01 Vue Herrera - Basics

Basics

- El primer ejemplo será con JS. Luego será todo con TypeScript
- Nos enfocaremos en el **Composition API**
  - Los componentes van a tener internamente una función setup donde va toda la lógica
  - En el ejemplo creo la variable reactiva que al retornarla puedo renderizarla en el template html

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
  const { createApp, ref } = Vue

  createApp({
    setup() {
      const message = ref('Hello vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

- Esto no renderiza otros componentes cuando las variables reactivas cambian

Hola mundo

- Para este ejemplo vamos a usar el CDN
- Creo un index.html, enlazo el cdn y el script app.js (que creo yo)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola mundo en Vue</title>
</head>
<body>
```

```
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- Para renderizar el contenido en el navegador con VSCode usar la extensión Live Server (si no hay que apretar Ctrl+R para refrescar el navegador y que renderice los cambios. Esto no pasa instalando Vue)
- Creo app.js (enlazado en el html)
- Puedo hacer un console.log(Vue) para ver todo el objeto de Vue
    - Puedo usar la desestructuración para tomar lo que me interesa del objeto Vue
    - **NOTA**: trabajando con el CDN no tenemos el intellisense
    - Uso el template para usar un template literal

```
const {createApp, ref} = Vue


const app = createApp({
    template:`
    <h1>Hola mundo</h1>
    <p>Desde app.js</p>
     `
})
```

- ¿Dónde renderizo la aplicación?
- Creo un div con un id

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola mundo en Vue</title>
</head>
<body>
    <div id="MyApp"></div>

    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- Lo monto en el app.js con mount y el selector de css #

```
const {createApp, ref} = Vue
```

```
const app = createApp({
    template:`
    <h1>Hola mundo</h1>
    <p>Desde app.js</p>
     `
})

app.mount("#MyApp")
```

- De esta manera puedo usar esta tecnología donde quiera, combinada con otras, destinando un espacio en el html para la aplicación creada con Vue

Estado del componente - Variables reactivas

- Para usar la variable reactiva en el html (template literal en este caso) debo retornarla en el return
- Para poder modificar la constante message y transformarla en una variable reactiva usamos ref
- Sigo sin poder cambiar el valor de message, pero puedo usar message.value

```
const {createApp, ref} = Vue


const app = createApp({
    template:`
    <h1>{{message}}</h1>
    <p>Desde app.js</p>
     `,

     setup(){

        const message =  ref("I'm Batman");

        setTimeout(()=>{
            message.value="Soy Goku";
        }, 1000)

        return{
            message
        }
     }
})

app.mount("#MyApp")
```

- Entonces, **ref me crea mi variable reactiva y con .value puedo cambiar su valor**
- **Debo retornarla en la función setup para poder renderizarla**

- **Uso doble llave para renderizarla en el html**
- Esta es una de las dos formas de hacer un cambio en una variable reactiva

```
const {createApp, ref} = Vue


const app = createApp({
    template:`
    <h1>{{message}}</h1>
    <p>{{author}}</p>
     `,

     setup(){

        const message =  ref("I'm Batman");
        const author = ref("-Bruce Wayne")

        setTimeout(()=>{
            message.value="Hola, soy Goku";
            author.value="-Goku"
        }, 1000)

        return{
            message,
            author
        }
     }
})

app.mount("#MyApp")
```

Separar HTML y eventos

- Supongamos que no queremos el html en el template sino que esté en el index.html
- En el div con id MyApp es donde Vue existe (es el id que usado con .mount para montar la app)
- Se puede “teletransportar” elementos de Vue fuera de este div, lo veremos más adelante

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola mundo en Vue</title>
</head>
<body>
    <div id="MyApp">
        <h3>{{message}}</h3>
        <p>{{author}}</p>
```

```
    </div>

    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- Pongamos que en lugar de usar el setTimeOut quiero usar un botón para cambiar el contenido
- Para disparar eventos uso **v-on**
- **NOTA**: cuando usemos las herramientas de Vue y no el CDN tendremos el intellisense para ver todos los eventos disponibles
- Creo la función (dentro del setup) para usar en el v-on, en app.js
- La exporto en el return

```
const {createApp, ref} = Vue


const app = createApp({
     setup(){

        const message =  ref("I'm Batman");
        const author = ref("-Bruce Wayne")

        const changeQuote=()=>{
            message.value="Hola, soy Goku",
            author.value="-Goku"
        }

        return{
            message,
            author,
            changeQuote
        }
     }
})

app.mount("#MyApp")
```

- Ahora solo tengo que usarla en el index.html
- La puedo mandar solo por referencia porque no tengo que pasarle ningún valor

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola mundo en Vue</title>
```

```
</head>
<body>
    <div id="MyApp">
        <h3>{{message}}</h3>
        <p>{{author}}</p>

        <button v-on:click="changeQuote">Cambiar Mensaje</button>

    </div>

    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- Las variables reactivas va a cambiar en todos los lugares donde se haga referencia a ellas
- Tenemos variables reactivas que son propiedades computadas. Esa propiedad computada va a ser basada en otras variables reactivas
- **Todo lo que hemos hecho se puede simplificar muchísimo una vez trabajemos en una aplicación de Vue completa**

v-for - Iterar elementos

- Nuevo ejercicio. Creo un nuevo index.html
- Volvemos a usar el CDN, un nuevo app.js,el div con #MyApp
- index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <div id="MyApp">
        <h1>Batman Quotes</h1>
        <hr>
    </div>

    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- En el app.js desestructuro de Vue createApp y ref
- Uso createApp con la función setup y el return

- Monto la aplicación con .mount usando el id del div del html usando el selector # de CSS

```
const {createApp, ref} = Vue


const app = createApp({
    setup(){


        return{

        }
    }
})

app.mount("#MyApp")
```

- v-for no solo va a permitir iterarlos, también desestructurarlos, usar índices, etc
- Usaremos estas quotes de Batman, las pego en app.js
- Aunque la constante esté declarada fuera del setup, tengo acceso porque JS tiene acceso a ella
- JS se antepone a cualquier cosa que haga Vue
- Entonces, puedo retornarla en el return de setup (no es un objeto reactivo, simplemente es un arreglo)

```
const {createApp, ref} = Vue

const quotes = [
    { quote:
'The night is darkest just before the dawn. And I promise you, the dawn is coming.',
author: 'Harvey Dent, The Dark Knight' },
    { quote: 'I believe what doesn’t kill you simply makes you, stranger.', author:
'The Joker, The Dark Knight' },
    { quote:
'Your anger gives you great power. But if you let it, it will destroy you… As it almost
did me', author: 'Henri Ducard, Batman Begins' },
    { quote: 'You either die a hero or live long enough to see yourself become the
villain.', author: 'Harvey Dent, The Dark Knight' },
    { quote: 'If you’re good at something, never do it for free.', author: 'The Joker,
The Dark Knight' },
    { quote: 'Yes, father. I shall become a bat.', author:
'Bruce Wayne/Batman, Batman: Year One' },
]

const app = createApp({
    setup(){


        return{
            quotes
```

```
        }
    }
})

app.mount("#MyApp")
```

- El v-for es como trabajar con un for of (también hay un for-in)

```
<div id="MyApp">
    <h1>Batman Quotes</h1>
    <hr>
    <ul>
        <li v-for="quote in quotes">
            {{quote.author}}
        </li>
    </ul>
</div>
```

- Puedo usar desestructuración

```
<div id="MyApp">
    <h1>Batman Quotes</h1>
    <hr>
    <ul>
        <li v-for="({quote, author}) in quotes">
            <span>{{quote}}</span>
            <blockquote>{{author}}</blockquote>
        </li>
    </ul>
</div>
```

- Para obtener el índices (bien podría usar ol, ordered list) pero si quiero controlarlos yo, uso index
- Uso index+1 porque los arreglos en JS empiezan en 0

```
<div id="MyApp">
    <h1>Batman Quotes</h1>
    <hr>
    <ul>
        <li v-for="({quote, author}, index) in quotes">
            <span>{{index+1}}- {{quote}}</span>
            <blockquote>{{author}}</blockquote>
        </li>
    </ul>
</div>
```

v-if VS v-show

- Estas directivas nos van a permitir ocultar elementos
- Pongamos que quiero trabajar con una variable para mostrar el autor
- Como va a cambiar (va a ser reactiva) uso ref

```
const {createApp, ref} = Vue

const quotes = [/*quotes de Batman*/]

const app = createApp({
    setup(){

        const showAuthor = ref(true);

         const showAuthorFunc=()=>{
            showAuthor.value= !showAuthor.value
        }

        return{
            quotes,
            showAuthor,
            showAuthorFunc
        }
    }
})

app.mount("#MyApp")
```

- Uso el v-if (si lo pongo en false no muestra el autor)
- En lugar del v-on puedo usar @

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <div id="MyApp">
        <h1>Batman Quotes</h1>
        <button @click="showAuthorFunc">Toggle Author</button>
        <hr>

        <ul>
            <li v-for="({quote, author}, index) in quotes">
                <span>{{index+1}}- {{quote}}</span>
                <blockquote v-if="showAuthor">{{author}}</blockquote>
```

```
            </li>
        </ul>
    </div>

    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="./app.js"></script>
</body>
</html>
```

- En el lado del template no tengo que usar .value, porque Vue ya desenvuelve el valor para ser consumido
- Entonces también podría usarse directamente en el @click

```
<div id="MyApp">
    <h1>Batman Quotes</h1>
    <button @click="showAuthor=!showAuthor">Toggle Author</button>
    <hr>
    <ul>
        <li v-for="({quote, author}, index) in quotes">
            <span>{{index+1}}- {{quote}}</span>
            <blockquote v-if="showAuthor">{{author}}</blockquote>
        </li>
    </ul>
</div>
```

- En vez del v-if también tenemos el v-show
- El **v-show** lo que hace es **ponerle el display:none al elemento** (o quitárselo)
- Con el **v-if el elemento deja de existir en el html**
- El v-if dispara el ciclo de creación del componente una vez se vuelve a crear
- El v-show solo le agrega la clase de CSS display:none, por lo que no tiene que volver a cargar todo el componente cuando se le da al toggle

Eventos y propiedades computadas

- Supongamos que tenemos un botón que añade una nueva frase
- Creo una función para añadir una nueva frase
- Uso unshift para añadir la frase al inicio del arreglo de quotes

```
const app = createApp({
    setup(){

        const showAuthor = ref(true);

        const showAuthorFunc=()=>{
            showAuthor.value= !showAuthor.value
        }
```

```
        const addQuote=()=>{
            quotes.unshift({quote: 'Hola mundo', author: 'Manolo García'})
            console.log(quotes)
        }

        return{
            quotes,
            showAuthor,
            showAuthorFunc,
            addQuote
        }
    }
})
```

- De esta manera, si toco el botón de AddQuote veo en el console.log que se agrega la frase pero no la renderiza
- Vue no sabe que eso es algo a lo que tiene que reaccionar
- Hay dos maneras de decirle a Vue de que esto tiene que ser una variable reactiva
- Hasta ahora solo hemos visto con ref
- quote.unshift no nos va a funcionar porque es algo que tenemos directamente en un arreglo, pero ya no es un arreglo, es una propiedad reactiva, por lo que tengo que apuntar a .value

```
const {createApp, ref} = Vue

const originalQuotes = [
    { quote:
'The night is darkest just before the dawn. And I promise you, the dawn is coming.',
author: 'Harvey Dent, The Dark Knight' },
    { quote: 'I believe what doesn’t kill you simply makes you, stranger.', author:
'The Joker, The Dark Knight' },
    { quote:
'Your anger gives you great power. But if you let it, it will destroy you… As it almost
did me', author: 'Henri Ducard, Batman Begins' },
    { quote: 'You either die a hero or live long enough to see yourself become the
villain.', author: 'Harvey Dent, The Dark Knight' },
    { quote: 'If you’re good at something, never do it for free.', author: 'The Joker,
The Dark Knight' },
    { quote: 'Yes, father. I shall become a bat.', author:
'Bruce Wayne/Batman, Batman: Year One' },
]

const app = createApp({
    setup(){

        const showAuthor = ref(true);
        const quotes= ref(originalQuotes);

        const showAuthorFunc=()=>{
            showAuthor.value= !showAuthor.value
        }
```

```
        const addQuote=()=>{
            quotes.value.unshift({quote: 'Hola mundo', author: 'Manolo García'})
        }



        return{
            quotes,
            showAuthor,
            showAuthorFunc,
            addQuote
        }
    }
})

app.mount("#MyApp")
```

- Podemos crear una propiedad computada que se llame totalQuotes
- Desestructuro de Vue computed

```
const app = createApp({
    setup(){

        const showAuthor = ref(true);
        const quotes= ref(originalQuotes);
        const totalQuotes=computed(()=>{
            return quotes.value.length
        })

        const showAuthorFunc=()=>{
            showAuthor.value= !showAuthor.value
        }

        const addQuote=()=>{
            quotes.value.unshift({quote: 'Hola mundo', author: 'Manolo García'})
        }



        return{
            quotes,
            showAuthor,
            showAuthorFunc,
            addQuote,
            totalQuotes
        }
    }
})
```

- Lo renderizo en el html

```
<div id="MyApp">
    <h1>Batman Quotes - {{totalQuotes}}</h1>
    <button @click="showAuthor=!showAuthor">Toggle Author</button>
    <button @click="addQuote">Add Quote</button>
    <hr>

    <ul>
        <li v-for="({quote, author}, index) in quotes">
            <span>{{index+1}}- {{quote}}</span>
            <blockquote v-if="showAuthor">{{author}}</blockquote>
        </li>
    </ul>
</div>
```

- Vue es lo suficientemente inteligente para determinar que dentro de computed, si estoy usando cualquier valor reactivo, se va a dar cuenta y no hay que defenir dependencias (como en el arreglo del useEffect) ni nada por el estilo
- Pongamos que colocamos un input y quiero que al apretar el enter introduzca la nueva frase
- Usaríamos el v-on, la forma corta es @

```
 <input type="text"
    placeholder="Add Quote"
    @keypress="addQuote"
    >
```

- Si lo pongo de esta manera, cada vez que presiono una tecla añade la frase
- Hago uso de un modificador para disparar la función

```
<div id="MyApp">
    <h1>Batman Quotes - {{totalQuotes}}</h1>
    <input type="text"
    placeholder="Add Quote"
    @keypress.enter="addQuote"
    >
    <button @click="showAuthor=!showAuthor">Toggle Author</button>
    <button @click="addQuote">Add Quote</button>
    <hr>

    <ul>
        <li v-for="({quote, author}, index) in quotes">
            <span>{{index+1}}- {{quote}}</span>
            <blockquote v-if="showAuthor">{{author}}</blockquote>
        </li>
```

```
    </ul>
</div>
```

- **NOTA**:no añade la frase que escribo en el input. Al apretar enter en el input dispara la función addQuote, gracias al modificador .enter lo que añade la frase en duro que pusimos a la función que es Hola Mundo

v-model

- Sirve para crear un 2 way data binding (un enlazado de dos lugares). Es decir, si cambia en el input html, que también cambie en el archivo de javascript y a la inversa
- Está limitado (según la documentación) a las etiquetas input, select, textarea y componentes personalizados
- Tiene ciertos modificadores: .lazy, .number y .trim
- Con el v-model en nuestro caso, tengo que apuntar a alguna variable
- La creo en app.js, newMessage. La puedo inicializar vacía
- Cambio de la función addQuote la quote en duro por newMessage.value
- Después la reseteo a un valor vacío para vaciar el input

```
const app = createApp({
    setup(){

        const showAuthor = ref(true);
        const quotes= ref(originalQuotes);
        const totalQuotes=computed(()=>{
            return quotes.value.length;
        })
        const newMessage=ref('');

        const showAuthorFunc=()=>{
            showAuthor.value= !showAuthor.value;
        }

        const addQuote=()=>{
            quotes.value.unshift({quote: newMessage.value, author: 'Manolo García'});
            newMessage.value='';
        }



        return{
            quotes,
            showAuthor,
            showAuthorFunc,
            addQuote,
            totalQuotes,
            newMessage
        }
```

```
    }
})
```

• En el html tengo la referencia a la variable

```
<div id="MyApp">
    <h1>Batman Quotes - {{totalQuotes}}</h1>
    <input type="text"
    placeholder="Add Quote"
    @keypress.enter="addQuote"
    v-model="newMessage"
    >
    <button @click="showAuthor=!showAuthor">Toggle Author</button>
    <button @click="addQuote">Add Quote</button>
    <hr>

    <ul>
        <li v-for="({quote, author}, index) in quotes">
            <span>{{index+1}}- {{quote}}</span>
            <blockquote v-if="showAuthor">{{author}}</blockquote>
        </li>
    </ul>
</div>
```

02 Vue Herrera - Vite Single File Components

- La opción recomendada en la documentación es npm create vue@latest

> npm create vue@latest

- name: Indecision-app
- TypeScript: yes
- JSX:no
- Router: no
- Pinia: no
- Vitest: no
- End-to-End: no
- ESLint: yes
- Prettier: yes
- El router, pinia, vitest aprenderemos como configurarlo luego
- Las devtools ya vienen por defecto!
- Hay que hacer un npm install en la carpeta del proyecto
- vamos con el análisis de los archivos y directorios que crea por defecto

Explicación de archivos y directorios

- vite.config.ts: archivo de configuración de vite. Tiene los plugins de vue y resuelve ./src

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
```

```
  },
})
```

