

# HowManyUnitsTo....

## Tarea 3: MyUnits - Vue

GRUPO 7 SECCIÓN 1

Camila Gervasoni  
Juan Pablo Palma  
Giacomo Pasquali

---

# Demo



# Principios de Vue Aplicados

- **Unidireccionalidad:** El flujo de datos va hacia los componentes desde una fuente única.
- **Reactividad:** Los componentes reaccionan exclusivamente a eventos de interés.
- **Renderizado Condicional:** Ciertas vistas se muestran para eventos exclusivos.

# Unidireccionalidad



La información mostrada  
debido al cambio de unidades  
proviene exclusivamente de la  
variable y función que realiza  
el cambio

El ratio de conversión tiene su  
fuente propia, no comulga con  
la transformación de unidades

# Reactividad

Unit Categories

Choose a measurement category to begin converting

- Length 8 UNITS
- Weight 6 UNITS**
- Volume 8 UNITS
- Temperature 3 UNITS
- Pressure 6 UNITS
- Area 7 UNITS
- Speed 5 UNITS
- Energy 7 UNITS

Weight

Convert between different weight units

FROM TO

Kilogram (kg) Gram (g)

10 kg 10000 g

1 kg = 1,000 g

Scale Comparison

Feather 1 g	Coin 10 g	Phone 200 g	Book 500 g
Cat 4 kg	Person 70 kg	Piano 300 kg	Car 1.5 t
Elephant 6 t	Truck 40 t		

Las tarjetas reaccionan a aquellos eventos de interés. Solo se marcan las que cumplen con, en este caso, el peso indicado en el transformador de unidades

# Renderizado Condicional

The screenshot shows a mobile application for unit conversion. At the top, it says "Area" and "Convert between different area units". Below this, there are two input fields: "FROM" (Acre (acre)) and "TO" (Hectare (ha)). The "FROM" field contains "100 acre" and the "TO" field contains "40.468627 ha". A note below says "1 acre = 0.404686 ha". Below the conversion form is a section titled "Scale Comparison" with a grid of items:

Stamp	Phone	Sheet	Door
4 cm <sup>2</sup>	100 cm <sup>2</sup>	0.06 m <sup>2</sup>	2 m <sup>2</sup>
Room	House	Football Field	City Block
20 m <sup>2</sup>	150 m <sup>2</sup>	5,351 m <sup>2</sup>	1 hectare

At the bottom, there is a card for "Central Park" with the text "341 hectares".

Dependiendo de la unidad transformada podemos mostrar distintas visualizaciones complementarias. Una de comparación y otra de peligro

The screenshot shows a mobile application for temperature conversion. At the top, it says "Temperatura" and "Convierte entre diferentes unidades de temperatura". Below this, there are two input fields: "DE" (Celsius (°C)) and "A" (Fahrenheit (°F)). The "DE" field contains "80 °C" and the "A" field contains "176 °F". A note below says "1 °C = 33.8 °F". Below the conversion form is a section titled "Escala de Peligro" with a red stick figure icon. A note next to it says "Peligro" and "Temperatura letal - peligro inmediato". At the bottom, there is a section titled "Valores Extremos" with cards for "Elefante" and "Ballena".

# Filosofía Flux / Vuex

- **Store:** Unidad o estructura donde se guardan datos, estados, variables.
- **Getters:** Métodos responsables de observar las variables, para reflejar los cambios en la aplicación.
- **Mutation:** Evento que genera el cambio en una unidad mediante el call de alguna acción.
- **Action:** Un trigger/gatillo que tiene como destino un cambio en alguna, unidad, dato, visualización.

# Flujo de transformación de unidades: Store

Almacena el estado, data y variables de la aplicación:

```
● ● ●  
1 const conversionFactors = {  
2   length: {  
3     meter: 1,  
4     kilometer: 0.001,  
5     centimeter: 100,  
6     millimeter: 1000,  
7     inch: 39.3701,  
8     foot: 3.28084,  
9     yard: 1.09361,  
10    mile: 0.000621371  
11  },  
12  weight: {  
13    kilogram: 1,  
14    gram: 1000,  
15    pound: 2.20462,  
16    ounce: 35.274,  
17    ton: 0.001,  
18    stone: 0.157473  
19  }, ...
```

# Flujo de transformación de unidades: Getters

Permiten obtener y computar valores derivados del estado:



```
1 currentCategory: (state) => state.categories[state.selectedCategory],
```

