Vuex

Almacenamiento de estado en Vuex

{desafío} latam_

Implementar un aplicativo web utilizando Vuex para el almacenamiento del estado de los componentes y dar solución a un problema.

- Unidad 1: Vue Router
- Unidad 2: Vuex
- Unidad 3: Firebase
- Unidad 4: Pruebas Unitarias y end-to-end en un entorno Vue

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Describe las características y utilidad de Vuex para el almacenamiento de estados de un aplicativo web.*
- *Implementa un módulo Vuex para el almacenamiento del estado de componentes que resuelven un problema.*

{desafío} latam_

Recordemos… ¿Cómo generamos comunicación entre componentes?

{desafío} latam_

¿Qué sucede si necesitamos enviar información de un componente hijo a un componente padre?

{desafío} latam_

Contexto antes de iniciar

Cuando iniciamos un desarrollo en VueJS y, a medida que generamos un componente tras otro, notaremos que hay secciones del proyecto que se deben (o deseamos) comunicar entre sí, debido a que comparten alguna característica en particular y/o cumplen alguna función de manera cooperativa.

Cuando un proyecto recién inicia, esto puede resultar sencillo de abordar. Pero a medida que el desarrollo crece y nos demandan desarrollar más funcionalidades, el asunto se puede complicar, lo que puede concluir en un proyecto difícil de mantener en el tiempo. Para hacer frente a este tipo de situaciones, disponemos de Vuex.

/* Vuex */

¿Qué es Vuex?

- Vuex se define como una librería que impone un patrón de gestión de estados para aplicaciones VueJS.
- ¿Qué nos quiere decir esto? que Vuex es una librería o extensión de VueJS que nos obliga a seguir pautas o guías predefinidas para gestionar las propiedades de los componentes en un almacén de datos centralizado.
- Por lo tanto, veamos a Vuex como un contenedor que almacenará las propiedades de nuestros componentes de manera global y luego nuestros componentes podrán acceder a estas propiedades.

¿Cuándo conviene utilizarlo?

- Utilizar el enfoque que nos propone Vuex nos puede resultar muy útil a la hora de desarrollar aplicaciones que dependan de la comunicación entre componentes.
- Analicemos el caso de un carro de compras online.

Ejercicio guiado: "Realicemos el diagrama de árbol de componentes"

{desafío} latam_

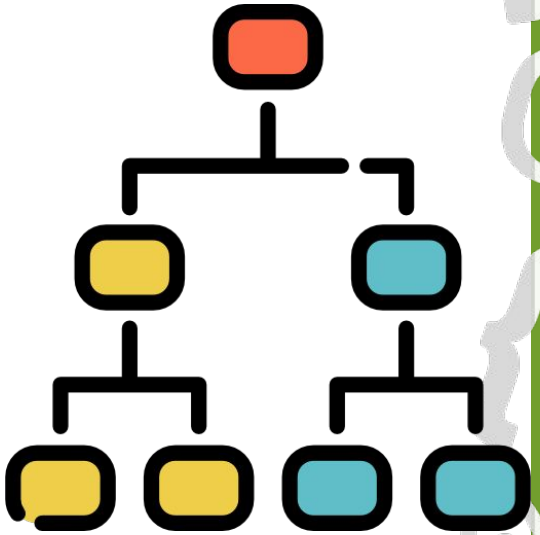
Realicemos el diagrama de árbol de componentes

Contexto

A continuación, realizaremos en conjunto el diagrama de componentes asociado a una tienda online de productos de feria. Para ello, utilizaremos un árbol de componentes de 3 niveles:

- Componente principal App.vue
- Componente ancestro Card.vue
- Componente ancestro ShoppingCart.vue

Nota: App.vue pasa los datos al componente Card y Card los envía a ShoppingCart.



¿Qué problema identificas en una aplicación que contenga 40 componentes?

{desafío} latam_

Ejemplos de utilidad de Vuex

- Utilizar el enfoque que nos propone Vuex nos puede resultar muy útil a la hora de desarrollar aplicaciones que dependan de la comunicación entre componentes.
- Podemos utilizar Vuex para gestionar los cambios y comunicación con un carrito de compras online.
- También generar un sitio de reserva de horas y una aplicación de monitoreo.
- Además, es muy útil para gestionar los datos que se generen en un formulario y para los procesos de autenticación en una web.

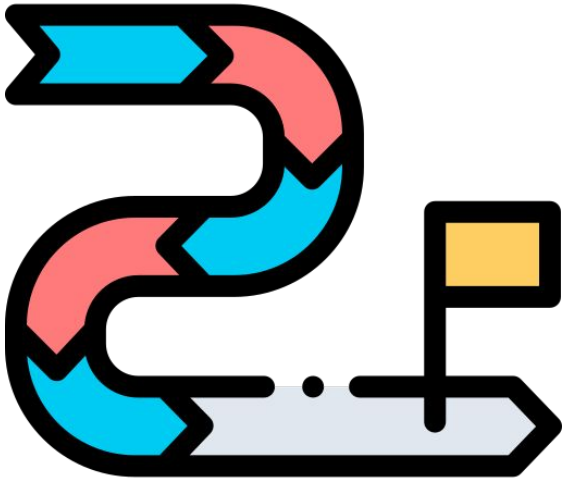
/* Estructura de una aplicación con Vuex como dependencia */

Ejercicio guido: "Estructura de directorios Vuex"

{desafío} latam_

Estructura de directorios Vuex

Contexto



A continuación, crearemos un nuevo proyecto con Vite, y luego incorporaremos Vuex a nuestro proyecto. El objetivo es identificar las dependencias que se incorporan, el orden de los directorios y cómo se habilita Vuex en nuestra aplicación.