- tsconfig son archivos de configuración de node, TypeScript. No hace falta tocarlos
- El README nos lo tendremos que currar para describir los pasos para que otros sepan como echar andar la aplicación
- El package.json ya lo conocemos, es donde están los scripts, las dependencias, el nombre de la app, la versión y otras cosas
- Hay varios scripts

```
"scripts": {
    "dev": "vite",
    "build": "run-p type-check \"build-only {@}\" --",
    "preview": "vite preview",
    "build-only": "vite build",
    "type-check": "vue-tsc --build",
    "lint": "eslint . --fix --cache",
    "format": "prettier --write --experimental-cli src/"
  }
```

- Tenemos el index.html donde esta el div con id app que es donde vivirá la aplicación de Vue
    - Aquí es donde agregaríamos bootstrap, por ejemplo

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="UTF-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vite App</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.ts"></script>
  </body>
</html>
```

- El env.d.ts nos facilitará trabajar con variables de entorno (aunque no en este archivo)
- .prettierc.json son las reglas de configuración del prettier

```
{
  "$schema": "https://json.schemastore.org/prettierrc",
  "semi": false, //pone todos los punto y coma al final en true
  "singleQuote": true,
  "printWidth": 100,
```

```
  "trailingComma": "all" //todo con comillas dobles
}
```

- .gitignore no sube los archivos a git que en el se incluyen
- eslint.config.ts

```
import { globalIgnores } from 'eslint/config'
import { defineConfigWithVueTs, vueTsConfigs } from '@vue/eslint-config-typescript'
import pluginVue from 'eslint-plugin-vue'
import skipFormatting from '@vue/eslint-config-prettier/skip-formatting'

// To allow more languages other than `ts` in `.vue` files, uncomment the following
lines:
// import { configureVueProject } from '@vue/eslint-config-typescript'
// configureVueProject({ scriptLangs: ['ts', 'tsx'] })
// More info at https://github.com/vuejs/eslint-config-typescript/#advanced-setup

export default defineConfigWithVueTs(
  {
    name: 'app/files-to-lint',
    files: ['**/*.{vue,ts,mts,tsx}'],
  },

  globalIgnores(['**/dist/**', '**/dist-ssr/**', '**/coverage/**']),

  ...pluginVue.configs['flat/essential'],
  vueTsConfigs.recommended,

  skipFormatting,
)
```

- En la carpeta src tenemos el main.ts y el App.vue
- main.ts

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

- En Vue.app no esta App porque decidí tener el proyecto en blanco. Lo haremos desde cero
- Muestra esto por defecto

```
<script setup lang="ts"></script>

<template>
  <h1>You did it!</h1>
  <p>
    Visit <a href="https://vuejs.org/" target="_blank" rel="noopener">vuejs.org</a> to
```

```
read the
    documentation
  </p>
</template>

<style scoped></style>
```

- En public es donde guardaremos recursos estáticos
- Los node_modules es donde están todos los paquetes de dependencias. No se tocan de manera manual
- En .vscode están las configuraciones de vscode para trabajar con Vue
    - settings.json es la configuración de vscode determinada por el equipo de Vue

SFC - Single File Component

- En src/App.vue (es el único componente que nos va a permitir tener una sola palabra)
- El template siempre requiere un child element
- Agregando la palabra setup a la etiqueta script es como si ya tuviera la función setup definida
- Con la etiqueta style puedo añadirle estilos
- Para que no los aplique de forma global y solo los aplique al componente uso la palabra **scoped**
- El scoped añade una data adicional al elemento en el html generado de ese componente que ayuda a Vue a que solo le aplique los estilos al elemento indicado

```
<script setup lang="ts" setup> //agrego aquí la palabra setup
  console.log("Hola mundo!")
</script>

<template>
  <h1>Hola mundo!</h1>
</template>


<style scoped>
h1{
  color: red
}
</style>
```

- Si el main.ts da error de que no encuentra App.vue hacer un **Developer: Reload Window** con Ctrl+Shift+P
- En la parte central inferior del navegador están las devtools, hay una V de Vue y una mira de disparo

- Si le doy a la mira de disparo y luego clico sobre un componente me lleva al código del componente en vscode

Estado y eventos

- Vamos a crear un contador
- El square será un valor computado según el valor que tengamos en el counter
- El esqueleto sería este

```
<script setup lang="ts"></script>

<template>
  <section>
    <h3>Counter: {{ 10 }}</h3>
    <h3>Square: {{ 10 * 10}}</h3>
    <div>
      <button>+1</button>
      <button>-1</button>
    </div>
  </section>
</template>
```

- Creo las variables reactivas
- Square es una propiedad computada que cambia cuando counter cambia

```
<script lang="ts" setup>
import { computed, ref } from 'vue';

const counter = ref(2);
const squareCounter = computed(()=>counter.value * counter.value)
</script>

<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button>+1</button>
      <button>-1</button>
    </div>
  </section>
</template>
```

- Vamos con los botones

```
<script lang="ts" setup>
import { computed, ref } from 'vue';
```

```
const counter = ref(2);
const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;
</script>

<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button @click="increment">+1</button>
      <button @click="decrement">-1</button>
    </div>
  </section>
</template>
```

- También podría haber usado counter++ directamente
- No hace falta usar .value en el template cuando usamos variables reactivas usando ref
- Veremos otras maneras de trabajar

```
<button @click="counter++">+1</button>
```

Nuestro primer componente

- Creemos un componente específico del counter
- Creo src/components/MyCounter.vue (el nombre tienen que ser dos palabras, excepto App.vue)
- App.vue queda así

```
<script lang="ts" setup>
import MyCounter from './components/MyCounter.vue';


</script>

<template>
  <h1>Mi primera app</h1>
  <hr>

  <MyCounter />
</template>
```

- Mycounter.vue queda así

```
<script lang="ts" setup>
import { computed, ref } from 'vue';

const counter = ref(2);
const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;
</script>

<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button @click="increment">+1</button>
      <button @click="decrement">-1</button>
    </div>
  </section>
</template>
```

- Hay otra sintaxis que es usando data(), que es la de Options API (sin el setup)
- custom1 es un elemento personalizado
- Ejemplo:

```
<template>
  <div class="example">{{ msg }}</div>
</template>

<script>
export default {
  data() { //Esto es el Options API
    return {
      msg: 'Hello world!'
    }
  }
}
</script>

<style>
.example {
  color: red;
}
</style>

<custom1>
  This could be e.g. documentation for the component.
</custom1>
```

Define Props - Recibir properties

- Es probable que necesite comunicar componentes
- En este ejemplo, puede que quiera especificar un valor por defecto en MyCounter
- Para ello se usa **v-bind**, la forma corta es :
- App.vue

```
<script lang="ts" setup>
import MyCounter from './components/MyCounter.vue';


</script>

<template>
  <h1>Mi primera app</h1>
  <hr>

  <MyCounter :value="5"/>
</template>
```

- Estamos viendo la manera en que se usa script setup
- Para las props tenemos las funciones defineProps (y también está defineEmits para los eventos)
- MyCounter.vue

```
<script lang="ts" setup>
import { computed, ref } from 'vue';

const props= defineProps({
    value: {type: Number, required: true}
    //value: Number   //de esta manera considera value opcional
})

const counter = ref(props.value);
const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;

</script>
```

- Podríamos desestructurar el value de las props

```
const {value}= defineProps({
    value: {type: Number, required: true}
```

```
    //value: Number   //de esta manera considera value opcional
})
```

- Se puede usar <> y unas llaves con defineProps para tipar las props. De esta manera se consideran obligatorias también

```
const props= defineProps<{
  value: number
  }>()
```

- Perfectamente puedo crear una interfaz

```
interface Props{
  value: number //si lo quiero opcional uso value?
}

const props = defineProps<Props>()
```

- Si lo pongo opcional (value?) TypeScript se queja porque puede la prop value puede ser undefined
- Si es nulo le pongo 5 por defecto

```
const counter = ref(props.value ?? 5);
```

Componente tradicional

- Hasta ahora hemos visto el script setup (Composition API)
- Es la forma abreviada de trabajar
- Cuando los componentes tienen mucha lógica vamos a usar los **Composable Functions** (es similar a los custom hooks)
- Para la forma tradicional, creo un nuevo componente MyCounterScript.vue
- Le quito la palabra setup a la etiqueta script
- Uso defineComponent

```
<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button @click="increment">+1</button>
      <button @click="decrement">-1</button>
    </div>
  </section>
```

```
</template>

<script lang="ts">
import { defineComponent, ref, computed} from 'vue';

export default defineComponent({
    props:{
        value: {type:Number, required: true}
    },
    setup(props){

        const counter = ref(props.value);
        const squareCounter = computed(()=>counter.value * counter.value);
        const increment =()=>counter.value ++;
        const decrement =()=>counter.value --;

        return {
            counter,
            squareCounter,
            increment,
            decrement
        }

    }
})
</script>
```

- Este código no es obsoleto, nos va a servir cuando tengamos mucho código en un archivo independiente

Separar lógica del SFC

- Con mucho código usaríamos Composable Functions y trabajaríamos con el script setup
- Lo veremos
- Creo MyCounterScript2.vue y MyCounterScript2.ts en components/my-counter-script
- Si tuviera css de este componente también lo colocaría dentro de esta carpeta
- Lo que había dentro del script en el componente de vue lo corto y lo pego en el .ts

```
import { defineComponent, ref, computed} from 'vue';

export default defineComponent({ //es importante usar defineComponent!!
    props:{
        value: {type:Number, required: true}
    },
    setup(props){

        const counter = ref(props.value);
        const squareCounter = computed(()=>counter.value * counter.value);
        const increment =()=>counter.value ++;
        const decrement =()=>counter.value --;
```

```
        return {
            counter,
            squareCounter,
            increment,
            decrement
        }

    }
})
```

- Es el código de TypeScript que corresponde a este componente de Vue
- Para referenciarlo uso src en la etiqueta script del componente .vue

```
<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button @click="increment">+1</button>
      <button @click="decrement">-1</button>
    </div>
  </section>
</template>

<script lang="ts" src="./MyCounterScript2.ts">

</script>
```

Integrar estilos de terceros

- Aprendamos a configurar Bootstrap pero usaremos Tailwind a lo largo del curso
- Hay varias maneras de trabajar con estilos
- Puedo crear un styles.css en src y trabajar con una hoja de estilos global
- Lo importo en el main

- styles.css

```
html,body{
    margin:0;
    background-color:#018992
}
```

- main.ts

```
```

```
import { createApp } from 'vue'
import App from './App.vue'
import './styles.css'

createApp(App).mount('#app')
```

- Para usar Bootstrap lo más fácil sería usar el CDN en el html
- index.html

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="UTF-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vite App</title>
    <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-
sRIl4kxILFvY47J16cr9ZwB07vP4J8+LH7qKQnuqkuIAvNWLzeN8tE5YBujZqJLB"
    crossorigin="anonymous" />
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.ts"></script>
  </body>
</html>
```

- Pero vamos a trabajar con Tailwind en Vite

> npm install tailwindcss @tailwindcss/vite

- En vite.config.ts

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
    tailwindcss()
  ],
  resolve: {
    alias: {
```

```
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
  },
})
```

- En el styles.css

```
@import "tailwindcss";
```

- Ya puedes usar tailwindcss
- Para juntar todas las clases de Tailwind en una sola clase uso @apply
- Debo usar @reference “tailwindcss” en el scope de style

```
<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button class="btn" @click="increment">+1</button>
      <button class="btn" @click="decrement">-1</button>
    </div>
  </section>
</template>

<script lang="ts" src="./MyCounterScript2.ts">

</script>

<style scoped>
  @reference "tailwindcss";

  .btn{
    @apply p-5 bg-red-500 rounded hover:bg-red-600 mr-2;
  }

</style>
```

- Para que desaparezca el warning de @reference abro las settings de VsCode con Ctrl+Shift+P Preferences: Open Settings (UI)
- Buscar unknown, en CSS > Lint: Unknown At Rules marcar ignore

Crear y desplegar aplicación

- Ejecutar **npm run build**
- Esto nos da un .html, un .css y un .js y los assets en una carpeta dist
- Estos son los archivos que se suben al hosting (Netlify u otros)

- Para comprobar la versión de producción instalar http-server y usar el comando http en la carpeta dist

Composable Functions

- Sería conveniente agrupar toda la lógica del contador para tenerla en un único lugar para que otros componentes puedan usarla
- Es simplemente una función con la lógica de la aplicación que retorna las variables (reactivas) para que cualquiera pueda usarlas
- Está inspirado en los custom hooks
- Creo en src/composables/useCounter.ts
- Para solucionar las importacionas usar **Ctrl + .**

```
import { computed, ref } from 'vue';

export const useCounter = ()=>{



const counter = ref(5);
const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;


    //puede regresar cualquier cosa, un objeto, un arreglo
    // si retorna un arreglo usar return [] as const para que siempre regrese el mismo
orden
    return  {
        counter,
        squareCounter,
        increment,
        decrement
    }
}
```

- Cuando la propiedad es read-only (como el squareCounter) se puede definir directamente en el return
- Para usar los valores desestructuro de la función

```
const {counter, squareCounter,increment,decrement} = useCounter()
```

- Para poder incluir un valor inicial al contador podría hacerse así

```
import { computed, ref } from 'vue';

export const useCounter = (initialValue: number)=>{

const counter = ref(initialValue);
const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;


    //puede regresar cualquier cosa, un objeto, un arreglo
    // si retorna un arreglo usar return [] as const para que siempre regrese el mismo
orden
    return  {
        counter,
        squareCounter,
        increment,
        decrement
    }
}
```

• Para usar las props en MyCounterScript.vue

```
<template>
  <section>
    <h3>Counter: {{ counter }}</h3>
    <h3>Square: {{ squareCounter}}</h3>
    <div>
      <button @click="increment">+1</button>
      <button @click="decrement">-1</button>
    </div>
  </section>
</template>

<script lang="ts">
import { useCounter } from '@/composables/useCounter';
import { defineComponent, ref, computed} from 'vue';

export default defineComponent({
    props:{
        value: {type:Number, required: true}
    },
    setup(props){
        const {counter, squareCounter,increment,decrement} = useCounter(props.value)


        return {
            counter,
            squareCounter,
            increment,
            decrement
        }

    }
```

```
})
</script>
```

- Si yo declaro la variable fuera del useCounter de esta manera

```
import { computed, ref } from 'vue';

const counter = ref(10);

export const useCounter = (initialValue: number)=>{


const squareCounter = computed(()=>counter.value * counter.value);
const increment =()=>counter.value ++;
const decrement =()=>counter.value --;


    //puede regresar cualquier cosa, un objeto, un arreglo
    // si retorna un arreglo usar return [] as const para que siempre regrese el mismo
orden
    return  {
        counter,
        squareCounter,
        increment,
        decrement
    }
}
```

- Todos los MyCounter que usen el useCounter tendrán el valor de 10 y cambiarán de manera simultánea
- Entonces he creado un **gestor de estado global** basado en un Composable

---

03 Vue Herrera - Indecision App

Objetivo de la sección

- Crearemos un chat
- Las respuestas las obtendremos de un API de Yes No que devuelve una imagen con un Si o un NO
- No hay IA aquí
- Vamos a tener scroll en la caja del chat
- Vamos a aprender sobre elenvío y comunicación entre componentes, manejo de estado, composables, etc
- Borramos toda la lógica de los contadores

Estructura y diseño del chat

- Gist - Estructura a utilizar (chat simplificado con la etiqueta template de Vue)

```
<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
      <span>Mi esposa</span>
    </div>

    <div class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
        <!-- Messages go here -->
        <!-- Example Message -->
        <div class="flex justify-end">
          <div class="bg-blue-200 text-black p-2 rounded-lg max-w-xs">
            Hey, how's your day going?
          </div>
        </div>

        <!-- Example Received Message -->
        <div class="flex">
          <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
            Not too bad, just a bit busy. How about you?
          </div>
        </div>
      </div>
    </div>

    <div class="bg-white p-4 flex items-center">
      <input
```

```
        type="text"
        placeholder="Type your message..."
        class="flex-1 border rounded-full px-4 py-2 focus:outline-none"
      />
      <button
        class="bg-blue-500 text-white rounded-full p-2 ml-2 hover:bg-blue-600
focus:outline-none"
      >
        <svg
          width="20px"
          height="20px"
          viewBox="0 0 24 24"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          stroke="#ffffff"
        >
          <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
          <g id="SVGRepo_tracerCarrier" stroke-linecap="round" stroke-
linejoin="round"></g>
          <g id="SVGRepo_iconCarrier">
            <path
              d="M11.5003 12H5.41872M5.24634 12.7972L4.24158 15.7986C3.69128 17.4424
3.41613 18.2643 3.61359 18.7704C3.78506 19.21 4.15335 19.5432 4.6078 19.6701C5.13111
19.8161 5.92151 19.4604 7.50231 18.7491L17.6367 14.1886C19.1797 13.4942 19.9512 13.1471
20.1896 12.6648C20.3968 12.2458 20.3968 11.7541 20.1896 11.3351C19.9512 10.8529 19.1797
10.5057 17.6367 9.81135L7.48483 5.24303C5.90879 4.53382 5.12078 4.17921 4.59799
4.32468C4.14397 4.45101 3.77572 4.78336 3.60365 5.22209C3.40551 5.72728 3.67772 6.54741
4.22215 8.18767L5.24829 11.2793C5.34179 11.561 5.38855 11.7019 5.407 11.8459C5.42338
11.9738 5.42321 12.1032 5.40651 12.231C5.38768 12.375 5.34057 12.5157 5.24634 12.7972Z"
              stroke="#ffffff"
              stroke-width="2"
              stroke-linecap="round"
              stroke-linejoin="round"
            ></path>
          </g>
        </svg>
      </button>
    </div>
  </div>
</template>
```

- Creo un nuevo directorio src/views/IndecisionView.vue
- Pego el código del Gist
- Lo renderizo en App.vue