Obtiene



```
1 convert: () => (value, fromUnit, toUnit, category) => {
2   if (fromUnit === toUnit) return value
3   const factors = conversionFactors[category]
4   if (category === 'temperature') {
5     if (fromUnit === 'celsius') {
6       if (toUnit === 'fahrenheit') {
7         return factors.fahrenheit.fromBase(value)
8       } else if (toUnit === 'kelvin') {
9         return factors.kelvin.fromBase(value)
10      }
11    } else if (fromUnit === 'fahrenheit') {
12      const celsius = factors.fahrenheit.toBase(value)
13      if (toUnit === 'celsius') {
14        return celsius
15      } else if (toUnit === 'kelvin') {
16        return factors.kelvin.fromBase(celsius)
17      }
18    } else if (fromUnit === 'kelvin') {
19      const celsius = factors.kelvin.toBase(value)
20      if (toUnit === 'celsius') {
21        return celsius
22      } else if (toUnit === 'fahrenheit') {
23        return factors.fahrenheit.fromBase(celsius)
24      }
25    }
26  } else {
27    const baseValue = value / factors[fromUnit]
28    const result = baseValue * factors[toUnit]
29    return Math.round(result * 1000000) / 1000000
30  }
31  return value
32 }
```

Computa

# Flujo de transformación de unidades: Mutations

Modifican el estado directamente:

```
● ● ●  
1 SET_CATEGORY(state, category) {  
2   state.selectedCategory = category  
3   const units = Object.keys(state.categories[category].units)  
4   state.fromUnit = units[0]  
5   state.toUnit = units[1] || units[0]  
6   state.fromValue = ''  
7   state.toValue = ''  
8 },  
9  
10 SET_FROM_UNIT(state, unit) {  
11   state.fromUnit = unit  
12 }, ...
```

# Flujo de transformación de unidades: Actions

Manejan la lógica asíncrona. Las acciones llaman a mutations para modificar estado:



```
1 convertFromTo({ commit, state }, value) {
2   if (value === '' || value === null || value === undefined) {
3     commit('SET_TO_VALUE', '')
4     return
5   }
6   const numValue = parseFloat(value)
7   if (isNaN(numValue)) {
8     commit('SET_TO_VALUE', '')
9     return
10 } ...
```

Funciones

# Flujo de transformación de unidades: Actions

Manejan la lógica asíncrona. Las acciones llaman a mutations para modificar estado:



```
1  onFromValueChange(event) {  
2    const value = event.value  
3    this.convertFromTo(value === null ? '' : value.toString())  
4  },
```

Llamado en eventos de componentes

# Flujo de vistas complementarias: Getters y Store



```
1 getters: {
2   currentCategory: (state) => state.categories[state.selectedCategory],
3   -----
4   currentTranslations: (state) => translations[state.language],
```



```
1 export default createStore({
2   state: {
3     temperature: {
4       name: 'Temperature',
5       icon: '🌡',
6       units: {
7         celsius: '°C',
8         fahrenheit: '°F',
9         kelvin: 'K'
10      }
11    },
12  },
13}
```

Encargados respectivamente de solicitar la información de la categoría a mostrar y de almacenar variables como ícono, nombre, unidades.

# Flujo de vistas complementarias: Mutations



```
1 mutations: {
2   SET_CATEGORY(state, category) {
3     state.selectedCategory = category
4     const units = Object.keys(state.categories[category].units)
5     state.fromUnit = units[0]
6     state.toUnit = units[1] || units[0]
7     state.fromValue = ''
8     state.toValue = ''
9   },

```

Permite que el componente UnitConverter muestre la información apropiada



# Flujo de vistas complementarias: Actions

```
1  }
2      return this.translate('scaleComparison')
3  },
4
5  getDangerLevel() {
6      const value = this.currentValue
7      const category = this.selectedCategory
8      const unit = this.toUnit
9
10     if (category === 'temperature') {
11         if (unit === 'celsius') {
12
13             if (value < 0 && value > -20) return 3
14             if (value < 37) return 2
15             if (value < 50) return 3
16             if (value < 70) return 4
17             if (value < 100) return 5
18             return 6
19     }
20 }
```

## Escala de Peligro



Extremo

Temperatura extrema - muerte instantánea

```
1  getLargerBeings() {
2      const level = this.getDangerLevel()
3
4      if (level >= 5) {
5          return [
6              {
7                  name: 'elephant',
8                  icon: '🐘',
9                  description: this.translate('elephantDescription'),
10                 isActive: level >= 5
11             },
12             {
13                 name: 'whale',
14                 icon: '🐳',
15                 description: this.translate('whaleDescription'),
16                 isActive: level >= 6
17             }
18         ]
19     }
20 }
```

## Valores Extremos



Elefante

Podría afectar a mamíferos grandes



Ballena

Podría afectar a los animales más grandes

# Dificultades /

- Resultados inesperados en ciertos cálculos: representación **number** en javascript basado en IEEE 754 (64 bits de coma flotante).
- Multiples interacciones entre componentes, especialmente en input y su relación con visualización adicional.



# ¡Gracias por su atención!

