



Tarea 5

Vue.js

Julio Andrade
Gerardo Crot
M^a Josefa Espinoza

Tabla de contenidos

01

Demo

02

Vuex y Store

03

**Interacción con
componentes**

04

**Conclusiones y
aprendizajes**



01

Demo



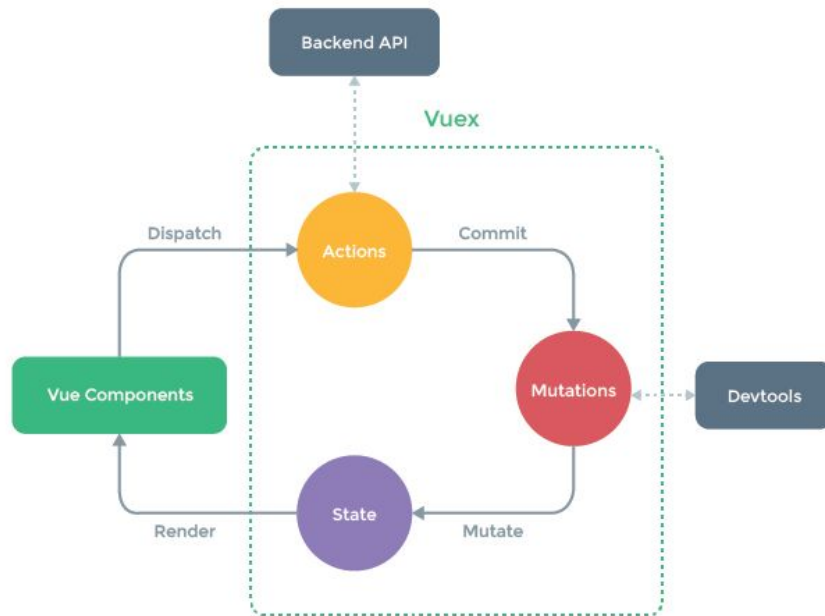
02

Vuex y Store

Vuex

Nos basamos en la última guía documentada por Vuex

Definimos en la store: estados, mutaciones, getters y actions.



Store

```
store.js

export const store = createStore({
  state() {
    return {
      count: 0,
      search: "",
      weather: {},
      searchVisibility: true,
      weatherSet: false,
    };
  },
  mutations: {
    // mutaciones no son asincronas
    changeSearch(state, value) {
      state.search = value;
    },
    changeWeather(state, weather) {
      state.weather = weather;
      console.log(state.weather);
      state.weatherSet = true;
    },
    hideSearchBar(state) {
      state.searchVisibility = false;
    },
    showSearchBar(state) {
      state.searchVisibility = true;
    },
    unsetWeather(state) {
      state.weatherSet = false;
      state.weather = "";
    },
    restoreSearch(state) {
      state.search = "";
    },
  },
});
```

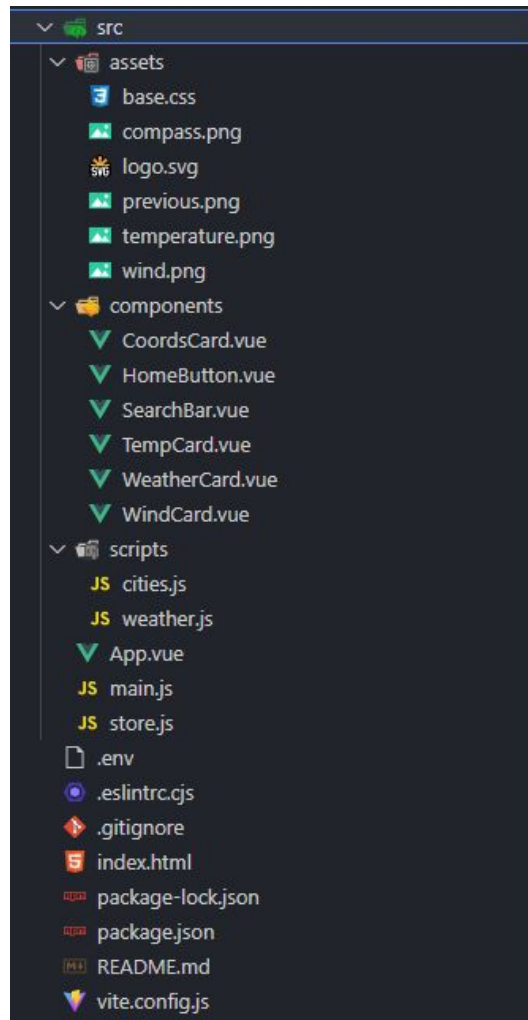
```
store.js

getters: {
  getSearch(state) {
    return state.search;
  },
  getSearchVisibility(state) {
    return state.searchVisibility;
  },
  getWeather(state) {
    return state.weather;
  },
  getWeatherSet(state) {
    return state.weatherSet;
  },
  getTempInfo(state) {
    return state.weather.main;
  },
  getCoords(state) {
    return state.weather.coord;
  },
  getWind(state) {
    return state.weather.wind;
  },
},
actions: {
  // acciones pueden ser asincronas
  async makeSearch(context, value) {
    const weather = await getWeatherFromCity(value);
    if (!weather) return null;
    context.commit("changeSearch", value);
    context.commit("hideSearchBar");
    context.commit("changeWeather", weather);
  },

  goHome(context) {
    context.commit("unsetWeather");
    context.commit("restoreSearch");
    context.commit("showSearchBar");
  },
},
});
```

Orden en el repositorio

Adicionalmente al store.js la aplicación contaba con distintos componentes y scripts para facilitar la interacción





03

Interacción con componentes

Barra de búsqueda

Usamos una condición v-show para condicionar si se muestra el componente dependiendo del estado de la aplicación mediante un getter de store.