```
<template>
 <IndecisionView />
</template>

<script lang="ts" setup>
import IndecisionView from './views/IndecisionView.vue';
```

```
</script>
```

- Nos da el layout del chat pero falta la lógica
- En tsconfig.app.json tenemos configurado el path relativo con la arroba

```
{
  "extends": "@vue/tsconfig/tsconfig.dom.json",
  "include": ["env.d.ts", "src/**/*", "src/**/*.vue"],
  "exclude": ["src/**/__tests__/*"],
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",

    "paths": {
      "@/*": ["./src/*"]
    }
  }
}
```

- Eso significa que puedo usar @ para entrar en src en las importaciones

```
import IndecisionView from '@/views/IndecisionView.vue';
```

- Se pueden separar las herramientas en otra ventana si molestan con Alt+Shift+D

> http://localhost:5173/**devtools**/

- Aunque podríamos trabajar sobre el código dado, lo recomendable sería trabajar con **componentes pequeños más controlables**
- Creo components/chat/Chatmessages.Vue
- Coloco la etiqueta template y dentro el código de los chat messages
- ChatMessages.vue

```
<template>
     <div class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
        <!-- Messages go here -->
        <!-- Example Message -->
        <div class="flex justify-end">
          <div class="bg-blue-200 text-black p-2 rounded-lg max-w-xs">
            Hey, how's your day going?
          </div>
        </div>

        <!-- Example Received Message -->
        <div class="flex">
          <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
```

```
            Not too bad, just a bit busy. How about you?
          </div>
        </div>
      </div>
    </div>
</template>
```

- Renderizo ChatMessages en el IndecisionView.vue
- Creo MessageBox.vue

```
<template>
    <div class="bg-white p-4 flex items-center">
      <input
        type="text"
        placeholder="Type your message..."
        class="flex-1 border rounded-full px-4 py-2 focus:outline-none"
      />
      <button
        class="bg-blue-500 text-white rounded-full p-2 ml-2 hover:bg-blue-600
focus:outline-none"
      >
        <svg
          width="20px"
          height="20px"
          viewBox="0 0 24 24"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          stroke="#ffffff"
        >
          <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
          <g id="SVGRepo_tracerCarrier" stroke-linecap="round" stroke-
linejoin="round"></g>
          <g id="SVGRepo_iconCarrier">
            <path
              d="M11.5003 12H5.41872M5.24634 12.7972L4.24158 15.7986C3.69128 17.4424
3.41613 18.2643 3.61359 18.7704C3.78506 19.21 4.15335 19.5432 4.6078 19.6701C5.13111
19.8161 5.92151 19.4604 7.50231 18.7491L17.6367 14.1886C19.1797 13.4942 19.9512 13.1471
20.1896 12.6648C20.3968 12.2458 20.3968 11.7541 20.1896 11.3351C19.9512 10.8529 19.1797
10.5057 17.6367 9.81135L7.48483 5.24303C5.90879 4.53382 5.12078 4.17921 4.59799
4.32468C4.14397 4.45101 3.77572 4.78336 3.60365 5.22209C3.40551 5.72728 3.67772 6.54741
4.22215 8.18767L5.24829 11.2793C5.34179 11.561 5.38855 11.7019 5.407 11.8459C5.42338
11.9738 5.42321 12.1032 5.40651 12.231C5.38768 12.375 5.34057 12.5157 5.24634 12.7972Z"
              stroke="#ffffff"
              stroke-width="2"
              stroke-linecap="round"
              stroke-linejoin="round"
            ></path>
          </g>
        </svg>
      </button>
```

```
    </div>
</template>
```

- Lo renderizo en IndecisionView.vue, quedando así

```
<script setup lang="ts">
import ChatMessages from '@/components/chat/ChatMessages.vue';
import MessageBox from '@/components/chat/MessageBox.vue';

</script>

<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
      <span>Mi esposa</span>
    </div>

   <ChatMessages />

    <MessageBox />
  </div>
</template>
```

- Si hay warnings en la consola, estos se quedan porque Vite lo que hace es el reemplazo de módulos en caliente, no hace un full reload, no significa que no se haya solucionado

Diseño de mensajes

- Creamos un nuevo componente para manejar las burbujas de los mensajes
- Pego los mensajes dentro de ChatMessages.vue
- ChatBubble.vue

```
<template>
    <div class="flex justify-end">
          <div class="bg-blue-200 text-black p-2 rounded-lg max-w-xs">
            Hey, how's your day going?
          </div>
        </div>

        <!-- Example Received Message -->
        <div class="flex">
          <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
            Not too bad, just a bit busy. How about you?
          </div>
```

```
        </div>
</template>
```

- Renderizo ChatBubble en ChatMessages.vue
- Voy a recibir unas props en este ChatBubble.vue

```
<script lang="ts" setup>
    interface Props{
        message: string;
        itsMine: boolean; //si el mensaje es mio
        image?: string
    }
    defineProps<Props>(); //importante definir las props para pasárselas al componente
</script>
```

- Mediante el itsMine (que identifica si el mensaje es mio o no) va a tener unas características de tailwind distintas, ya que si es mio el mensaje se sitúa a la derecha, y los de mi esposa a la izquierda (básicamente si es mio tiene la clase de tailwind justify-end)
- Puedo usar **:class** y hacer un ternario
- Pero en este caso es más sencillo usar un v-if y un v-else. Si tiene el itsMine en true irá a la derecha

```
<template>
    <div v-if="itsMine" class="flex justify-end">
          <div class="bg-blue-200 text-black p-2 rounded-lg max-w-xs">
            {{ message }}
          </div>
        </div>

        <!-- Example Received Message -->
        <div v-else class="flex">
          <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
            {{ message }}
          </div>
        </div>
</template>
```

- Se podría hacer en un único componente y usar :class, usar ternarios para colocar el css, etc
- Debo pasarle el itsMine al componente que renderizo en ChatMessages
- Para mandar el true (y no un string) debo usar el v-bind (versión abreviada :)
- El message lo paso como string si no le paso el v-bind
- Para que lo pase como string lo pongo con comilla sencilla dentro de las comillas dobles

```

```

```
<ChatBubble :its-mine="true" :message="'Hola Mundo'"/>
```

- También podría pasarse así al ser un booleano en true y el message un string

```
<ChatBubble its-mine message="Hola Mundo"/>
```

- Copio la linea y le pongo el its-mine en false para tener la respuesta del mensaje de mi esposa a la izquierda

```
<ChatBubble :its-mine="true" :message="'Hola Mundo'"/>
<ChatBubble :its-mine="false" :message="'Hola Mundo'"/>
```

- Arreglamos el mensaje de mi esposa para lo que será la respuestsa de la API (con un YES o NO y una imagen)
- Como la imagen puede venir o no, usamos un v-if

```
<div v-else class="flex">
  <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
    <span class="capitalize">{{ message }}</span>
    <img v-if="image" :src="image" alt="YesNoImage" class="w-52 h-52 object-cover
rounded-md">
  </div>
</div>
```

- La API es yesno.wtf/#
- La url de la imagen desde la página de la API no funciona

```
{
  "answer": "yes",
  "forced": false,
  "image": "https://yesno.wtf/assets/yes/2.gif" //no funciona
}
```

- Mejor usamos POSTMAN con la url del sitio (sin el numeral)

> https://yesno.wtf/api

- Esto devuelve

```
{
    "answer": "yes",
    "forced": false,
```

```
    "image": "https://yesno.wtf/assets/yes/2-5df1b403f2654fa77559af1bf2332d7a.gif"
}
```

- Copiamos la url que devuelve POSTMAN y probamos la imagen para ver si queda bien

```
<ChatBubble :its-mine="false" :message="'yes'" image="https://yesno.wtf/assets/yes/
2-5df1b403f2654fa77559af1bf2332d7a.gif"/>
```

- Vamos con la lógica

Comunicación entre componentes

- Luego lo transformaremos un un composable
- La comunicación entre componentes es algo sumamente importante
- Creo una propiedad reactiva messages como un arreglo vacío en IndecisionView.vue
- Para tiparlo creo en src/interfaces/chat-message.interface.ts

```
export interface ChatMessage{
    id: number;
    message: string;
    itsMine: boolean;
    image?: string;
}
```

- Ya puedo tipar el arreglo de mensajes
- Se lo paso a ChatMessages (que todavía no lo tiene implementado)
- IndecisionView.vue

```
<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
      <span>Mi esposa</span>
    </div>

   <ChatMessages  :messages="messages"/>

    <MessageBox />
  </div>
</template>

<script setup lang="ts">
import ChatMessages from '@/components/chat/ChatMessages.vue';
import MessageBox from '@/components/chat/MessageBox.vue';
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import { ref } from 'vue';
```

```
const messages = ref<ChatMessage[]>([
  {
    id: new Date().getTime(), //debería ser un UUID
    message: "Hola mundo",
    itsMine: true
  },
  {
    id: new Date().getTime()+1, //debería ser un UUID
    message: "yes",
    itsMine: false,
    image: "https://yesno.wtf/assets/yes/2-5df1b403f2654fa77559af1bf2332d7a.gif"
  }

]);
</script>
```

- Lo implementamos en ChatMessages.vue
- Para recibir esa property uso defineProps
- Puedo usar estos mensajes con un v-for (me pide el key)

```
<template>
     <div class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="message in messages"
       :key="message.id"
       :its-mine="message.itsMine"
       :message="message.message"
       :image="message.image"
       />
      </div>
    </div>
</template>

<script setup lang="ts">
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import ChatBubble from './ChatBubble.vue';

interface Props{
  messages: ChatMessage[];
}

defineProps<Props>()
</script>
```

- Puedo desestructurar la data para no usar message. en todos lados

```
<template>
     <div class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="({id, itsMine,message,image}) in messages"
       :key="id"
       :its-mine="itsMine"
       :message="message"
       :image="image"
       />
      </div>
    </div>
</template>
```

- Pero **aún hay una forma más corta**. Todas las propiedades de message ya las he definido en las props salvo el id
- Puedo usar el v-bind (que la forma corta es :) y pasarle el message

```
<template>
     <div class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="message in messages"
       :key="message.id"
       v-bind="message"
       />
      </div>
    </div>
</template>
```

- Esto hace el mapeo directamente. Las que no va a usar (como el id) simplemente son ignoradas
- Siguiente paso. El MessageBox. Lo que tiene que hacer es mandar llamar un evento (presionar el enter) que emita el valor de la caja de texto para renderizarlo como mensaje en pantalla y hacer el llamado a la API para obtener la respuesta y traer una imagen con un si o un no
- Este mensaje debe impactar de alguna manera este arreglo de mensajes que está en IndecisionView.vue
- Cuando esta propiedad reactiva cambie va a notificar a nuestro ChatMessages (que es donde está el v-for para renderizarlos en pantalla)

Emitir Eventos - defineEmits

- Debo poder tocar el botón o presionar enter, y mandar el texto del input para crear un mensaje que se renderice al pasar a nuestro listado de mensajes
- Vamos a MessageBox

- Tenemos el input y tenemos el botón con el svg dentro
- Este es el input del cual necesitamos capturar su valor. Para ello creo una variable reactiva message
- Al inciarlo como u string vacío TypeScript infiere el tipo string por nosotros
- Para conectarlo al input usamos **v-model**
- Para añadir la función uso @keypress.enter
- Añado la función al button con @click

```
<template>
    <div class="bg-white p-4 flex items-center">
      <input
        type="text"
        placeholder="Type your message..."
        class="flex-1 border rounded-full px-4 py-2 focus:outline-none"
        v-model="message"
        @keypress.enter="sendMessage"
      />
      <button
        class="bg-blue-500 text-white rounded-full p-2 ml-2 hover:bg-blue-600
focus:outline-none"
        @click="sendMessage"
      >
        <svg
          width="20px"
          height="20px"
          viewBox="0 0 24 24"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          stroke="#ffffff"
        >
          <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
          <g id="SVGRepo_tracerCarrier" stroke-linecap="round" stroke-
linejoin="round"></g>
          <g id="SVGRepo_iconCarrier">
            <path
              d="M11.5003 12H5.41872M5.24634 12.7972L4.24158 15.7986C3.69128 17.4424
3.41613 18.2643 3.61359 18.7704C3.78506 19.21 4.15335 19.5432 4.6078 19.6701C5.13111
19.8161 5.92151 19.4604 7.50231 18.7491L17.6367 14.1886C19.1797 13.4942 19.9512 13.1471
20.1896 12.6648C20.3968 12.2458 20.3968 11.7541 20.1896 11.3351C19.9512 10.8529 19.1797
10.5057 17.6367 9.81135L7.48483 5.24303C5.90879 4.53382 5.12078 4.17921 4.59799
4.32468C4.14397 4.45101 3.77572 4.78336 3.60365 5.22209C3.40551 5.72728 3.67772 6.54741
4.22215 8.18767L5.24829 11.2793C5.34179 11.561 5.38855 11.7019 5.407 11.8459C5.42338
11.9738 5.42321 12.1032 5.40651 12.231C5.38768 12.375 5.34057 12.5157 5.24634 12.7972Z"
              stroke="#ffffff"
              stroke-width="2"
              stroke-linecap="round"
              stroke-linejoin="round"
            ></path>
          </g>
        </svg>
      </button>
    </div>
```

```
</template>

<script lang="ts" setup>
import { ref } from 'vue';

  const message= ref("");
  const sendMessage=()=>{
    if(!message.value) return //no emitimos si no hay nada escrito
    //TODO: lógica
    message.value="" //una vez se manda limpio el valor
  }

</script>
```

- **¿Cómo hago para llegar al componente padre (IndecisionView.vue) para llegar al arreglo de messages?**
- Necesitamos disparar un evento personalizado, algo así como agregarle al MessageBox un @on-message
- Para ello usaremos defineEmits trabajando con el Composition API
- Ejemplo: en este ejemplo tengo dos funciones, change y update, una emite un número y la otra un string

```
const emit = defineEmits<{
  change: [id: number]
  update: [value: string]
}>()
```

- En MessageBox.vue uso defineEmits y creo la función sendMessage que emite un testo (string)
- Guardo defineEmits en la constante emits
- Uso emits y entre paréntesis le paso la función que creé con defineEmits (sendMessage) y le paso el valor del text, que en este caso es message.value, el valor de la propiedad reactiva que enlacé al input con v-model (donde guardo el texto del input)

```
<script lang="ts" setup>
import { ref } from 'vue';

  const emits= defineEmits<{
    sendMessage: [text:string]
  }>();

  const message= ref("");
  const sendMessage=()=>{
    if(!message.value) return //no emitimos si no hay nada escrito
    emits('sendMessage', message.value) //le paso el message.value que es el valor de
la propiedad reactiva
                                        //que tengo enlazada al input con v-model
```

```
    message.value="" //una vez se manda limpio el valor
  }
</script>
```

- Ahora, si voy al IndecisionView.vue donde está el componente MessageBox y añado @ veo que tengo disponible el evento send-message
- Puedo tomar el valor emitido (text que es message.value) con $event como parámetro de una función que creemos
- Creo la función onMessage y uso .push para agregar al final del arreglo el mensaje (que será si o si mio)

```
<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
      <span>Mi esposa</span>
    </div>

   <ChatMessages  :messages="messages"/>

    <MessageBox @send-message="onMessage($event)"/>
  </div>
</template>

<script setup lang="ts">
import ChatMessages from '@/components/chat/ChatMessages.vue';
import MessageBox from '@/components/chat/MessageBox.vue';
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import { ref } from 'vue';



const messages = ref<ChatMessage[]>([
  {
    id: new Date().getTime(), //debería ser un UUID
    message: "Hola mundo",
    itsMine: true
  },
  {
    id: new Date().getTime()+1, //debería ser un UUID
    message: "yes",
    itsMine: false,
    image: "https://yesno.wtf/assets/yes/2-5df1b403f2654fa77559af1bf2332d7a.gif"
  }

]);

const onMessage=(text: string)=>{
  messages.value.push({
    id: new Date().getTime(),
    itsMine: true,
    message: text
```

```
  })
}
</script>
```

- La forma corta es como este @send-message está emitiendo un text y lo único que hago con $event es mandárselo como referencia a send-message (y onMessage también está esperando solo un text), puedo poner solo onMessage

```
<MessageBox @send-message="onMessage"/>
```

- Hay varias formas de hacer el defineEmits
- Esta es una buena manera para trabajar con el script setup
- Cuando trabajamos con el defineComponent (y la función setup()) lo que haría sería declarar un objeto emits y definirlos

```
import { defineComponent, defineEmits } from 'vue';

export default defineComponent({
  props: {
    value: { type: Number, required: true }
  },
  emits: ['sendMessage'], // Solo declaramos el evento que vamos a emitir
  setup(props) {
    // Tipamos el evento con defineEmits, indicando que 'sendMessage' espera un string
    const emit = defineEmits<{
      (event: 'sendMessage', message: string): void;
    }>();

    // Función que emite el evento 'sendMessage' con un mensaje de tipo string
    const sendMessage = (message: string) => {
      emit('sendMessage', message);
    };