¡Sigue los pasos!

Sigue los pasos…

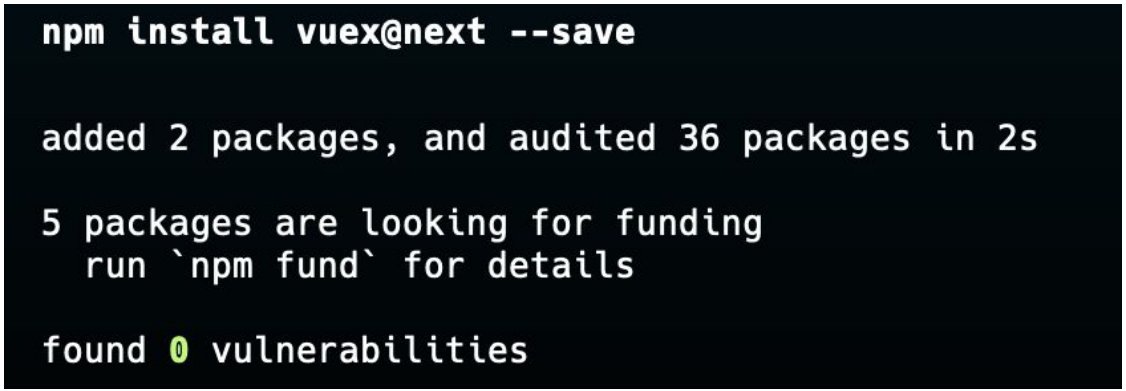
- **Paso 1:** Creamos un nuevo proyecto con Vite y le asignamos el nombre bases-vuex.
  Además seleccionamos la opción de agregar Router

```
npm create vue@latest

> npx
> create-vue


Vue.js - The Progressive JavaScript Framework

✔ Project name: … bases-vuex
✔ 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 Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in /Users/ale/Desktop/ADL/Vue/proyecto/bases-vuex...

Done. Now run:

  cd bases-vuex
  npm install
  npm run dev
```

Sigue los pasos…

- **Paso 2:** Con el proyecto ya creado, instalamos vuex usando el siguiente comando:

```
npm install vuex@next --save
```



Sigue los pasos…

- **Paso 3:** Abrimos el proyecto en Visual Studio Code y nos dirigimos al archivo package.json.

```
"dependencies": {
  "vue": "^3.4.29",
  "vue-router": "^4.3.3",
  "vuex": "^4.0.2"
},
"devDependencies": {
  "@vitejs/plugin-vue": "^5.0.5",
  "vite": "^5.3.1"
}
```

Sigue los pasos…

- **Paso 4:** Seguidamente, ubicamos el directorio /src y creamos una nueva carpeta llamada /store y dentro creamos un archivo llamado index.js.

```
import { createStore } from "vuex";

export default createStore({
  state: {
    count: 0,
  },
  mutations: {
    increment(state) {
      state.count++;
    },
  },
  actions: {
    // Aquí irían tus acciones
  },
  modules: {
    // Aquí irían tus módulos
  },
});
```

Sigue los pasos…

- **Paso 5:** Abre el archivo **main.js** en la carpeta src y modifícalo para incluir el store

src > JS main.js > ...

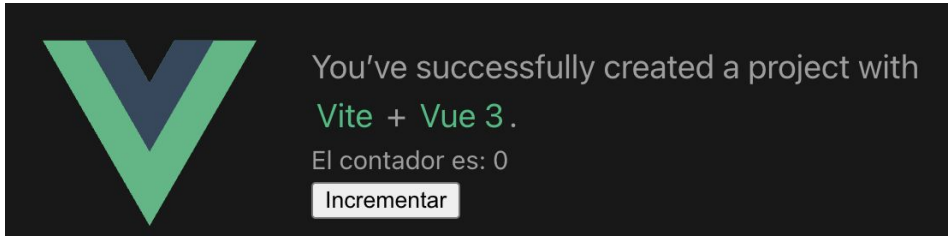
```
1   import "./assets/main.css";
2
3   import { createApp } from "vue";
4   import App from "./App.vue";
5   import router from "./router";
6   import store from "./store"; // Importa el store
7
8   const app = createApp(App);
9
10  app.use(router);
11  app.use(store); // Usa el store
12
13  app.mount("#app");
```

Hasta este punto ya sabemos cómo se integra Vuex en una aplicación, ahora veamos cómo trabajar con store.

Sigue los pasos…

- **Paso 6:** Ahora puedes acceder al store desde cualquier componente. Por ejemplo, en **HelloWorld.vue**:

```
<div>
  <p>El contador es: {{ $store.state.count }}</p>
  <button @click="$store.commit('increment')">Incrementar</button>
</div>
```



{desafío} latam_

Ejercicio:
“Ingresa dos nuevas propiedades al state”

Ingresa dos nuevas propiedades al state

Contexto

A partir del ejercicio guiado anterior, ingresa dos nuevas propiedades al state, estas serán:

- nombre: “Tu nombre”
- carrera: “La carrera que estás estudiando”

Luego de haber ingresado estas dos nuevas propiedades, muéstralas en el navegador siguiendo la sintaxis ya vista.

Recordemos que en las siguientes sesiones veremos qué es el state.

Resumen de la sesión

- Vuex es una funcionalidad que se incorpora en Vue JS para trabajar con datos ubicados en una posición global o centralizada.

- Dicha posición global, permitirá que podamos acceder a la información desde cualquier nivel en el árbol de componentes.

¿Qué fue lo que más te complicó de esta sesión?

Próxima sesión...

- *Guía de ejercicios.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales