Vue: "Batalla Pokémon"

Integrantes:

- Francisco Olivares
- Gabriel Valenzuela

Primero un repaso sobre vue

- Creación de componentes
- Instanciación de componentes
- Creación de store

Creación de componentes (Pokemon.vue)

```
<template>
  <div>
    <!-- HTML del componente aquí -->
  </div>
</template>
<script>
import { mapState } from 'vuex' /*esto se explicara más adelante ;)*/
export default {
  /* PROPS A RECIBIR */
  props: {
    orientation: String
  },
  /* ESTADO INTERNO */
  data: function () {
    return {
      variable1: "hola!!!"
    }
  },
  /* METODOS COMPUTADOS (auto-actualizan) */
  computed: {
    /*...más código arriba...*/
    isEnemy() {
      return this.orientation === "front" ? true : false
    },
    isPlayer() {
      return this.orientation === "back" ? true : false
    }
  }
}
</script>
/* ESTILOS DEL COMPONENTE */
<style>
.hp_container {
  border-style: solid;
  border-width: 2px;
  border-color: rgba(0, 0, 0, 0.3);
  border-radius: 5px;
}
</style>
```

Instanciación de componentes

El componente Home ocupa a la componente Pokemon

```
<template>
  <div>
    <!-- ...más código arriba... -->
    <div id="battle_container">
      <div class="columns is-mobile" v-if="has_pressed_start">
        <div class="column is-6-desktop is-offset-3-desktop">
          <pokemon orientation="front"/> <!-- ACÁ SE INSTANCIA -->
        </div>
      </div>

      <div class="columns is-mobile" v-if="has_pressed_start">
        <div class="column is-6-desktop is-offset-3-desktop">
          <pokemon orientation="back"/> <!-- ACÁ SE INSTANCIA -->
        </div>
      </div>
    </div>
  </div>
</template>
<script>
import pokemon from "../components/Pokemon"

export default {
  components: {
    pokemon
  },
  /* ...más código abajo... */
}
</script>
```

Creación de la store (Vuex)

Como mínimo se necesitan estas 3 cosas

- estado
- acciones
- mutaciones

Store: El estado

```
const state = {
  player_pokemon: {
    hp: 0,
    name: "",
    attacks: [],
    sprite: ""
  },
  enemy_pokemon: {
    hp: 0,
    name: "",
    attacks: [],
    sprite: "",
  },
  /* Creado para la presentación */
  ejemplo:{
    array_of_pokemons: []
  }
}
export default state
```

Store: Mutaciones

```
const mutations = {
    SET_PLAYER_POKEMON(state, payload) {
        state.player_pokemon = payload
    },

    SET_ENEMY_POKEMON(state, payload) {
        state.enemy_pokemon = payload
    }
    /* Creado para la presentación */
    ADD_POKEMON(state, payload) {
      /* Válido en Vuex */
      state.ejemplo.array_of_pokemons.push(payload.pokemon)

      /* En redux, pensando que esto seria un reducer */
      return [...state.ejemplo.array_of_pokemons, payload.pokemon]
    }
}
export default mutations
```

Store: Accciones

```
const actions = {
  async loadPokemon({ commit }, { pokemon_name, target }) {
    /*... más código arriba ...*/

    /* Se procede a mandar a guardar el nuevo pokemon */
    const pokemon_to_set = {
      name: _.capitalize(pokemon_name),
      hp: 100, // TODO: Ver que vida se le va a poner
      attacks: pokemon_attacks,
      sprite: target === "player" ? response.data.sprites['back_default'] : response.data.sprites['front_default']
    }

    const ACTION_NAME = target === "player" ? 'SET_PLAYER_POKEMON' : 'SET_ENEMY_POKEMON'
    commit(ACTION_NAME, pokemon_to_set) /* SE MANDA A EJECUTAR LA MUTACIÓN */
  }
}
export default actions
```

View Home (Template)

```
<template>
  <div>
    <!-- OPCIONES DE ARRIBA -->
    <div id="battle_setup" class="card">
      <div class="card-content">
        <div class="subtitle">Selecciona con que pokemon quieres jugar</div>

        <div class="columns">
          <div class="column">
            <div class="select">
              <select v-model="selected_pokemon">
                <option value selected disabled>Selecciona</option>
                <option value="charmander">Charmander</option>
                <option value="bulbasaur">Bulbasaur</option>
                <option value="squirtle">Squirtle</option>
              </select>
            </div>
          </div>
          <div class="column">
            <button
              class="button is-success"
              @click="handleClick"
              v-bind:disabled="!has_selected_pokemon"
            >Iniciar combate!</button>
          </div>
        </div>
      </div>
    </div>
```

```
<!-- CUADRO DE LA PARTIDA -->
    <div id="battle_container">
      <div class="columns is-mobile" v-if="has_pressed_start">
        <div class="column is-6-desktop is-offset-3-desktop">
          <pokemon orientation="front"/>
        </div>
      </div>

      <div class="columns is-mobile" v-if="has_pressed_start">
        <div class="column is-6-desktop is-offset-3-desktop">
          <pokemon orientation="back"/>
        </div>
      </div>
    </div>
    <div class="modal is-clipped" v-bind:class="[{'is-active':wins}]">
      <div class="modal-background"></div>
      <div class="modal-card">
        <header class="modal-card-head">
          <p class="modal-card-title" v-if="playerWins">Has ganado!</p>
          <p class="modal-card-title" v-else-if="enemyWins">Has perdido :c</p>
        </header>
        <footer class="modal-card-foot">
          <button class="button is-success"
          @click="restart"
          >Jugar Nuevamente!</button>
        </footer>
      </div>
    </div>
  </div>
</template>
```