    // Retornamos la función para que pueda ser usada en el template
    return {
      sendMessage
    };
  }
});
```

- Transformaremos el IndecisionView en un Composable para que quede más sencillo
- entonces, he creado un evento personalizado al que le paso la función que capta el valor emitido (text) que está guardado en la variable reactiva message, que he enlazado en el input con el v-model, y disparado usando @keypress.enter

Pensemos en Composables

- Cuando un componente empieza a crecer con mucha lógica y queremos ir trabajando con el script setup se presta mucho a separar la lógica, no solo para reutilizarla, sino para manetener el estado de la aplicación o del componente
- De hecho con el composable podría evitar tener que mandar argumentos, pero el envío y emisión de eventos valía la pena entenderlo
- Creo useChat.ts, corto la lógica de IndecisionVue.vue y la pego en el useChat, retorno el arreglo y la función en un objeto

```
import type { ChatMessage } from "@/interfaces/chat-message.interface";
import { ref } from "vue";

export const useChat=()=>{

    const messages = ref<ChatMessage[]>([
  {
    id: new Date().getTime(), //debería ser un UUID
    message: "Hola mundo",
    itsMine: true
  },
  {
    id: new Date().getTime()+1, //debería ser un UUID
    message: "yes",
    itsMine: false,
    image: "https://yesno.wtf/assets/yes/2-5df1b403f2654fa77559af1bf2332d7a.gif"
  }

]);

const onMessage=(text: string)=>{
  messages.value.push({
    id: new Date().getTime(),
    itsMine: true,
    message: text
  })
}

    return {
        messages,
        onMessage
    }
}
```

- Desestructuro el message y el onMessage de useChat en IndecisionView.vue

```
<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
```

```
      <span>Mi esposa</span>
    </div>

   <ChatMessages  :messages="messages"/>

    <MessageBox @send-message="onMessage"/>
  </div>
</template>

<script setup lang="ts">
import ChatMessages from '@/components/chat/ChatMessages.vue';
import MessageBox from '@/components/chat/MessageBox.vue';
import { useChat } from '@/components/chat/useChat';

const {messages, onMessage} =useChat()
</script>
```

- Los mensajes del arreglo de mensajes ya **los podemos borrar**, eran solo de ejemplo

Realizar petición HTTP

- Vamos a la lógica del useChat
- Hago unavalidación, si en text.length no hay nada return
- Si el mensaje no termina en ? return
- Entonces, si el mensaje acaba en ? hago la petición HTTP
- Creo la función getHerResponse
- Para tipar la respuesta cojo la respuesta que me devuelve POSTMAN, creo una nueva interfaz y uso paste json as code

```
export interface YesNoResponse {
    answer: string;
    forced: boolean;
    image:  string;
}
```

- Puedo tipar la respuesta del fetch usando la interfaz (habría que envolverlo en un try catch porque la petición podría fallar)
- Hago el onMessage async para poder usar el await con getHerResponse y desestructuro el answer y la image
- Vuelvo a hacer un .push y coloco el itsMine en false para que coloque a la izquierda la imagen y la respuesta
- useChat.ts

```
import type { ChatMessage } from "@/interfaces/chat-message.interface";
import type { YesNoResponse } from "@/interfaces/yes-no.response";
```

```
import { ref } from "vue";

export const useChat=()=>{

    const messages = ref<ChatMessage[]>([]);

    const getHerResponse =async()=>{
      const resp = await fetch("https://yesno.wtf/api")
      const data = (await resp.json()) as YesNoResponse

      return data
    }

const onMessage=async(text: string)=>{
  if(text.length === 0) return;

  messages.value.push({
    id: new Date().getTime(),
    itsMine: true,
    message: text
  })
  //evaluar si el mensaje termina con ?
  if(!text.endsWith('?')) return;
  const {answer, image}= await getHerResponse()

   messages.value.push({
    id: new Date().getTime(),
    itsMine: false,
    message: answer,
    image: image
    })
  }
    return {
        messages,
        onMessage,
    }
}
```

- Ahora, si le hago una pregunta desde el navegador, me devuelve una imagen con un yes o un no
- Pero hay un problema: cuando hacemos varias preguntas hay que hacer scroll para verlo
- De momento crearemos una función para simular un tiempo de espera para la respuesta (de mi esposa)
- Creo una carpeta src/helpers/sleep.ts

```
export const sleep = (seconds: number = 1)=>{
 return new Promise(resolve =>{
    setTimeout(resolve, seconds*1000)
```

```
 })
}
```

- Llamo a la función en useChat

```
const onMessage=async(text: string)=>{
  if(text.length === 0) return;

  messages.value.push({
    id: new Date().getTime(),
    itsMine: true,
    message: text
  })

  if(!text.endsWith('?')) return;
  await sleep(1.5) //retraso 1.5 segundos la respuesta
  const {answer, image}= await getHerResponse()

   messages.value.push({
    id: new Date().getTime(),
    itsMine: false,
    message: answer,
    image: image
  })
}
```

- Hay que poner un scroll automático cuando recibimos mensajes

Referencias a elementos HTML

- Una manera es hacer referencia al div de ChatMessages.vue y decirle que cuando haya un nuevo mensaje (cuando nuestra property de los messages cambia) quiero moverlo a la posición final
- Creo una variable reactiva con ref
- Si no le especifico un valor es undefined, por eso la inicializo con null
- Le paso la referencia chatRef al div

```
<template>
     <div ref="chatRef" class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="message in messages"
       :key="message.id"
       v-bind="message"
       />
      </div>
    </div>
</template>
```

```
<script setup lang="ts">
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import ChatBubble from './ChatBubble.vue';
import { ref } from 'vue';

interface Props{
  messages: ChatMessage[];
}

defineProps<Props>()

  //arreglar scroll
const chatRef= ref<HTMLDivElement| null>(null)

console.log(chatRef.value) // devuelve null

</script>
```

- Si coloco el console.log dentro de un setTimeOut de un segundo, veré que ya tengo la referencia al div
- Esto deja claro que con el script setup, primero se ejecuta el código de TypeScript, y luego cuando se construye el HTML se asigna la referencia
- Hay funciones con el onMounted pero no es lo que me interesa
- Lo que me interesa es mover la pantalla cuando se recibe un nuevo mensaje
- Para estar pendiente de cuando recibo nuevos mensajes uso la función **watchEffect**
- watchEffect realiza la lógica del callback cada vez que hay un cambio en la reactividad del bloque
- Hay que darle un poco de tiempo a Vue para que le de tiempo de renderizar el elemento (si no el scroll no acaba de llegar al final)
- Para ello usamos nextTick, como necesita el await hago async el callback
- Usando scrollHeight como el valor de top en scrollTo, estás diciendo que el contenedor debe desplazarse hasta el final del contenido. - Esto lleva el scroll de tu contenedor hasta la parte más baja

```
<template>
     <div ref="chatRef" class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="message in messages"
       :key="message.id"
       v-bind="message"
       />
      </div>
    </div>
</template>

<script setup lang="ts">
```

```
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import ChatBubble from './ChatBubble.vue';
import { nextTick, ref, watchEffect } from 'vue';

interface Props{
  messages: ChatMessage[];
}

defineProps<Props>()

//arreglar scroll
const chatRef= ref<HTMLDivElement| null>(null)

watchEffect(async () => {
    await nextTick();

    chatRef.value?.scrollTo({
        top: chatRef.value.scrollHeight,
        behavior: 'smooth',
    });

})

</script>
```

- Paso todos los componentes de la aplicación
- App.vue

```
<template>
 <IndecisionView />
</template>

<script lang="ts" setup>
import IndecisionView from './views/IndecisionView.vue';

</script>
```

- views/IndecisionView.vue

```
<!-- Fuente: https://tailwindcomponents.com/component/chat-layout -->
<template>
  <div class="bg-gray-100 h-screen flex flex-col max-w-lg mx-auto">
    <div class="bg-blue-500 p-4 text-white flex justify-between items-center">
      <span>Mi esposa</span>
    </div>

   <ChatMessages  :messages="messages"/>

    <MessageBox @send-message="onMessage"/>
  </div>
</template>
```

```vue
<script setup lang="ts">
import ChatMessages from '@/components/chat/ChatMessages.vue';
import MessageBox from '@/components/chat/MessageBox.vue';
import { useChat } from '@/components/chat/useChat';


const {messages, onMessage} =useChat()


</script>
```

• components/chat/ChatMessages.vue

```vue
<template>
     <div ref="chatRef" class="flex-1 overflow-y-auto p-4">
      <div class="flex flex-col space-y-2">
       <ChatBubble
       v-for="message in messages"
       :key="message.id"
       v-bind="message"
       />
      </div>
    </div>
</template>

<script setup lang="ts">
import type { ChatMessage } from '@/interfaces/chat-message.interface';
import ChatBubble from './ChatBubble.vue';
import { nextTick, ref, watchEffect } from 'vue';

interface Props{
  messages: ChatMessage[];
}

defineProps<Props>()

//arreglar scroll
const chatRef= ref<HTMLDivElement| null>(null)

watchEffect(async () => {

    await nextTick();
    chatRef.value?.scrollTo({
        top: chatRef.value.scrollHeight,
        behavior: 'smooth',
    });

})
```

```
</script>
```

• components/chat/ChatBubble.vue

```
<template>
    <div v-if="itsMine" class="flex justify-end">
          <div class="bg-blue-200 text-black p-2 rounded-lg max-w-xs">
            {{ message }}
          </div>
        </div>

        <!-- Example Received Message -->
        <div v-else class="flex">
          <div class="bg-gray-300 text-black p-2 rounded-lg max-w-xs">
            <span class="capitalize">{{ message }}</span>
            <img v-if="image" :src="image" alt="YesNoImage" class="w-52 h-52 object-
cover rounded-md">
          </div>
        </div>
</template>


<script lang="ts" setup>
    interface Props{
        message: string;
        itsMine: boolean; //si el mensaje es mio
        image?: string
    }
    defineProps<Props>();
</script>
```

• components/chat/MessageBox.vue

```
<template>
    <div class="bg-white p-4 flex items-center">
      <input
        type="text"
        placeholder="Type your message..."
        class="flex-1 border rounded-full px-4 py-2 focus:outline-none"
        v-model="message"
        @keypress.enter="sendMessage"
      />
      <button
        class="bg-blue-500 text-white rounded-full p-2 ml-2 hover:bg-blue-600
focus:outline-none"
        @click="sendMessage"
      >
        <svg
          width="20px"
          height="20px"
          viewBox="0 0 24 24"
```

```
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          stroke="#ffffff"
        >
          <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
          <g id="SVGRepo_tracerCarrier" stroke-linecap="round" stroke-
linejoin="round"></g>
          <g id="SVGRepo_iconCarrier">
            <path
              d="M11.5003 12H5.41872M5.24634 12.7972L4.24158 15.7986C3.69128 17.4424
3.41613 18.2643 3.61359 18.7704C3.78506 19.21 4.15335 19.5432 4.6078 19.6701C5.13111
19.8161 5.92151 19.4604 7.50231 18.7491L17.6367 14.1886C19.1797 13.4942 19.9512 13.1471
20.1896 12.6648C20.3968 12.2458 20.3968 11.7541 20.1896 11.3351C19.9512 10.8529 19.1797
10.5057 17.6367 9.81135L7.48483 5.24303C5.90879 4.53382 5.12078 4.17921 4.59799
4.32468C4.14397 4.45101 3.77572 4.78336 3.60365 5.22209C3.40551 5.72728 3.67772 6.54741
4.22215 8.18767L5.24829 11.2793C5.34179 11.561 5.38855 11.7019 5.407 11.8459C5.42338
11.9738 5.42321 12.1032 5.40651 12.231C5.38768 12.375 5.34057 12.5157 5.24634 12.7972Z"
              stroke="#ffffff"
              stroke-width="2"
              stroke-linecap="round"
              stroke-linejoin="round"
            ></path>
          </g>
        </svg>
      </button>
    </div>
</template>

<script lang="ts" setup>
import { ref } from 'vue';

  const emits= defineEmits<{
    sendMessage: [text:string]
  }>();

  const message= ref("");
  const sendMessage=()=>{
    if(!message.value) return //no emitimos si no hay nada escrito
    emits('sendMessage', message.value)
    message.value="" //una vez se manda limpio el valor
  }
</script>
```

• helpers/useChat.ts

```
import { sleep } from "@/helpers/sleep";
import type { ChatMessage } from "@/interfaces/chat-message.interface";
import type { YesNoResponse } from "@/interfaces/yes-no.response";
import { ref } from "vue";

export const useChat=()=>{

    const messages = ref<ChatMessage[]>([]);
```

```
    const getHerResponse =async()=>{
      const resp = await fetch("https://yesno.wtf/api")
      const data = (await resp.json()) as YesNoResponse

      return data
    }

const onMessage=async(text: string)=>{
  if(text.length === 0) return;

  messages.value.push({
    id: new Date().getTime(),
    itsMine: true,
    message: text
  })
  //evaluar si el mensaje termina con ?
  if(!text.endsWith('?')) return;
  await sleep(1.5)
  const {answer, image}= await getHerResponse()

   messages.value.push({
    id: new Date().getTime(),
    itsMine: false,
    message: answer,
    image: image
  })


}

    return {
        messages,
        onMessage,
    }
}
```

05 Vue Herrera - Pokemon Game

- Nos vamos a enfocar a crear un nuego de Pokemon
- Quién es ese Pokemon? Y seleccionar uno del listado
- Con un contador (las veces que se ha ganado o perdido)
- Rango de pokemones (los primeros 51, o los 200, los 800)
- Vamos a ir excluyendo las selecciones (siempre serán 4 pokemones entre los que elegir)
- En la pantalla se verá ¿Quién es el pokemon?, debajo una silueta y debajo cuatro cajas con los diferentes nombres a elegir
- Cuando se clique en una de las opciones se iluminarán todas remarcando la correcta y aparecerá la opción jugar de nuevo
- Cuando se acierta veremos confeti
- Tailwind, clases condicionales, emisiones de eventos, manejo de properties, manejar el estado, etc
- Trabajaremos con el Composition API y el script setup, perfectamente lo podremos desarrollar d esta manera si usar defineComponents

Inicio del proyecto

> npm create vue@latest

- name: pokemon-game
- TypeScript: si
- JSX: no
- Vue Router: no
- Pinia: no
- Vitest: si
- E2E: no
- EsLint: si
- Prettier: si
- Entro en la carpeta del proyecto y le doy a npm install
- Tenemos varias maneras de organizar el filesystem
- Por tipos (con todos los components en /components)
- Por tipos y features (cuando la aplicación es un poco más grande, con cada componente en su carpeta dentro de components)
    - El inconveniente de esta manera es que hay muchas carpetas duplicadas, es dificil de rastrear, mantener...

- Screaming architecture (dividir por módulos, core, payments, y dentro de cada módulo components, composables, store, lib, etc)
    - De esta manera agrupamos el código por módulos
    - Trabajaremos de esta manera
- En App.vue, uso el snippet vbase y selecciono vbase-3-ts-setup (Vue VsCode Snippets)
- Creo src/assets/styles.css
- Lo importo en el main.ts

```
import './assets/styles.css'
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

- Instalamos tailwind

> npm install tailwindcss @tailwindcss/vite

- vite.config.ts

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
    tailwindcss()
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
  },
})
```

- En el styles.css

```

```

```
@import "tailwindcss";
```

- Cuando uses clases de tailwind en dentro del style scoped hay que usar @referenece “tailwindcss”
- Ejemplo

```
<style scoped>
  @reference "tailwindcss";
</style>
```

Estructura de la aplicación

- src/modules/pokemon/composables
- src/modules/pokemon/components
- src/modules/pokemon/pages/PokemonGame.vue
- Creamos un loading en PokemonGame.vue
- Con w-screen y h-screen lo centro en la pantalla
- El animate-pulse hace que parpadee
- Una vez visto el resultado renderizándolo en App.vue le coloco un v-if=false para no mostrarlo
- Debajo del section del loader vamos a tener el juego, abro otro section
- PokemonGame.vue

```
<template>
    <section v-if="false" class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section class="flex flex-col justify-center items-center w-screen h-screen">
        <h1>¿Quién es este Pokemón?</h1>

        <!--Pokemon Picture-->

        <!--Pokemon Options-->

    </section>
</template>

<script setup lang="ts">

</script>

<style scoped>
```

```
</style>
```

- pokemon/components/PokemonPicture.vue
- En la pokeapi las imágenes están en sprites/other/dream_world/front_default
- Colocamos el string en el src de la imágen para ver cómo queda al renderizarlo en PokemonGame.vue
- Le coloco el brightness-0 y unas clases para que no se pueda arrastrar la imagen oscura y ver cuál es

```
<template>
  <section>
    <img
    class="brightness-0 h-[200px]"
    src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/
other/official-artwork/132.png">
  </section>
</template>

<script setup lang="ts">

</script>

<style scoped>
/*para que la persona pno pueda mover la imagen oscura y ver cual es*/
img{
    user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    -webkit-user-drag: none;
    -webkit-user-select:none
}
</style>
```

- Vamos con PokemonOptions.vue, hagámoslo provisional

```
<template>
  <section class="mt-5">
    <ul>
        <li>Pókemon 1</li>
        <li>Pókemon 2</li>
        <li>Pókemon 3</li>
        <li>Pókemon 4</li>
    </ul>
  </section>
</template>

<script setup lang="ts">

</script>
```

```
<style scoped>

</style>
```

• Lo renderizo también en PokemonGame.vue

```
<template>
    <section v-if="false" class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section class="flex flex-col justify-center items-center w-screen h-screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>

