen juntos y formen un equipo compacto. Ellos serán: el Capitán América (Chris Evans), Thor (Chris Hemsworth), Iron Man (Robert Downey Jr.), Hulk (Mark Ruffalo), Ojo de Halcón (Jeremy Renner) y la Viuda Negra (Scarlett Johansson).

Basado en los cómics de Marvel, el film está dirigido por Joss Whedon (Buffy Cazavampiros, Angel), que ya practicó el género en su primera película, Serenity. Completan el reparto Gwyneth Paltrow (que repite su papel en Iron Man), Paul Bettany (que pone la voz a Jarvis), Cobie Smulders (Cómo conocí a vuestra madre), como Maria Hill, Stellan Skarsgård (Rompiendo las olas), como el profesor Erik Selveng y Tom Hiddleston en el papel que interpretó en Thor.

Fecha de Estreno: 2012

Volver

parámetros

Interpolación del ID

El html tendrá las evaluaciones para mostrar la pelicula segun el id pasado por parametro

```
<div v-if="i=='1'">
```

```
    <!-- html -->
```

```
<div v-else-if="i=='2'">
```

```
    <!-- html -->
```

```
<div v-else>
```

VueJS

- Como toda **variable** que necesitemos en un componente, necesitamos **declararla** en el data de la function

```
i:"", e:""
```

- Dentro de nuestros **métodos**, tendremos una **función** que recuperará los valores pasados por parámetro.
- Asociamos nuestras variables inicializadas en data con los parámetros recibidos

```
this.i = this.$route.params.id;  
This.e = this.$route.params.estreno
```

- Esta función se ejecutará el hook **mounted**

VueJS

- Para **volver a la pantalla anterior** le ofrecemos al usuario un botón que trabaja con la instancia del enrutador **\$route**.

```
<button @click="$router.go(-1)">Volver</button>
```

- Este **método** toma un **entero como parámetro** que indica **cuántos pasos hay que avanzar o retroceder en la pila de historial**.

Editar información

- El **paso de información por parámetros** nos puede facilitar la tarea en la **edición** de información.



VueJS

- Si el usuario tiene la posibilidad de generar **su propia información**, la **edición** es una funcionalidad necesaria, al igual que **no dependa de la red** para manipularla.

[Ingresar](#)

[Ver](#)

Titulo

Comentario

Seleccionar

Terror
Animacion
Ciencia Ficcion
Drama

Estreno

Guardar

Router

[Home](#)

[Ingresar](#)

[Ver](#)

Datos guardados

Nombre: Bill y Ted, **Comentario:** Proxima a estrenar ! : **Temática:** Ciencia Ficcion , 2020

[Editar](#)

Nombre: Black Widow, **Comentario:** Probablemente no la veamos este año :(: **Temática:** Ciencia Ficcion Otro , 2022

[Editar](#)

Editar

Titulo

Bill y Ted

Comentario

Proxima a estrenar !

Seleccionar

Terror
Animacion
Ciencia Ficcion
Drama

Estreno

2020

Guardar

Vuejs

- Cada vez que el usuario ingrese información iremos guardando en localStorage con el key datos.
- Con el método push vamos a ir incorporando los objetos al array que contendrá por cada pelicula, un objeto.
- Por medio de la función **JSON.stringify** convertiremos la info para que se pueda almacenar
- Los valores que guardaremos serán: **titulo**, **comentario**, **selected**, **estreno** y **fecha**.



Vuejs

- La fecha nos servirá para establecer una identificación única por cada pelicula.
- Por cada pelicula generaremos un nuevoObj

```
nuevoObj={ comentario: this.form_data.comentario,  
           titulo: this.form_data.titulo,  
           selected: this.form_data.selected,  
           estreno: this.form_data.estreno,  
           fecha: new Date().getTime() }
```

fecha: new Date().getTime() --> Nos devolverá el tiempo en milisegundos cuando se guarde este objeto, y nos sirve como un id irrepetible para manipularlo.

Vuejs

- O Podemos usar el método `Object.assign()` y asignarle esta nueva propiedad de fecha al objeto

```
form_data = Object.assign({}, form_data, { fecha: new Date().getTime() })
```

- Este método copia todas las propiedades y valores de uno o más objetos a un objeto destino y devuelve el objeto destino.