Home (data & components)

```
components: {
    pokemon,
  },
  data: function () {
    return {
      selected_pokemon: '',
      enemy_pokemon: _.sample(['charmander', 'bulbasaur', 'squirtle']),
      has_pressed_start: false
    }
  },
```

Home (methods)

```
methods: {
    ...mapActions(['loadPokemon','reset']),
    handleClick() {
      this.loadPokemon({ pokemon_name: this.selected_pokemon, target: 'player' })
      this.loadPokemon({ pokemon_name: this.enemy_pokemon, target: 'enemy' })
      this.has_pressed_start = true
    },
    restart: function () {
      window.location.reload(true)
    }
  },
```

Cosas relevantes:

- Uso de MapActions para obtener el componente a la store (y obtener las acciones).

Home (computed)

```
computed: {
    has_selected_pokemon: function () {
      return this.selected_pokemon === '' ? false : true
    },
    ...mapState({
      pokemon(state) {
        return state.player_pokemon
      },
      ...
    }),

    playerWins: function(){
      return this.enemy.current_hp <= 0 && this.enemy.name!="" ? true : false
    },
    ...
    wins: function(){
      return this.has_pressed_start && (this.enemyWins || this.playerWins) ? true : false
    }
  }
```

Cosas relevantes:

- Uso de mapState para conectar el componente al estado de la app.

Component Pokemon (Template)

```
<template>
  <div>
    <!-- PARTE DE ARRIBA -->
    <div class="columns is-mobile" v-if="isEnemy">
      <div class="column">
        <div class="hp_container">
          <div class="columns is-marginless">
            <div class="column" style="padding-bottom: 0px;">{{pokemon.name}}</div>
          </div>
          <div class="columns is-marginless is-mobile">
            <div class="column is-narrow">HP:</div>
            <div class="column is-flex" style="align-items: center;">
              <progress class="progress is-success" v-bind:value="pokemon.current_hp" v-bind:max="pokemon.max_hp"></progress>
            </div>
          </div>
        </div>
      </div>
      <div class="column is-two-quarters-mobile"></div>
    </div>

    <div class="columns is-mobile" v-else>
    ...
</template>
```

Pokemon (Template)

```
    <!-- PARTE DEL MEDIO -->
     <div class="columns is-mobile" v-if="isEnemy">
       <div class="column"></div>
       <div class="column">
         <div class="pokemon_container">
           <img v-bind:src="pokemon.sprite" alt>
         </div>
       </div>
     </div>

     <div class="columns is-mobile" v-else>
       <div class="column">
         <div class="pokemon_container">
           <img v-bind:src="pokemon.sprite" alt>
         </div>
       </div>
       <div class="column"></div>
     </div>
     <div v-if="pokemon.name !== '' && isPlayer">
       <movements/>
     </div>
   </div>
</template>
```

Pokemon (Components, Props & Data)

```
components: {
    movements
  },

  props: {
    orientation: String
  },
```

Cosas relevantes:

- Uso de props para instanciar el elemento según las propiedades entregadas.

Pokemon (Computed)

```
computed: {
  ...mapState({
    pokemon(state) {
      // Es el jugador
      if (this.orientation === "back") {
        return state.player_pokemon
      }
      // Es el enemigo
      else {
        return state.enemy_pokemon
      }
    }
  }),
  isEnemy() {
    return this.orientation === "front" ? true : false
  },
  isPlayer() {
    return this.orientation === "back" ? true : false
  }
}
```

Component Movement (Template)

```
<template>
  <div class="columns is-mobile">
    <div class="column is-three-quarters-mobile">
      <div class="movements_container" style="padding-bottom:6px;">
        <div class="columns is-marginless is-mobile">
          <div class="column" v-bind:style="{backgroundColor:getBackground(0)}">
            <button
              class="button is-small is-fullwidth is-focused"
              value=0
              @click="handleClick"
            >{{pokemon.attacks[0].name}}</button>
          </div>
          <div class="column" v-bind:style="{backgroundColor:getBackground(1)}">
            <button
              class="button is-dark is-small is-fullwidth is-focused"
              value=1
              @click="handleClick"
            >{{pokemon.attacks[1].name}}</button>
          </div>
        </div>
        ...
      </div>
    </div>
    <div class="column" style="padding-left:0px;">
      <div class="movements_container">
          <div class="column" style="padding-bottom: 0px;">PP: {{pokemon.attacks[selected_attack].pp}}</div>
          <div class="column is-narrow">Daño: {{pokemon.attacks[selected_attack].power}}</div>
      </div>
    </div>
  </div>
</template>
```

Movements (data & methods)

```
data: function () {
  return {
    selected_attack: "0",

  }
},
methods: {
  getBackground : function (attack){
    return attack == this.selected_attack ? "red" : "white"
  },
  ...mapActions(['makeAttack']),
  handleClick(event) {
    if (event.target.value === this.selected_attack){
      this.makeAttack({origin:"player", target: this.enemy, attack:this.pokemon.attacks[this.selected_attack]})
      let random = Math.round(Math.random()*3)
      this.makeAttack({origin:"enemy", target:this.pokemon, attack:this.enemy.attacks[random]})
    }
    else{
      this.selected_attack = event.target.value
    }
  }
},
```

Movements (Computed)

```
computed: {
    ...mapState({
      pokemon(state) {
        // Se retorna el pokemon del jugador
        return state.player_pokemon
      },
      enemy(state) {
        // Se retorna el pokemon del jugador
        return state.enemy_pokemon
      }
    })
  }
```