        <PokemonPicture />
        <PokemonOptions />

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';


</script>

<style scoped>
@reference "tailwindcss";

li{
    @apply bg-white rounded-lg shadow-md p-3 m-2 cursor-pointer w-40 text-center
transition-all hover:bg-gray-100
}
</style>
```

• Puedo aplicarle unas clases temporales porque luego los voy a transformar en botones

• En styles.css añado unas clases para el color de fondo

```
@import "tailwindcss";

html, body{
    background-color: #f1f1f1;
}
```

• Creo en src/assets/animation.css y pego este código

```
.fade-in {
    animation: fadeIn 0.3s;
    -webkit-animation: fadeIn 0.3s;
    -moz-animation: fadeIn 0.3s;
    -o-animation: fadeIn 0.3s;
    -ms-animation: fadeIn 0.3s;
  }
  @keyframes fadeIn {
    0% {opacity:0;}
    100% {opacity:1;}
  }

  @-moz-keyframes fadeIn {
    0% {opacity:0;}
    100% {opacity:1;}
  }

  @-webkit-keyframes fadeIn {
    0% {opacity:0;}
    100% {opacity:1;}
  }

  @-o-keyframes fadeIn {
    0% {opacity:0;}
    100% {opacity:1;}
  }

  @-ms-keyframes fadeIn {
    0% {opacity:0;}
    100% {opacity:1;}
  }
```

- Esto hace que cuando añada la clase fade-in añade un efecto de fade-in, para cuando mostremos el pokemon que es

Enumeraciones y tipado de datos

- Voy a iniciar la lógica en el composable
- En pokemon/composables/usePokemonGame.ts
- El juego tiene tres estados: cuando se gana, se pierde y cuando se está jugando
- Este estado lo almacenaré en una variable reactiva gameStatus
- Para tiparlo creo pokemon/interfaces/game-status.enum.ts
- Aunque los enum ni los types sean interfaces, colocaré aquí todo lo que es tipado
- game-status.enum.ts

```
export enum GameStatus{
 Playing = 'playing',
 Won = 'won',
```

```
 Lost='lost'
}
```

Puedo crear un archivo de barril en interfaces/index.ts
- index.ts

```
export * from './game-status.enum';
```

- Ahora ya puedo usarlo en usePokemonGame.ts
- Necesito crear el listado random de pokemons
- Si en la pokeapi uso limit=151 me trae 151 pokemons. Trabajaremos con eso
- Hagámoslo con axios

> npm i axios

- pokemon/api/pokemon-api.ts

```
import axios from 'axios'


const pokemonApi = axios.create({
    baseURL: "https://pokeapi.co/api/v2/pokemon"
})


export {pokemonApi} //después usaremos middlewares
```

- Llamo con el onMounted a getPokemons, que es para cuando se monte se ejecute, aunque en este caso es redundante ya que al pasar por el script setup se va a montar igual si invocaramos la función sin el onMounted

```
import { onMounted, ref } from "vue"
import { GameStatus } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)

    const getPokemons = async() =>{
        const response = await pokemonApi.get('/?limit=151')
        console.log(response)
    }

    //getPokemons()

    onMounted(()=>{
```

```
        getPokemons()
    })

    return {
        gameStatus
    }
}
```

- Usemos el usePokemonGame en PokemonGame.vue, desestructuremos el gameStatus para que TS no se queje
- PokemonGame.vue

```
<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';

const {gameStatus} = usePokemonGame()
</script>
```

- Deberíamos poder ver en consola el resultado de la petición gracias al console.log
- Los pokemons están en data/results (results es un arreglo de objetos con name, url)
- Lo que me gustaría tener es el id y el nombre del pokemon
- Y también quiero tipar la respuesta. Abrimos POSTMAN, hago la petición y uso paste json as code
- En interfaces/pokemon-list.response.ts

```
export interface PokemonListResponse {
    count:    number;
    next:     string;
    previous: null;
    results:  Result[];
}

export interface Result {
    name: string;
    url:  string;
}
```

- Lo coloco en el archivo de barril
- Ahora ya puedo tipar la respuesta de axios en el composable usePokemonGame

```
const getPokemons = async() =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')
        console.log(response)
```

```
    }
```

Propiedades computadas y orden aleatrorio

- Lo que necesito es el id del pokemon (que viene en la url en la última posición antes del /) y el nombre
- Creo la pokemon.interface.ts

```
export interface Pokemon{
    id: number
    name: string
}
```

- Incluyo la interfaz en el archivo de barril

```
export * from './game-status.enum';
export * from './pokemon-list.response';
export * from './pokemon.interface'
```

- La url de un pokemon es “https://pokeapi.co/api/v2/pokemon/1/”
- Uso el split para dividir la url por /, me interesa la posición -2 que es dónde está el id
- Uso el .at para indicarle la posición. Este método solo está disponible en JS 2022
- Guardo el id en una constante y le pongo que si no hay id le ponga 0 para que TS no se queje si viene undefined
- Parseo el id a número en el return con un +
- Ahora si puedo regresar el pokemonsArray
- En el onmounted guardo el array en pokemons y los imprimo en consola

```
import { onMounted, ref } from "vue"
import { GameStatus, type Pokemon, type PokemonListResponse } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)

    const getPokemons = async(): Promise<Pokemon[]> =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')

        const pokemonsArray= response.data.results.map(pokemon=>{
            const urlParts= pokemon.url.split('/') //lo corto por el /, el id está en
la posición -2
            const id = urlParts.at(-2) ?? 0; //para que no se queje si el id es
```

```
undefined

            return{
                name: pokemon.name,
                id: +id  //parseo el id a número
            }
        })

        return pokemonsArray
    }

      onMounted(async ()=>{
       const pokemons= await getPokemons()
       console.log(pokemons)
    })

    return {
        gameStatus
    }
}
```

- Para poder usar .at he tenido que agregar estas dos lineas (target y lib) al compilerOptions
- tsconfig.app.json

```
{
  "extends": "@vue/tsconfig/tsconfig.dom.json",
  "include": ["env.d.ts", "src/**/*", "src/**/*.vue"],
  "exclude": ["src/**/__tests__/*"],
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "target": "ES2022",
    "lib": ["ES2022", "dom"],

    "paths": {
      "@/*": ["./src/*"]
    }
  }
}
```

- Veo en consola que tengo el name y el id en el array
- Ahora voy a necesitar mezclarlos para tenerlos de manera aleatoria
- Math.random va desde 0 a 0.99999999. Es lo que voy a usar cuando retorno el array de pokemons

```
return pokemonsArray.sort(()=> Math.random() - 0.5)
```

- Creo una variable reactiva pokemons y en lugar de crear la constante pokemons en el onMounted, uso pokemons.value para guardar los pokemons

- El isLoading es una propiedad computada, ya que depende de si hay o no elementos en pokemons (la variable reactiva que acabao de crear donde guardo los pokemons)

```
import { computed, onMounted, ref } from "vue"
import { GameStatus, type Pokemon, type PokemonListResponse } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)

    const pokemons = ref<Pokemon[]>([])

    const isLoading= computed(()=> pokemons.value.length === 0)

    const getPokemons = async(): Promise<Pokemon[]> =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')

        const pokemonsArray= response.data.results.map(pokemon=>{
            const urlParts= pokemon.url.split('/') //lo corto por el /, el id está en
la posición -2
            const id = urlParts.at(-2) ?? 0; //para que no se queje si el id es
undefined

            return{
                name: pokemon.name,
                id: +id  //parseo el id a número
            }
        })

        return pokemonsArray.sort(()=> Math.random() - 0.5)

    }

    onMounted(async ()=>{
       pokemons.value= await getPokemons()
       console.log(pokemons)
    })

    return {
        gameStatus,
        isLoading
    }
}
```

- Ya puedo usar el isLoading para renderizar el Loading de PokemonGame.vue con un v-if (y el juego con un v-else)

```
<template>
    <section v-if="isLoading"
class="flex flex-col justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
```

```
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>

        <PokemonPicture />
        <PokemonOptions />

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';

const {gameStatus, isLoading} = usePokemonGame()
</script>

<style scoped>

</style>
```

- También necesito crearme una variable reactiva para las opciones a elegir de pokemon en usePokemonGame.ts
- La llamaré pokemonOptions
- Creo la función getNextOptions para manejarla
- Le paso cuántos pokemons quiero mostrar en la lista para elegir (para hacerlo más reutilizable si quiero añadirle dificultad)
- Le pongo 4 por defecto
- En pokemons.value tengo todos los pokemones
- Uso slice para cortar desde la posición 0 a la indicada por howMany
- slice no modifica el arreglo, ahora yo debo eliminar esos 4 pokemons del arreglo original
- Vuelvo a usar el slice, pero ahora solo le paso el howmany, para que corte los primeros 4 y quede el arreglo sin ellos
- getNextOptions va a ser el botón que inicie un nuevo juego
- Llamo a la función en el onMounted (si no le paso un valor tiene 4 por defecto)
- usePokemonGame.ts

```
import { computed, onMounted, ref } from "vue"
import { GameStatus, type Pokemon, type PokemonListResponse } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)
```

```
    const pokemons = ref<Pokemon[]>([])
    const pokemonOptions= ref<Pokemon[]>([]) //aquí guardaré las 4 opciones de pokemon
a elegir

    const isLoading= computed(()=> pokemons.value.length === 0)

    const getPokemons = async(): Promise<Pokemon[]> =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')

        const pokemonsArray= response.data.results.map(pokemon=>{
            const urlParts= pokemon.url.split('/')
            const id = urlParts.at(-2) ?? 0;

            return{
                name: pokemon.name,
                id: +id  //parseo el id a número
            }
        })

        return pokemonsArray.sort(()=> Math.random() - 0.5)

    }

    const getNextOptions = (howMany: number= 4) =>{
        gameStatus.value = GameStatus.Playing
        pokemonOptions.value= pokemons.value.slice(0, howMany) //me quedo con los
primeros 4
        pokemons.value = pokemons.value.slice(howMany) //el array se queda sin los
primeros 4
    }

    onMounted(async ()=>{
       pokemons.value= await getPokemons()
       getNextOptions()

       //console.log(pokemonOptions.value)  --> tengo 4 pokemons
    })

    return {
        gameStatus,
        isLoading,
        pokemonOptions,
        getNextOptions
    }
}
```

Determinar el pokemon correcto

- Necesito escoger uno de los pokemons de pokemonOptions para que sea el correcto (el de la imagen)
- Hagámoslo como unapropiedad computada randomPokemon

- Uso Math.floor para aplanar el resultado sin decimales (obtener un entero), que redondee, y muliplico el Math.random por la cantidad de pokemons
- Retorno randomPokemon para poder desestructurarlo del composable y renderizarlo en PokemonGame.vue
- usePokemonGame.ts

```
const pokemonOptions= ref<Pokemon[]>([])

    const randomPokemon= computed(()=>{
        return pokemonOptions.value[Math.floor(Math.random()*
pokemonOptions.value.length)]
    })
```

- Renderizo el randomPokemon en PokemonGame

```
<template>
    <section v-if="isLoading"
class="flex flex-col justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <h3>{{ randomPokemon }}</h3>

        <PokemonPicture />
        <PokemonOptions />

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';

const {gameStatus, isLoading, randomPokemon} = usePokemonGame()
</script>

<style scoped>

</style>
```

Componente PokemonPicture

- Ya tenemos el pokemon correcto (randomPokemon, que está en la lista de las 4 opciones a elegir)

- Ahora debo mandarle la imagen a PokemonPicture, lo haremos a través del id
- PokemonGame.vue

```
<PokemonPicture :pokemon-id="randomPokemon?.id"/>
```

- En PokemonPicture recibamos esta property
- Defino las props
- Hago una propiedad computada basada en el pokemon que le mando
- Hago un bind en el src para pasarle la imagen

```
<template>
  <section>
    <img
    class="brightness-0 h-[200px]"
    :src="pokemonImage">
  </section>
</template>

<script setup lang="ts">
import { computed } from 'vue';


  interface Props{
    pokemonId: number | undefined
  }

const props=  defineProps<Props>();

  const pokemonImage = computed(
    ()=> `https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/
other/dream-world/${props.pokemonId}.svg`
  )

</script>

<style scoped>
/*para que la persona pno pueda mover la imagen oscura y ver cual es*/
img{
    user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    -webkit-user-drag: none;
    -webkit-user-select:none
```

```
}
</style>
```

- Ahora ya tengo la imagen de una de las 4 opciones, el nombre y la imagen de randomPokemon es el correcto
- En PokemonGame.vue añado una condición al v-if, que randomPokemon.id sea null

```
<template>
    <section v-if="isLoading || randomPokemon?.id == null" class="flex flex-col
justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <h3>{{ randomPokemon }}</h3>

        <PokemonPicture :pokemon-id="randomPokemon?.id"/>
        <PokemonOptions />

    </section>
</template>
```

- Tenemos que recibir otra propiedad para mostrar la imagen real del pokemon (con brightness y que no se vea osucra)
- Llamo a la property show-pokemon, es un boolean. Lo pongo en true para hacer la prueba, pero será dinámico

```
<PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="true"/>
```

- En PokemonPicture.vue, puedo usar la función withDefaults pasándole las props, y de segundo argumento un objeto con las props por defecto
- Uso un v-if con el !showPokemon para la imagen en false por defecto (con el brightness a 0, se ve una sombra de la imagen) y un v-else para mostrar la imagen real (con colores)
- Le añado la calse fade-in que tengo en el animations.css (tiene que estar importado en el main!)

```
<template>
  <section>
    <img v-if="!showPokemon" class="brightness-0 h-[200px]" :src="pokemonImage">
    <img v-else class=" fade-in h-[200px]" :src="pokemonImage">
  </section>
</template>

<script setup lang="ts">
```

```
import { computed } from 'vue';


  interface Props{
    pokemonId: number | undefined
    showPokemon?: boolean
  }

const props=  withDefaults(defineProps<Props>(), {
  showPokemon: false
});

  const pokemonImage = computed(
    ()=> `https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/
other/dream-world/${props.pokemonId}.svg`
  )

</script>

<style scoped>
/*para que la persona pno pueda mover la imagen oscura y ver cual es*/
img{
    user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    -webkit-user-drag: none;
    -webkit-user-select:none
}
</style>
```

- Determinaremos este true y false según el gameStatus
- En playing estará en false, en won o lost estará en true (mostrará la imagen)
- Desestructuro el gameStatus del composable y hago la logica en el :show-pokemon
- Se muestra solo si no está jugando

```
<PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="gameStatus !=
GameStatus.Playing"/>
```

Mostrar las posibles opciones

- Es el momento de trabajar con el componente PokemonOptions
- Desde PokemonGame,vue, Puedo usar un alias en la desestructuración del composable para no confundirme usando :
- PokemonGame.vue

```
<template>
    <section v-if="isLoading || randomPokemon?.id == null" class="flex flex-col
justify-center items-center w-screen h-screen">
```

```
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <h3>{{ randomPokemon }}</h3>

        <PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="gameStatus !=
GameStatus.Playing"/>
        <PokemonOptions :options="options"/>

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';
import { GameStatus } from '../interfaces';
                                                    //uso un alias
const {gameStatus, isLoading, randomPokemon, pokemonOptions:options} = usePokemonGame()
</script>

<style scoped>

</style>
```

- En PokemonOptions.vue tengo que definir las props
- Uso un v-for en el botón, siempre debo pasarle el key
- Uso la clase capitalize para que paraezca la primera letra del nombre en mayúsculas
- Cambio li por button en el style

```
<template>
  <section class="mt-5 flex flex-col">
    <button v-for="{name, id} in options" :key="id" class="capitalize">
      {{ name }}
    </button>
  </section>
</template>

<script setup lang="ts">
import type { Pokemon } from '../interfaces';


interface Props{
  options: Pokemon[]
}

defineProps<Props>()
</script>
```

```
<style scoped>
@reference "tailwindcss";

button{
    @apply bg-white rounded-lg shadow-md p-3 m-2 cursor-pointer w-40 text-center transition-all hover:bg-gray-100
}
</style>
```

- Cuando toco una de las opciones (alguno de los botones) yo tengo que **emitir** qué opción es
- Uso defineEmits en el script setup de PokemonOptions
- Puedo usar el tipado entre llaves, poniendo la propiedad a emitir tipada entre corchetes

```
defineEmits<{
  selectedOption: [id: number]
}>()
```

- Tengo que decirle al componente que emita el valor, para ello usamos **@click**
- Le paso el evento entre comillas, y como segundo parámetro el valor

```
<button v-for="{name, id} in options" :key="id" class="capitalize"
@click="$emit('selectedOption', id)">
{{ name }}
</button>
```

- Vayamos al componente padre (PokemonGame.vue) a recoger ese evento con @selected-option
- Hagamos una función onSelectedOption con un console.log para comprobar que el id se está pasando correctamente

```
<template>
    <section v-if="isLoading || randomPokemon?.id == null" class="flex flex-col justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <h3>{{ randomPokemon }}</h3>

        <PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="gameStatus != GameStatus.Playing"/>
        <PokemonOptions :options="options" @selected-option="onSelectedOption"/>