Además de despachar una acción al apretar el botón submit.

```
SearchBar.vue

<template>
  <div id="bar" v-show="getVisibility" class="search-wrapper">
    <input
      class="input-search"
      type="text"
      v-model="search"
      placeholder="Busca un lugar"
    />
    <button type="submit" v-on:click="makeSearch">Buscar</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      search: "",
    };
  },
  methods: {
    makeSearch() {
      this.$store.dispatch("makeSearch", this.search);
      this.search = "";
    },
  },
  computed: {
    getVisibility() {
      return this.$store.getters.getSearchVisibility;
    },
  },
};
</script>
```

Tarjetas

Ahora se condiciona su renderización mediante un v-if y la propiedad computed.

Se utilizan métodos y getters para obtener la información

```
WeatherCard.vue

<template>
  <div v-if="weatherSet" class="weather-card">
    <div class="weather-icon">
      
    </div>
    <div class="weather-info">
      <p>{{ description() }}</p>
    </div>
  </div>
</template>

<script>
export default {
  methods: {
    getUrl() {
      const icon = this.$store.getters.getWeather.weather[0].icon;
      return `http://openweathermap.org/img/wn/${icon}@2x.png`;
    },
    description() {
      return this.$store.getters.getWeather.weather[0].description;
    },
  },
  computed: {
    weatherSet() {
      return this.$store.getters.getWeatherSet;
    },
  },
};
</script>
```

Funciones útiles

Estas funciones son externalizadas al funcionamiento del store y proveen la información del clima

```
weather.js

export const getWeatherFromCity = async (city) => {
  const coords = await getCoords(city);
  if (coords.message) return coords.message;
  const url = `https://api.openweathermap.org/data/2.5/weather?lat=${coords.lat}&lon=${coords.lng}&appid=${apiKey}`;
  try {
    const response = await axios.get(url);
    return response.data;
  } catch (error) {
    console.log(error);
  }
};
```

```
cities.js

import cities from "cities.json";

export const getCoords = (city) => {
  const myCity = cities.filter(
    (c) => c.name.toLowerCase() === city.toLowerCase()
  );
  if (!myCity[0]) return null;
  return { lat: myCity[0].lat, lng: myCity[0].lng };
};
```



04

Conclusiones y aprendizajes

Modularización

A medida que la aplicación crece se hace necesario modularizar los componentes del store de manera de mantener una mejor legibilidad.

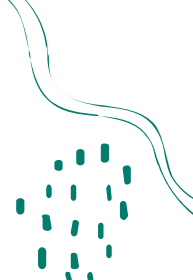

```
store.js

export const store = createStore({
  state() {
    return {
      count: 0,
      search: "",
      weather: {},
      searchVisibility: true,
      weatherSet: false,
    };
  },
  mutations: {
    // mutaciones no son asíncronas
    changeSearch(state, value) {
      state.search = value;
    },
    changeWeather(state, weather) {
      state.weather = weather;
      console.log(state.weather);
      state.weatherSet = true;
    },
    hideSearchBar(state) {
      state.searchVisibility = false;
    },
    showSearchBar(state) {
      state.searchVisibility = true;
    },
    unsetWeather(state) {
      state.weatherSet = false;
      state.weather = "";
    },
    restoreSearch(state) {
      state.search = "";
    },
  },
  getters: {
    getSearch(state) {
      return state.search;
    },
    getSearchVisibility(state) {
      return state.searchVisibility;
    },
    getWeather(state) {
      return state.weather;
    },
    getWeatherSet(state) {
      return state.weatherSet;
    },
    getTempInfo(state) {
      return state.weather.main;
    },
    getCoords(state) {
      return state.weather.coord;
    },
    getWind(state) {
      return state.weather.wind;
    },
  },
  actions: {
    // acciones pueden ser asíncronas
    async makeSearch(context, value) {
      const weather = await getWeatherFromCity(value);
      if (!weather) return null;
      context.commit("changeSearch", value);
      context.commit("hideSearchBar");
      context.commit("changeWeather", weather);
    },
    goHome(context) {
      context.commit("unsetWeather");
      context.commit("restoreSearch");
      context.commit("showSearchBar");
    },
  },
});
```



Facilidad de interacción

Nuestra opinión es que la interacción sobre todo entre componentes se facilita al tener centralizado el manejo de estados, sus cambios, acciones ,etc. En comparación a manejar dichos elementos en cada componente.



```
return this.$store.getters.getWeatherSet;
```

```
description() {  
  return this.$store.getters.getWeather.weather[0].description;  
},
```

```
goHome() {  
  this.$store.dispatch("goHome");  
},
```





Dudas o comentarios



Tarea 5

Vue.js

Julio Andrade
Gerardo Crot
M^a Josefa Espinoza