The screenshot shows a browser's developer tools console. At the top, there are standard toolbar icons: a play button, a stop button, a refresh button, and a 'top' button. To the right of these are dropdown menus for 'Filter' and 'Default levels', and a settings gear icon. Below the toolbar, the console output is displayed. A red arrow points to a line of code: `{titulo: "Pesadilla en Elm Street", comentario: "Clasico de terror", selected: Array(1), estreno: "1984", fecha: 1620737394497}`. Above this line, the text `ingresar.js:57` is displayed in green, indicating the file and line number where the code was run.

- Cuando se guarde redireccionamos a la vista `/ver`, para que se muestren los datos guardados en `localStorage`.

VueJS

- En el **hook mounted** llamaremos a la función `ver_local()`, que tenemos desarrollada en la **propiedad methods**
- Esta función será la encargada de **evaluar si existe el key datos** en `localStorage`, en cuyo caso parsearemos la info para mostrar.
- **Y si no existe el key datos**, le mostraremos al usuario **un mensaje** avisándole que todavía no guardó info.

```
<div class="lista" v-for="item in local" :key="item.fecha">
  <p><span>Nombre: </span>{{ item.titulo }} </p>
  <p><span>Comentario: </span>{{ item.comentario }} </p>
  <p><span>Temática: </span> <span class="normal" v-for="(x, index) in
  item.selected" :key="index"> {{x}} </span></p>
  <p><span><Estreno: </span>{{ item.estreno }}</p>
  <router-link :to="/editar/" + item.titulo + '/' + item.comentario + '/' +
  item.selected + '/' + item.estreno + '/' + item.fecha " > Editar </router-link>
```

VueJS

- Por medio `router-link` indicaremos la vista y los parámetros para que el usuario pueda editar.
- Lo llevaremos a un `formulario` que permita `levantar esos datos`, salvo la fecha que no se mostrará.
- Dentro de la `función data` asociaremos `los parámetros a las propiedades de un nuevo Objeto`

```
nuevoObj: { titulo : this.$route.params.titulo,  
            comentario : this.$route.params.comentario,  
            selected : this.$route.params.selected.split( "," ),  
            estreno : this.$route.params.estreno,  
            fecha : this.$route.params.fecha }
```

VueJS

- Para que podamos mostrar estos valores los v-model deben tener como valor este objeto y la propiedad (Por ej: **v-model= "nuevoObj.titulo"**)
- En el caso del select, al recibir múltiples valores, el v-model espera que le pasemos un array, caso contrario nos arrojará error y no mostrará lo que el usuario seleccionó previamente.
- El método split(", ") devuelve el nuevo array, con las posiciones que encuentre de coma.

```
✖ ► [Vue warn]: <select multiple v-           vue.js:634  
    model="nuevoObj.selected"> expects an Array value for  
    its binding, but got String  
  
    found in  
  
      ---> <Edicion>  
            <Edicion>  
            <Root>
```

VueJS

- Cuando el usuario **guarde las modificaciones**, tenemos que saber a qué elemento del localStorage tenemos que afectar para reemplazar el dato original.

1) Obtenemos lo guardado en localStorage parseado.

2) Recorremos lo que tiene la variable que trajo la info del key dato.

3) Evaluamos en el **for** por cada iteración si la propiedad fecha coincide con la fecha de este nuevo objeto que el usuario editó

4) Si coincide, lo eliminamos.

```
this.local= JSON.parse( localStorage.getItem ("dato"))
for (var i=0; i < this.local.length; i++){
    if (this.local[i].fecha == this.nuevoObj.fecha ) {
        this.local.splice(i, 1);
    }
}
```

VueJS

5) Mediante el método *push* agregamos este nuevo objeto a la variable

6) Lo guardamos en `localStorage` pasado a string *de nuevo*

```
this.local.push(this.nuevoObj);
```

```
localStorage.setItem("dato", JSON.stringify(this.local))
```

Dudas?



ADM

The slide features a background image of a long wooden pier extending into a body of water under a cloudy sky. Overlaid on this background are several small, semi-transparent windows, each containing a different type of digital interface or form. These include:

- A top row of three windows labeled 'Form'.
- A middle row of two windows labeled 'Form'.
- A bottom row of four windows labeled 'Form'.
- A single window labeled 'Vista' in the center.
- A single window labeled 'Vista' on the right.
- A single window labeled 'Vista' at the bottom.
- A single window labeled 'Vista' on the far left.
- A bottom-left window labeled 'Vista'.
- A bottom-right window labeled 'Vista'.

Each window displays a different type of digital interface, such as a dashboard, a form with various input fields, or a list of items. Some windows have specific annotations or highlights, such as a red box around certain text or a green checkmark icon.

Dudas?



Prezi