    </section>
</template>
```

```
<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';
import { GameStatus } from '../interfaces';
                                                    //uso un alias
const {gameStatus, isLoading, randomPokemon, pokemonOptions:options} = usePokemonGame()

const onSelectedOption= (value: number)=> console.log(value)
</script>

<style scoped>

</style>
```

- Cuando el jugador elige una opción vamos a bloquear los botones
- Ahora me interesa determinar que la persona ganó
- Si el id del botón presionado hace match con el del randomPokemon la persona ganó

Determinar si la persona gana o pierde

- Renderizamos en PokemonGame.vue el gameStatus en vez del randomPokemon, para que se vea Playing si estamos jugando, Win si ganamos o Lost si perdemos
- En el composable usePokemonGame.ts es donde haremos la lógica, con la función checkAnswer
- Para el confetti usaremos canvas-confetti (npm i canvas-confetti, instalo los @types)
- usePokemonGame.ts

```
import { computed, onMounted, ref } from "vue"
import { GameStatus, type Pokemon, type PokemonListResponse } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"
import confetti from 'canvas-confetti'

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)
    const pokemons = ref<Pokemon[]>([])
    const pokemonOptions= ref<Pokemon[]>([])

    const randomPokemon= computed(()=>{
        return pokemonOptions.value[Math.floor(Math.random()*
pokemonOptions.value.length)]
    })

    const isLoading= computed(()=> pokemons.value.length === 0)

    const getPokemons = async(): Promise<Pokemon[]> =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')
```

```
        const pokemonsArray= response.data.results.map(pokemon=>{
            const urlParts= pokemon.url.split('/') //lo corto por el /, el id está en
la posición -2
            const id = urlParts.at(-2) ?? 0; //para que no se queje si el id es
undefined

            return{
                name: pokemon.name,
                id: +id  //parseo el id a número
            }
        })

        return pokemonsArray.sort(()=> Math.random() - 0.5)

    }

    const getNextOptions = (howMany: number= 4) =>{
        gameStatus.value = GameStatus.Playing
        pokemonOptions.value= pokemons.value.slice(0, howMany)
        pokemons.value = pokemons.value.slice(howMany)
    }

    const checkAnswer=(id: number)=>{
        const hasWon = randomPokemon.value?.id === id
        if(hasWon){
            gameStatus.value = GameStatus.Won
            confetti({
                particleCount: 300,
                spread: 150,
                origin: {y:0.6}
            })
        }
    }

    onMounted(async ()=>{
       pokemons.value= await getPokemons()
       getNextOptions()
    })

    return {
        gameStatus,
        isLoading,
        pokemonOptions,
        getNextOptions,
        randomPokemon,
        checkAnswer
    }
}
```

- Desestructuro en el componente padre PokemonGame checkAnswer del composable usePokemonGame
- Coloco la función en el @selected-option

```
<template>
    <section v-if="isLoading || randomPokemon?.id == null" class="flex flex-col
justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <h3>{{ gameStatus }}</h3>

        <PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="gameStatus !=
GameStatus.Playing"/>
        <PokemonOptions :options="options" @selected-option="checkAnswer"/>

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';
import { GameStatus } from '../interfaces';
                                                    //uso un alias
const {gameStatus, isLoading, randomPokemon, pokemonOptions:options, checkAnswer} =
usePokemonGame()

</script>

<style scoped>

</style>
```

- Ahora, si le doy a la opción indicada en uno de los botones, cambia el status solo si acierto (pone Won) y salta el confetti
- Falta la lógica de si la persona no gana en el composable usePokemonGame.ts

```
const checkAnswer=(id: number)=>{
        const hasWon = randomPokemon.value?.id === id
        if(hasWon){
            gameStatus.value = GameStatus.Won
            confetti({
                particleCount: 300,
                spread: 150,
                origin: {y:0.6}
            })
            return
        }
```

```
        gameStatus.value= GameStatus.Lost
    }
```

- De esta manera, cuando pierdo me muestra la imagen y cambia el status, pero todavía puedo darle a otras opciones hasta acertar
- Hay que bloquear los botones una vez ganó o perdió y añadir un botón para un nuevo juego

Bloquear opciones

- Vamos a mandar una nueva property a PokemonOptions, la llamaremos block-selection

```
<PokemonOptions
        :options="options"
        @selected-option="checkAnswer"
        :block-selection="gameStatus != GameStatus.Playing"/>
```

- Añado la prop en el componente PokemonOptions
- Quiero saber cuando fallé cuál era la opción correcta, remarquémosla
- Puedo ponerle a class :, que quede **:class** y usar corchetes (entre comillas) para poner clases de CSS y loógica
- Cuando mando un objeto es una forma condicional de mandar CSS
- Necesito recibir la respuesta correcta en el PokemonOptions para aplicar clases de manera condicional a los botones, para mostrar en azul si ha acertado y en rojo cuando sea una opción incorrecta
- Le paso otra property llamada correct-answer a PokemonOptions que recibirá desde el padre PokemonGame
- Debo declararla en las props de PokemonOptions
- La clase correct o incorrect solo se aplicará si el blockSelection está en true, para que no se muestren marcadas las opciones sin ni siquiera haber jugado

```
<template>
  <section class="mt-5 flex flex-col">
    <button v-for="{name, id} in options" :key="id"
    :class="['capitalize disabled:shadow-none disbaled:bg-gray-300',{
      correct: id === correctAnswer && blockSelection,
      incorrect: id !== correctAnswer && blockSelection
    }]"
    @click="$emit('selectedOption', id)"
    :disabled="blockSelection"
    >
      {{ name }}
    </button>
  </section>
</template>
```

```
<script setup lang="ts">
import type { Pokemon } from '../interfaces';


interface Props{
  options: Pokemon[],
  blockSelection: boolean,
  correctAnswer: number
}

defineProps<Props>()

defineEmits<{
  selectedOption: [id: number]
}>()

</script>

<style scoped>
@reference "tailwindcss";

button{
    @apply bg-white rounded-lg shadow-md p-3 m-2 cursor-pointer w-40 text-center
transition-all hover:bg-gray-100
}

.correct{
  @apply bg-blue-500 text-white
}

.incorrect{
  @apply bg-red-100 opacity-70
}
</style>
```

Nuevo juego

- Un cambio. Para cambiar el nombre de una función en todos los lugares presionaremos F2 con la función clicada
- Cambiaremos GetNextOptions por getNextRound
- usePokemonGames.ts

```
import { computed, onMounted, ref } from "vue"
import { GameStatus, type Pokemon, type PokemonListResponse } from "../interfaces"
import { pokemonApi } from "../api/pokemon-api"
import confetti from 'canvas-confetti'

export const usePokemonGame =()=>{

    const gameStatus = ref<GameStatus>(GameStatus.Playing)
```

```
    const pokemons = ref<Pokemon[]>([])
    const pokemonOptions= ref<Pokemon[]>([])

    const randomPokemon= computed(()=>{
        return pokemonOptions.value[Math.floor(Math.random()*
pokemonOptions.value.length)]
    })

    const isLoading= computed(()=> pokemons.value.length === 0)

    const getPokemons = async(): Promise<Pokemon[]> =>{
        const response = await pokemonApi.get<PokemonListResponse>('/?limit=151')

        const pokemonsArray= response.data.results.map(pokemon=>{
            const urlParts= pokemon.url.split('/') //lo corto por el /, el id está en
la posición -2
            const id = urlParts.at(-2) ?? 0; //para que no se queje si el id es
undefined

            return{
                name: pokemon.name,
                id: +id  //parseo el id a número
            }
        })

        return pokemonsArray.sort(()=> Math.random() - 0.5)

    }

    const getNextRound = (howMany: number= 4) =>{
        gameStatus.value = GameStatus.Playing
        pokemonOptions.value= pokemons.value.slice(0, howMany)
        pokemons.value = pokemons.value.slice(howMany)
    }

    const checkAnswer=(id: number)=>{
        const hasWon = randomPokemon.value?.id === id
        if(hasWon){
            gameStatus.value = GameStatus.Won
            confetti({
                particleCount: 300,
                spread: 150,
                origin: {y:0.6}
            })
            return
        }
        gameStatus.value= GameStatus.Lost
    }

    onMounted(async ()=>{
       pokemons.value= await getPokemons()
       getNextRound()
    })

    return {
        gameStatus,
```

```
        isLoading,
        pokemonOptions,
        getNextRound,
        randomPokemon,
        checkAnswer,

    }
}
```

- Ahora si, creemos un botón que dispare la función y que aparezca solo cuando no se esté en GameStatus.Playing
- Pongo el botón dentro de un div para darle espacio
- PokemonGame.vue

```
<template>
    <section v-if="isLoading || randomPokemon?.id == null" class="flex flex-col
justify-center items-center w-screen h-screen">
        <h1 class="text-3xl">Esper, por favor</h1>
        <h3 class="animate-pulse">Cargando pokemons...</h3>
    </section>
    <section v-else class="flex flex-col justify-center items-center w-screen h-
screen">
        <h1 class="m-5">¿Quién es este Pokemón?</h1>
        <div class="h-20">
            <button class="m-3 p-4 bg-blue-500 text-white rounded-md"
            v-if="gameStatus != GameStatus.Playing"
            @click="getNextRound(4)"
            >¿Nuevo juego?</button>
        </div>


        <PokemonPicture :pokemon-id="randomPokemon?.id" :show-pokemon="gameStatus !=
GameStatus.Playing"/>
        <PokemonOptions
        :options="options"
        @selected-option="checkAnswer"
        :block-selection="gameStatus != GameStatus.Playing"
        :correct-answer="randomPokemon.id"
        />

    </section>
</template>

<script setup lang="ts">
import PokemonOptions from '../components/PokemonOptions.vue';
import PokemonPicture from '../components/PokemonPicture.vue';
import { usePokemonGame } from '../composables/usePokemonGame';
import { GameStatus } from '../interfaces';
                                                    //uso un alias
const {gameStatus, isLoading, randomPokemon, pokemonOptions:options, checkAnswer,
getNextRound} = usePokemonGame()
```

```
</script>

<style scoped>

</style>
```

- Podemos configurar el path para usar alias y reducir el largo de las importaciones
- En vite.config.ts

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
    tailwindcss()
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url)),
      '@pokemon': fileURLToPath(new URL('./src/modules/pokemon', import.meta.url)),
    },
  },
})
```

- En el tsconfig.app.json también debemos añadir el nuevo alias

```
{
  "extends": "@vue/tsconfig/tsconfig.dom.json",
  "include": ["env.d.ts", "src/**/*", "src/**/*.vue"],
  "exclude": ["src/**/__tests__/*"],
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "target": "ES2022",
    "lib": ["ES2022", "dom"],

    "paths": {
      "@/*": ["./src/*"],
      "@pokemon/*": ["./src/modules/pokemon/*"]
    }
  }
}
```

Explicacion argumentos función setup y eventos

Te explico: en la Composition API de Vue 3, `setup()` es la función donde defines la lógica reactiva de tu componente. La diferencia está en **qué necesita acceso a la instancia del componente** y qué no.

Lo que NO se pasa como argumentos:

```
import { ref, computed, onMounted } from 'vue'

export default defineComponent({
  setup() {
    // Variables reactivas básicas
    const count = ref(0)
    const double = computed(() => count.value * 2)

    // Hooks del ciclo de vida
    onMounted(() => {
      console.log('Componente montado')
    })

    // Funciones normales
    const increment = () => {
      count.value++
    }

    return { count, double, increment }
  }
})
```

Lo que SÍ se pasa como argumentos:

El `setup()` recibe dos parámetros cuando necesita interactuar con:

1. `props` - Acceder a las props

```
export default defineComponent({
  props: {
    title: String,
    initialCount: Number
  },

  setup(props) {  // ← props como primer argumento
    // Usar props dentro de lógica reactiva
    const count = ref(props.initialCount || 0)
    const titleWithCount = computed(() =>
      `${props.title}: ${count.value}`
    )
```

```
    return { count, titleWithCount }
  }
})
```

2. `context` - Acceder al contexto del componente

```
export default defineComponent({
  setup(props, context) {  // ← context como segundo argumento

    // Acceder a:
    // a) Eventos emitidos
    const emitEvent = () => {
      context.emit('custom-event', 'datos')  // Emitir eventos
    }

    // b) Slots
    console.log(context.slots.default())  // Acceder a slots

    // c) Attrs (atributos no declarados como props)
    console.log(context.attrs.class)  // Clases CSS no declaradas

    // d) Exponer métodos públicos (raro, pero posible)
    context.expose({ publicMethod: () => {} })

    return { emitEvent }
  }
})
```

Ejemplo completo combinado:

```
import { ref, computed } from 'vue'

export default defineComponent({
  props: {
    initialValue: {
      type: Number,
      default: 0
    }
  },

  emits: ['value-changed'],  // Eventos declarados

  setup(props, { emit }) {  // ← Desestructuración de context

    // Lógica reactiva usando props
    const count = ref(props.initialValue)

    const increment = () => {
      count.value++
```

```
      emit('value-changed', count.value)  // Emitir evento
    }

    const double = computed(() => count.value * 2)

    return {
      count,
      increment,
      double
    }
  }
})
```

¿Por qué esta diferencia?

1. **`props` se pasa como argumento** porque:
   - Necesitas acceder a valores que vienen del padre
   - Vue necesita saber que tu `setup()` depende de ciertas props para la reactividad

2. **`context` se pasa como argumento** porque:
   - Proporciona acceso a características específicas de la instancia del componente
   - Permite emitir eventos, acceder a slots, etc.

3. **Lo demás no se pasa** porque:
   - `ref()`, `computed()`, `onMounted()` son funciones independientes del ciclo de vida
   - No necesitan conocer la instancia específica del componente para funcionar

Alternativa con `<script setup>` (SFC):

En componentes de un solo archivo (Single File Components):

```
<script setup>
// Props y emits se declaran de forma diferente
const props = defineProps(['initialValue'])
const emit = defineEmits(['value-changed'])

// El resto igual que antes
const count = ref(props.initialValue)
const increment = () => {
  count.value++
  emit('value-changed', count.value)
}
</script>
```

Conclusión: Pasas como argumentos solo lo que necesita interactuar directamente con la instancia del componente (props y contexto). La lógica reactiva pura no necesita estos argumentos.

Ejemplo completo con `<script setup>`, `defineEmits` y padre

Componente Hijo (ChildComponent.vue)

Este componente emitirá eventos al padre:

```
<template>
  <div class="child">
    <h3>Componente Hijo</h3>
    <p>Contador interno: {{ internalCount }}</p>

    <button @click="increment">Incrementar +1</button>
    <button @click="reset">Reiniciar a 0</button>
    <button @click="sendCustomData">
      Enviar datos personalizados
    </button>

    <!-- Input para enviar texto -->
    <div style="margin-top: 20px">
      <input
        v-model="message"
        placeholder="Escribe un mensaje"
        @keyup.enter="sendMessage"
      />
      <button @click="sendMessage">Enviar Mensaje</button>
    </div>
  </div>
</template>

<script setup>
import { ref, defineEmits, defineProps } from 'vue'

// 1. Definir las props que recibe (opcional)
const props = defineProps({
  initialCount: {
    type: Number,
    default: 0
  }
})

// 2. Definir los eventos que emite
const emit = defineEmits([
  'increment',          // Evento simple
  'reset',              // Evento sin datos
```

```
  'custom-event',       // Evento con datos complejos
  'message-sent',       // Evento con string
  'update:count'        // Convención para v-model
])

// 3. Estado interno del componente hijo
const internalCount = ref(props.initialCount)
const message = ref('')

// 4. Funciones que emiten eventos
const increment = () => {
  internalCount.value++

  // Emitir evento con datos (número)
  emit('increment', internalCount.value)

  // También emitir con formato v-model
  emit('update:count', internalCount.value)
}

const reset = () => {
  internalCount.value = 0

  // Emitir evento sin datos
  emit('reset')

  // Emitir para v-model
  emit('update:count', internalCount.value)
}

const sendCustomData = () => {
  // Emitir evento con objeto complejo
  emit('custom-event', {
    count: internalCount.value,
    timestamp: new Date().toISOString(),
    randomId: Math.random().toString(36).substring(7)
  })
}

const sendMessage = () => {
  if (message.value.trim()) {
    // Emitir evento con string
    emit('message-sent', message.value)
    message.value = ''
  }
}
</script>

<style scoped>
.child {
  border: 2px solid #42b883;
  padding: 20px;
  border-radius: 10px;
  margin: 10px;
  background-color: #f9f9f9;
}
```

```
button {
  margin: 5px;
  padding: 8px 16px;
  background-color: #42b883;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #33a06f;
}

input {
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
  margin-right: 10px;
}
</style>
```

Componente Padre (ParentComponent.vue)

Este componente recibe y maneja los eventos:

```
<template>
  <div class="parent">
    <h2>Componente Padre</h2>

    <!-- Mostrar estado del padre -->
    <div class="stats">
      <p>Contador en padre: {{ parentCount }}</p>
      <p>Último mensaje: {{ lastMessage || 'Ninguno aún' }}</p>
      <p>Último evento personalizado:</p>
      <pre>{{ lastCustomEvent || 'Ninguno aún' }}</pre>
    </div>

    <!-- Usar componente hijo -->
    <ChildComponent
      :initial-count="5"
      @increment="handleIncrement"
      @reset="handleReset"
      @custom-event="handleCustomEvent"
      @message-sent="handleMessage"
      @update:count="handleCountUpdate"
      v-model:count="parentCount"  <!-- Sintaxis v-model -->
    />

    <!-- Otro hijo independiente -->
    <ChildComponent
```

```
      @increment="(count) => console.log('Segundo hijo:', count)"
    />
  </div>
</template>

<script setup>
import { ref } from 'vue'
import ChildComponent from './ChildComponent.vue'

// Estado del componente padre
const parentCount = ref(0)
const lastMessage = ref('')
const lastCustomEvent = ref(null)

// Funciones manejadoras de eventos
const handleIncrement = (newCount) => {
  console.log('Evento "increment" recibido:', newCount)
  // Aquí podrías actualizar estado del padre si es necesario
}

const handleReset = () => {
  console.log('Evento "reset" recibido')
  parentCount.value = 0
}

const handleCustomEvent = (data) => {
  console.log('Evento personalizado recibido:', data)
  lastCustomEvent.value = JSON.stringify(data, null, 2)
}

const handleMessage = (message) => {
  console.log('Mensaje recibido:', message)
  lastMessage.value = message
  alert(`Mensaje del hijo: "${message}"`)
}

const handleCountUpdate = (newValue) => {
  console.log('Evento update:count recibido:', newValue)
  // parentCount ya se actualiza automáticamente con v-model
}
</script>

<style scoped>
.parent {
  border: 3px solid #35495e;
  padding: 25px;
  border-radius: 12px;
  max-width: 600px;
  margin: 20px auto;
  background-color: #f0f8ff;
}

.stats {
  background-color: white;
  padding: 15px;
  border-radius: 8px;
```

```
  margin-bottom: 20px;
  border: 1px solid #ddd;
}

pre {
  background-color: #f5f5f5;
  padding: 10px;
  border-radius: 5px;
  font-size: 14px;
  overflow-x: auto;
}

h2 {
  color: #35495e;
  margin-bottom: 20px;
}
</style>
```

App.vue (Componente raíz)

```
<template>
  <div id="app">
    <h1>Comunicación Padre-Hijo en Vue 3</h1>
    <p>Usando script setup y defineEmits</p>

    <ParentComponent />
  </div>
</template>

<script setup>
import ParentComponent from './components/ParentComponent.vue'
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 20px;
}

body {
  margin: 0;
  padding: 20px;
  background-color: #f8f9fa;
}
</style>
```

Características clave demostradas:

1. `defineEmits()` para declarar eventos en el hijo
2. `emit('nombre-evento', datos)` para emitir eventos
3. `@evento="handler"` para escuchar en el padre
4. **Tipos de datos enviados:**
   - Números: `emit('increment', count)`
   - Strings: `emit('message-sent', message)`
   - Objetos complejos: `emit('custom-event', { ... })`
   - Sin datos: `emit('reset')`
5. **v-model personalizado** con `update:propName`
6. **Múltiples hijos** independientes

Para ejecutar este ejemplo:

1. Crea un proyecto Vue 3:

```
npm create vue@latest
# o
npm init vue@latest
```

1. Crea los componentes en `src/components/`
2. Reemplaza el contenido de `App.vue`

La comunicación fluye así: **Hijo emite → Padre escucha → Padre ejecuta función**

¡Excelente pregunta! Hay dos formas distintas de emitir eventos en Vue, y la confusión es común:

1. `$emit` - Options API (Vue 2 / Options API de Vue 3)

Se usa **dentro del template** o en métodos con `this`:

```
<!-- Options API - Template -->
<template>
  <button @click="$emit('increment', 1)">Incrementar</button>
  <!-- $emit directamente en el template -->
</template>

<script>
export default {
  methods: {
```

```
    increment() {
      this.$emit('increment', 1) // $emit con this
    }
  }
}
</script>
```

2. `emit()` - Composition API (Vue 3 con setup())

Se obtiene del contexto y se usa en el `setup()`:

```
<!-- Composition API con <script setup> -->
<script setup>
import { defineEmits } from 'vue'

// Declarar los eventos
const emit = defineEmits(['increment', 'reset'])

const handleClick = () => {
  emit('increment', 1) // Usar la función emit
}
</script>

<template>
  <!-- Usamos función, NO $emit directo -->
  <button @click="handleClick">Incrementar</button>
  <!-- O directamente en el template: -->
  <button @click="emit('increment', 1)">Otro botón</button>
</template>
```

3. `emit()` - Composition API (setup() function)

```
<script>
import { defineComponent } from 'vue'

export default defineComponent({
  setup(props, { emit }) { // emit desestructurado del contexto
    const handleClick = () => {
      emit('increment', 1)
    }

    return { handleClick }
  }
})
</script>
```

Comparación lado a lado:

Característica	`$emit` (Options API)	`emit()` (Composition API)
Donde se usa	En template o métodos	En script setup o setup()
Acceso	`this.$emit()`	`const emit = defineEmits()`
Declaración	`emits: ['evento']`	`defineEmits(['evento'])`
En template	`@click="$emit('evento')"`	`@click="emit('evento')"`
Recomendación	Vue 2 o Options API	Vue 3 Composition API

Ejemplo práctico completo:

Componente Hijo (ChildComponent.vue)

```
<script setup>
import { ref } from 'vue'

// Definir eventos que se emitirán
const emit = defineEmits(['saludo', 'contador'])

const nombre = ref('Juan')
const contador = ref(0)

// Función que emite evento
function enviarSaludo() {
  emit('saludo', `¡Hola desde ${nombre.value}!`)
}

// Función que emite con datos
function incrementar() {
  contador.value++
  emit('contador', {
    valor: contador.value,
    timestamp: new Date().toISOString()
  })
}

// Emitir directamente en template también es válido
const emitDirecto = () => emit('saludo', 'Mensaje directo')
</script>

<template>
  <div>
    <!-- Opción 1: Llamar a función que emite -->
    <button @click="enviarSaludo">Saludar (función)</button>
```

```
    <!-- Opción 2: Emitir directamente en template -->
    <button @click="emit('saludo', 'Directo desde template')">
      Saludar (directo)
    </button>

    <!-- Opción 3: Usar función intermedia -->
    <button @click="emitDirecto">Saludar (intermedio)</button>

    <button @click="incrementar">Contador: {{ contador }}</button>
  </div>
</template>
```

Componente Padre (ParentComponent.vue)

```
<script setup>
import { ref } from 'vue'
import ChildComponent from './ChildComponent.vue'

const mensajeRecibido = ref('')
const ultimoContador = ref(null)

// Función que maneja el evento 'saludo'
function manejarSaludo(mensaje) {
  mensajeRecibido.value = mensaje
  console.log('Saludo recibido:', mensaje)
}

// Función que maneja el evento 'contador'
function manejarContador(datos) {
  ultimoContador.value = datos
  console.log('Contador actualizado:', datos)
}
</script>

<template>
  <div>
    <h2>Mensaje: {{ mensajeRecibido }}</h2>
    <p>Último contador: {{ ultimoContador?.valor }}</p>

    <!-- Escuchar eventos del hijo -->
    <ChildComponent
      @saludo="manejarSaludo"
      @contador="manejarContador"
    />

    <!-- También puedes usar función inline -->
    <ChildComponent
      @saludo="(msg) => console.log('Inline:', msg)"
    />
  </div>
</template>
```

¿Por qué el cambio en Composition API?

1. **Sin `this`**: Composition API no usa `this`, por eso no hay `this.$emit`
2. **Mejor tipado**: `defineEmits` permite tipado TypeScript
3. **Más explícito**: Declaras explícitamente qué eventos emites

Tipado TypeScript (opcional pero recomendado):

```
<script setup lang="ts">
// Con tipado TypeScript
const emit = defineEmits<{
  saludo: [mensaje: string]  // Un parámetro string
  contador: [datos: { valor: number, timestamp: string }]  // Objeto
  reset: []  // Sin parámetros
}>()

// Ahora TypeScript valida los tipos:
emit('saludo', 'Hola') // ✓ Correcto
emit('saludo', 123) // ✗ Error: debe ser string
emit('contador', { valor: 1, timestamp: '...' }) // ✓ Correcto
</script>
```

Resumen:

- **`$emit`** → Options API (Vue 2 o Vue 3 Options API)
- **`emit()`** → Composition API (Vue 3 con `<script setup>` o `setup()`)
- **En el padre**, siempre recibes igual: `@evento="funcionManejadora"`
- **El manejador** recibe automáticamente los argumentos emitidos

Regla mnemotécnica: Si usas `<script setup>`, usa `defineEmits()` y `emit()`. Si usas Options API (con `export default {}`), usa `$emit`.

---

07 Vue Herrera - Router

- Tendremos un Home, Features, Pricing, Contact, Pokemons, Login
- Pondremos un contador en el Home, al que si yo le cambio el valor y me muevo entre pantallas, seguirá manteniendo el número que había dejado en el contador. Lo haremos sin un gestor de estado, sino con algo del Vue Router llamado keep alive
- En el login noharemos autenticación todavía
- Si hago Login (simplemente clicando el botón de Login) guardo en el localstorage userId: ABC-123
- En pantalla vemos un pokémon con un id, que se rige por la url. Hay un botón de siguiente que incrementa el id del pokemon en 1 en la url, mostrando el pokemon correspondiente a ese id
- La url va a lucir tipo localhost:5173/pokemon/1
- Es una Single Page Aplication

Inicio del proyecto

> npm create vue@latest

- name vue-spa
- Router: si //mostraremos como configurarlo
- Vitest: si
- ESLint: si
- Prettier: si
- En la carpeta del proyecto

> npm i

- Configuramos Tailwind

> npm install tailwindcss @tailwindcss/vite

- vite.config.ts

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
import tailwindcss from '@tailwindcss/vite'
```

```
// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
    tailwindcss(),
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
  },
})
```

- En styles.css

```
@import "tailwindcss";
```

- Debo importar styles.css en el main

```
import './assets/styles.css'
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')
```

Preparación de la estructura de archivos

- src/assets/styles.css
- src/modules
- src/modules/landing/pages/ContactPage.vue
- En pages creo también FeaturesPages.vue, HomePage.vue, PricingPage.vue
- Aprendamos a instalar el Vue Router

Instalación de Vue Router

> npm i vue-router@4

- src/router/index.ts

```
import HomePage from '@/landing/pages/HomePage.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home', //opcional
      component: HomePage
    }
  ],
})

export default router
```

- En el main.ts

```
import './assets/styles.css'
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')
```

- En lugar de llamar al componente en el router, podemos usar una función de flecha y usar import con la dirección del componente
- Cuando se haga el build se crean unos chunks
- Usando la función se carga la página solo cuando se entra en ella (lazy loading)

```
import HomePage from '@/modules/landing/pages/HomePage.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home', //opcional
      component: HomePage
    },
    {
      path: '/features',
      name: 'features', //opcional
      component: ()=>import('@/modules/landing/pages/FeaturesPage.vue')
    },
```

```
    {
      path: '/pricing',
      name: 'pricing', //opcional
      component: ()=>import('@/modules/landing/pages/PricingPage.vue')
    },
    {
      path: '/contact',
      name: 'contact', //opcional
      component: ()=>import('@/modules/landing/pages/ContactPage.vue')
    },
  ],
})

export default router
```

- En App.vue tengo que usar el RouterView
- Creo un nav con links para la navegación (provisional)

```
<template>
  <nav>
    <a href="/">Home</a>
    <a href="/features">Features</a>
    <a href="/pricing">Pricing</a>
    <a href="/contact">Contact</a>
  </nav>

  <main>
    <RouterView />
  </main>
</template>

<style scoped></style>

<script setup lang="ts">
</script>
```

- De esta manera el nav se ve en todas las páginas (Features, Pricing,etc)

Re-utilización de estructura HTML

- En el router usamos createWebHistory pero también hay un createWebHashHistory (con la que aparece # en la url)
- ¿Cuál es la diferencia?
- Usaremos el createWebHashHistory cuando no se puede cambiar l amanera en que la aplicación es serivda del lado del backend
- Por ejemplo, cuando la subes a githubPages, Netlify en versión gratuita, donde el servidor solo sirve archivos html, css y js, y no puedes indicarle al backend que todas las rutas de mi

app (los requests, solicitudes, que están llegando a mi backend) los redireccione a mi aplicación

- La parte después del # es manejada completamente por el navegador, lo que significa que no se hace una solicitud al servidor cuando cambian las rutas. Solo cambia lo que se muestra en el navegador, pero el servidor no recibe ninguna solicitud diferente.
- Es útil cuando no tienes control sobre el servidor o no puedes configurarlo para que redirija todas las solicitudes a tu index.html. En este caso, el navegador solo necesita saber qué ruta está activa en la aplicación, pero el servidor no tiene que preocuparse por las rutas de la SPA.
- Si subes tu aplicación a plataformas como GitHub Pages o Netlify en su versión gratuita, estas plataformas solo sirven archivos estáticos (HTML, CSS, JS). No puedes decirle al servidor que todas las rutas deben redirigir a tu aplicación. Por eso, el uso de # en las URLs es más sencillo y funciona en estos entornos.
- Si quieres usar createWebHistory (sin el # en el url) se necesita porder servir la app de tal manera que se le diga que en todas las requests que vengan que sirvan esta aplicación. Es más SEO friendly
  Esto significa que el navegador sigue el flujo normal de URLs y el servidor tiene que estar configurado para manejar estas rutas.
- ¿Por qué es mejor?
- El uso de URLs limpias sin # tiene ventajas en términos de SEO (optimización para motores de búsqueda) y la estética de la URL, ya que se ve más profesional y “nativa” en la web. Además, las herramientas modernas para la web tienden a favorecer el uso de URLs normales.
- Entonces, **si vamos a subir la app a la nube y no tenemos acceso al backend: createWebHashHistory**
- Aunque trabaje con el createWebhashHistory no usaré el # en el href de los anchor tags
- Se usa el elemento **RouterLink**
- No se usa href, se usa **to**

```
<template>
  <nav>
    <RouterLink to="/">Home</RouterLink>
    <RouterLink to="/features">Features</RouterLink>
    <RouterLink to="/pricing">Pricing</RouterLink>
    <RouterLink to="/contact">Contact</RouterLink>
  </nav>

  <main>
    <RouterView />
  </main>
</template>

<style scoped></style>
```

```
<script setup lang="ts">
</script>
```

- Tenemos un gist en klerith/VueRouterTemplate.vue
- Lo pego en App.vue
- Uso RouterLink, le coloco las rutas, copio el favicon.ico y lo pego en assets, lo renombro a logo.ico

```
<template>
  <div class="flex flex-col h-screen">
    <!-- Header -->
    <header
class="flex items-center h-14 px-4 border-b border-gray-300 sm:h-16 md:px-6 lg:px-8">
      <div>
        <a class="flex items-center gap-2 font-semibold" href="#">
          <img alt="Vue logo" class="logo" src="@/assets/logo.ico" width="40"
height="40" />
        </a>
      </div>
      <nav class="ml-auto space-x-4 flex items-center h-10 sm:space-x-6">
        <RouterLink to="/"> Home </RouterLink>
        <RouterLink to="/features"> Features </RouterLink>
        <RouterLink to="/pricing"> Pricing </RouterLink>
        <RouterLink to="/contact"> Contact </RouterLink>
      </nav>
    </header>
    <!-- Fin Header -->

    <!-- Main -->
    <main class="flex-1 flex items-center justify-center py-6">
      <div class="text-center">
        <h1 class="text-4xl font-bold tracking-tighter sm:text-5xl">
          Bienvenido a nuestro sitio web
        </h1>
        <p class="mx-auto max-w-[600px] text-gray-500 md:text-xl">
          Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        </p>
      </div>
    </main>
    <!-- Fin Main -->

    <!-- Footer -->
    <footer
class="flex items-center h-14 px-4 border-t border-gray-300 sm:h-16 md:px-6 lg:px-8">
      <p class="flex-1 text-sm text-gray-500 text-center">
        © 20xx Acme Corporation. Derechos reservados
      </p>
    </footer>
    <!-- Fin Footer -->
```

```
  </div>
</template>
```

- Quiero que todas mis páginas tengan este header con el nav
- Todas van a tener este footer
- El único contenido que va a cambiar es lo que hay debajo del comentario de Main (en App.vue)
- Entonces, lo borro y coloco dentro del div el RouterView

```
<template>
  <div class="flex flex-col h-screen">
    <!-- Header -->
    <header
class="flex items-center h-14 px-4 border-b border-gray-300 sm:h-16 md:px-6 lg:px-8">
      <div>
        <a class="flex items-center gap-2 font-semibold" href="#">
          <img alt="Vue logo" class="logo" src="@/assets/logo.ico" width="40"
height="40" />
        </a>
      </div>
      <nav class="ml-auto space-x-4 flex items-center h-10 sm:space-x-6">
        <RouterLink to="/"> Home </RouterLink>
        <RouterLink to="/features"> Features </RouterLink>
        <RouterLink to="/pricing"> Pricing </RouterLink>
        <RouterLink to="/contact"> Contact </RouterLink>
      </nav>
    </header>
    <!-- Fin Header -->

    <!-- Main -->
    <main class="flex-1 flex items-center justify-center">
        <RouterView />
    </main>
    <!-- Fin Main -->

    <!-- Footer -->
    <footer
class="flex items-center h-14 px-4 border-t border-gray-300 sm:h-16 md:px-6 lg:px-8">
      <p class="flex-1 text-sm text-gray-500 text-center">
        © 20xx Acme Corporation. Derechos reservados
      </p>
    </footer>
    <!-- Fin Footer -->
  </div>
</template>
```

- Para tener siempre actualizado el año del footer podemos usar new Date

```
<footer class="flex items-center h-14 px-4 border-t border-gray-300 sm:h-16 md:px-6
lg:px-8">
```

```
      <p class="flex-1 text-sm text-gray-500 text-center">
        © {{ new Date().getFullYear() }} Acme Corporation. Derechos reservados
      </p>
</footer>
```

Tailwind Components

- Puedo usar el v-bind en el to del RouterLink y mandarle un objeto
- Con Ctrl+Space veo todas las opciones disponibles: force, hash, name, params, path, query, replace y state
- Con name puedo usar el name que indiqué en el router

```
<nav class="ml-auto space-x-4 flex items-center h-10 sm:space-x-6">
  <RouterLink :to="{name: 'home'}"> Home </RouterLink>
  <RouterLink :to="{name: 'features' }"> Features </RouterLink>
  <RouterLink :to="{name: 'pricing'}"> Pricing </RouterLink>
  <RouterLink :to="{name: 'contact'}"> Contact </RouterLink>
</nav>
```

- En el home page coloco esto

```
<div class="text-center">
  <h1 class="text-4xl font-bold tracking-tighter sm:text-5xl">
    Bienvenido a nuestro sitio web
  </h1>
  <p class="mx-auto max-w-[600px] text-gray-500 md:text-xl">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  </p>
</div>
```

- En TailwindComponents hay componentes gratuitos (algunos son responsive otros no)
- Para componentes de features visitar https://www.creative-tim.com/twcomponents/component/statics-viewer
- Se le da al botón ShowCode o al clipboard para copiar el código
- Lo pego en el template de FeaturesPage.vue

```
<!-- component -->
<section class="text-gray-600 body-font">
  <div class="container px-5 py-24 mx-auto">
    <div class="flex flex-col text-center w-full mb-20">
      <h1 class="sm:text-3xl text-2xl font-medium title-font mb-4 text-
gray-900">Master Cleanse Reliac Heirloom</h1>
      <p class="lg:w-2/3 mx-auto leading-relaxed text-base">Whatever cardigan tote bag
tumblr hexagon brooklyn asymmetrical gentrify, subway tile poke farm-to-table. Franzen
you probably haven't heard of them man bun deep jianbing selfies heirloom prism food
```

```
truck ugh squid celiac humblebrag.</p>
    </div>
    <div class="flex flex-wrap -m-4 text-center">
      <div class="p-4 md:w-1/4 sm:w-1/2 w-full">
        <div class="border-2 border-gray-200 px-4 py-6 rounded-lg">
          <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="text-indigo-500 w-12 h-12 mb-3 inline-block"
viewBox="0 0 24 24">
            <path d="M8 17l4 4 4-4m-4-5v9"></path>
            <path d="M20.88 18.09A5 5 0 0018 9h-1.26A8 8 0 103 16.29"></path>
          </svg>
          <h2 class="title-font font-medium text-3xl text-gray-900">2.7K</h2>
          <p class="leading-relaxed">Downloads</p>
        </div>
      </div>
      <div class="p-4 md:w-1/4 sm:w-1/2 w-full">
        <div class="border-2 border-gray-200 px-4 py-6 rounded-lg">
          <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="text-indigo-500 w-12 h-12 mb-3 inline-block"
viewBox="0 0 24 24">
            <path d="M17 21v-2a4 4 0 00-4-4H5a4 4 0 00-4 4v2"></path>
            <circle cx="9" cy="7" r="4"></circle>
            <path d="M23 21v-2a4 4 0 00-3-3.87m-4-12a4 4 0 010 7.75"></path>
          </svg>
          <h2 class="title-font font-medium text-3xl text-gray-900">1.3K</h2>
          <p class="leading-relaxed">Users</p>
        </div>
      </div>
      <div class="p-4 md:w-1/4 sm:w-1/2 w-full">
        <div class="border-2 border-gray-200 px-4 py-6 rounded-lg">
          <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="text-indigo-500 w-12 h-12 mb-3 inline-block"
viewBox="0 0 24 24">
            <path d="M3 18v-6a9 9 0 0118 0v6"></path>
            <path d="M21 19a2 2 0 01-2 2h-1a2 2 0 01-2-2v-3a2 2 0 012-2h3zM3 19a2 2 0
002 2h1a2 2 0 002-2v-3a2 2 0 00-2-2H3z"></path>
          </svg>
          <h2 class="title-font font-medium text-3xl text-gray-900">74</h2>
          <p class="leading-relaxed">Files</p>
        </div>
      </div>
      <div class="p-4 md:w-1/4 sm:w-1/2 w-full">
        <div class="border-2 border-gray-200 px-4 py-6 rounded-lg">
          <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" class="text-indigo-500 w-12 h-12 mb-3 inline-block"
viewBox="0 0 24 24">
            <path d="M12 22s8-4 8-10V5l-8-3-8 3v7c0 6 8 10 8 10z"></path>
          </svg>
          <h2 class="title-font font-medium text-3xl text-gray-900">46</h2>
          <p class="leading-relaxed">Places</p>
        </div>
      </div>
    </div>
```

```
  </div>
</section>
```

- Con el pricing hay que hacer un poco de carpintería que luego mejoraremos
- Copio los links de los íconos y los coloco primero dentro del template
- Luego copio todo lo que hay en el body (sin la etiqueta body) y lo pego después de los links (dentro del template)
- https://www.creative-tim.com/twcomponents/component/pricing-sections

```
<template>
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?
family=Montserrat:wght@100;200;300;400;500;600;700;800;900&display=swap"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <div class="min-h-screen flex justify-center items-center">
        <div class="">
            <div class="text-center font-semibold">
                <h1 class="text-5xl">
                    <span class="text-blue-700 tracking-wide">Flexible </span>
                    <span>Plans</span>
                </h1>
                <p class="pt-6 text-xl text-gray-400 font-normal w-full px-8 md:w-
full">
                    Choose a plan that works best for you and<br/> your team.
                </p>
            </div>
            <div class="pt-24 flex flex-row">
                <!-- Basic Card -->
                <div class="w-96 p-8 bg-white text-center rounded-3xl pr-16 shadow-xl">
                    <h1 class="text-black font-semibold text-2xl">Basic</h1>
                    <p class="pt-2 tracking-wide">
                        <span class="text-gray-400 align-top">$ </span>
                        <span class="text-3xl font-semibold">10</span>
                        <span class="text-gray-400 font-medium">/ user</span>
                    </p>
                    <hr class="mt-4 border-1">
                    <div class="pt-8">
                        <p class="font-semibold text-gray-400 text-left">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                Get started with <span class="text-black">messaging</
span>
                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
                                done
                            </span>
```

```
                            <span class="pl-2">
                                Flexible <span class="text-black">team meetings</span>
                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                <span class="text-black">5 TB</span> cloud storage
                            </span>
                        </p>

                        <a href="#" class="">
                            <p class="w-full py-4 bg-blue-600 mt-8 rounded-xl text-
white">
                                <span class="font-medium">
                                    Choose Plan
                                </span>
                                <span class="pl-2 material-icons align-middle text-sm">
                                    east
                                </span>
                            </p>
                        </a>
                    </div>
                </div>
                <!-- StartUp Card -->
                <div class="w-80 p-8 bg-gray-900 text-center rounded-3xl text-white
border-4 shadow-xl border-white transform scale-125">
                    <h1 class="text-white font-semibold text-2xl">Startup</h1>
                    <p class="pt-2 tracking-wide">
                        <span class="text-gray-400 align-top">$ </span>
                        <span class="text-3xl font-semibold">24</span>
                        <span class="text-gray-400 font-medium">/ user</span>
                    </p>
                    <hr class="mt-4 border-1 border-gray-600">
                    <div class="pt-8">
                        <p class="font-semibold text-gray-400 text-left">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                All features in <span class="text-white">Basic</span>
                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                Flexible <span class="text-white">call scheduling</
span>

                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
```

```
                                done
                            </span>
                            <span class="pl-2">
                                <span class="text-white">15 TB</span> cloud storage
                            </span>
                        </p>

                        <a href="#" class="">
                            <p class="w-full py-4 bg-blue-600 mt-8 rounded-xl text-
white">
                                <span class="font-medium">
                                    Choose Plan
                                </span>
                                <span class="pl-2 material-icons align-middle text-sm">
                                    east
                                </span>
                            </p>
                        </a>
                    </div>
                    <div class="absolute top-4 right-4">
                        <p class="bg-blue-700 font-semibold px-4 py-1 rounded-full
uppercase text-xs">Popular</p>
                    </div>
                </div>
                <!-- Enterprise Card -->
                <div class="w-96 p-8 bg-white text-center rounded-3xl pl-16 shadow-xl">
                    <h1 class="text-black font-semibold text-2xl">Enterprise</h1>
                    <p class="pt-2 tracking-wide">
                        <span class="text-gray-400 align-top">$ </span>
                        <span class="text-3xl font-semibold">35</span>
                        <span class="text-gray-400 font-medium">/ user</span>
                    </p>
                    <hr class="mt-4 border-1">
                    <div class="pt-8">
                        <p class="font-semibold text-gray-400 text-left">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                All features in <span class="text-black">Startup</span>
                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                Growth <span class="text-black">oriented</span>
                            </span>
                        </p>
                        <p class="font-semibold text-gray-400 text-left pt-5">
                            <span class="material-icons align-middle">
                                done
                            </span>
                            <span class="pl-2">
                                <span class="text-black">Unlimited</span> cloud storage
```

```
                            </span>
                        </p>

                        <a href="#" class="">
                            <p class="w-full py-4 bg-blue-600 mt-8 rounded-xl text-
white">
                                <span class="font-medium">
                                    Choose Plan
                                </span>
                                <span class="pl-2 material-icons align-middle text-sm">
                                    east
                                </span>
                            </p>
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</template>

<script setup lang="ts">

</script>

<style scoped>

</style>
```

- Vamos con el de contacto (que tiene un mapa de fondo y un pequeño formulario). Lo pego dentro del template de ContactPage.vue
- Le añado al section el flex flex-1 h-full para que tome todo el ancho
- https://www.creative-tim.com/twcomponents/component/contact-map

```
<!-- component -->
<section class="text-gray-600 body-font relative flex flex-1 h-full">
  <div class="absolute inset-0 bg-gray-300">
    <iframe width="100%" height="100%" frameborder="0" marginheight="0"
marginwidth="0" title="map" scrolling="no" src="https://maps.google.com/maps?
width=100%&height=600&hl=en&q=%C4%B0zmir+
(My%20Business%20Name)&ie=UTF8&t=&z=14&iwloc=B&output=embed" style=""></iframe>
  </div>
  <div class="container px-5 py-24 mx-auto flex">
    <div class="lg:w-1/3 md:w-1/2 bg-white rounded-lg p-8 flex flex-col md:ml-auto w-
full mt-10 md:mt-0 relative z-10 shadow-md">
      <h2 class="text-gray-900 text-lg mb-1 font-medium title-font">Feedback</h2>
      <p class="leading-relaxed mb-5 text-gray-600">Post-ironic portland shabby chic
echo park, banjo fashion axe</p>
      <div class="relative mb-4">
        <label for="email" class="leading-7 text-sm text-gray-600">Email</label>
        <input type="email" id="email" name="email" class="w-full bg-white rounded
border border-gray-300 focus:border-indigo-500 focus:ring-2 focus:ring-indigo-200 text-
```

```
base outline-none text-gray-700 py-1 px-3 leading-8 transition-colors duration-200
ease-in-out">
      </div>
      <div class="relative mb-4">
        <label for="message" class="leading-7 text-sm text-gray-600">Message</label>
        <textarea id="message" name="message" class="w-full bg-white rounded border
border-gray-300 focus:border-indigo-500 focus:ring-2 focus:ring-indigo-200 h-32 text-
base outline-none text-gray-700 py-1 px-3 resize-none leading-6 transition-colors
duration-200 ease-in-out"></textarea>
      </div>
      <button class="text-white bg-indigo-500 border-0 py-2 px-6 focus:outline-none
hover:bg-indigo-600 rounded text-lg">Button</button>
      <p class="text-xs text-gray-500 mt-3">Chicharrones blog helvetica normcore
iceland tousled brook viral artisan.</p>
    </div>
  </div>
</section>
```

- Después tendremos una sección de mapas
- Podemos copiar un iframe de una ubicación específica desde googlemaps, dándole a compartir
- Lo pego en el src del iframe
- Uso el #f1f1f1 para el color del background de la app en styles.css

```
@import "tailwindcss";

html, body{
    background-color: #f1f1f1;
}
```

- Es una manera de cargar un mapa sin código, y se carga de manera perezosa
- Si necesito una página de Login, ¿cómo hago eso? (para que no se vea la aplicación entera, con el navbar, etc)
- O argumentos de url, validadores de ruta...
-

Rutas hijas y padres

- Tenemos otro componente de Tailwind para hacer un login. Lo copio en modules/auth/pages/LoginPage.vue dentro de un template
- https://www.creative-tim.com/twcomponents/component/login-page-with-tailwind-css

```
<!-- component -->
<div class="bg-gray-100 flex justify-center items-center h-screen">
    <!-- Left: Image -->
<div class="w-1/2 h-screen hidden lg:block">
```

```
  <img src="https://placehold.co/800x/667fff/ffffff.png?
text=Your+Image&font=Montserrat" alt="Placeholder Image" class="object-cover w-full h-
full">
</div>
<!-- Right: Login Form -->
<div class="lg:p-36 md:p-52 sm:20 p-8 w-full lg:w-1/2">
  <h1 class="text-2xl font-semibold mb-4">Login</h1>
  <form action="#" method="POST">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Username</label>
      <input type="text" id="username" name="username" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Password</label>
      <input type="password" id="password" name="password" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!-- Remember Me Checkbox -->
    <div class="mb-4 flex items-center">
      <input type="checkbox" id="remember" name="remember" class="text-blue-500">
      <label for="remember" class="text-gray-600 ml-2">Remember Me</label>
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">Forgot Password?</a>
    </div>
    <!-- Login Button -->
    <button type="submit"
class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2 px-4 w-
full">Login</button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
    <a href="#" class="hover:underline">Sign up Here</a>
  </div>
</div>
</div>
```

- Con el router tal y como lo tenemos configurado solo tenemos rutas padres
- Se montan directamente en el root de la aplicación
- Tendrían hijas si tuvieran con la propiedad children (en el router)
- En el home sería útil si tuviera varias tabs
- Un uso muy común es para resolver el problema del Login
- Digamos que la barra de navegación, el header, el footer, lo quiero en todas mis páginas (pero en el login no)
- Creemos en modules/landing/layouts/LandingLayout.vue

- Copio todo lo que tengo en App.vue y lo pego en LandingLayout.vue
- En App.vue solo renderizo el RouterView

```
<template>
  <RouterView />
</template>

<script setup lang="ts">
</script>
```

- Todo lo que quiera que tenga el LandingLayout (todas las páginas menos el Login) deben ser rutas hijas del LandingLayout
- En la propiedad children, dentro del arreglo, pego todas las rutas que tenía
- Fuera del arreglo de children, en otro objeto, coloco la página de login
- router/index.ts

```
import HomePage from '@/modules/landing/pages/HomePage.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
   {
    path: '/',
    name: 'landing',
    component: ()=>import("@/modules/landing/layouts/LandingLayout.vue"),
    children:[
       {
      path: '/',
      name: 'home', //opcional
      component: HomePage
    },
    {
      path: '/features',
      name: 'features', //opcional
      component: ()=>import('@/modules/landing/pages/FeaturesPage.vue')
    },
    {
      path: '/pricing',
      name: 'pricing', //opcional
      component: ()=>import('@/modules/landing/pages/PricingPage.vue')
    },
    {
      path: '/contact',
      name: 'contact', //opcional
      component: ()=>import('@/modules/landing/pages/ContactPage.vue')
    },
    ]
   },
   {
```

```
    path:'/login',
    name: 'login',
    component: ()=> import('@/modules/auth/pages/LoginPage.vue')
   }
  ],
})

export default router
```

- De esta manera las páginas Pricing, Features, Contact y Home tienen el header y el footer, y la página Login no
- **NOTA:** Tenemos que manejar el 404 de una ruta mal especificada!
- Vamos a crear también un RegisterPage dónde el formulario que hay en el Login será el layout y al registerPage le vamos a agregar más cosas

Auth Layout

- Creo auth/pages/RegisterPage.vue
- Creo un auth/layouts/AuthLayout.vue donde colocaré el html que van a compartir estos dos componentes
- Copio todo lo que hay en el LoginPage (pues es la estructura) y lo pego en el AuthLayout
- Todo lo que hay dentro del div que corresponde al formulario, eso va en el LoginPage.vue
- Lo corto y lo pego en el LoginPage, y no me olvido de colocar el RouterView allí donde va todo el código de LoginPage y RegisterPage
- **No hace falta importar RouterView ni RouterLink porque está de forma global en mi app**

```
<template>
    <!-- component -->
<div class="bg-gray-100 flex justify-center items-center h-screen">
    <!-- Left: Image -->
<div class="w-1/2 h-screen hidden lg:block">
  <img src="https://placehold.co/800x/667fff/ffffff.png?
text=Your+Image&font=Montserrat" alt="Placeholder Image" class="object-cover w-full h-
full">
</div>
    <RouterView />
</div>
</template>

<script setup lang="ts">¡
</script>
```

- En el Login, dentro de la etiqueta template, pego lo que había en el div donde ahora está el RouterView
- Uso un RouterLink en lugar del anchor tag

- **NOTA:** es mejor usar el v-bind en el to de RouterLink y especificar el nombre, por si los paths cambian tener el nombre para redireccionar

```
<template>
  <div class="lg:p-36 md:p-52 sm:20 p-8 w-full lg:w-1/2">
  <h1 class="text-2xl font-semibold mb-4">Login</h1>
  <form action="#" method="POST">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Username</label>
      <input type="text" id="username" name="username" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Password</label>
      <input type="password" id="password" name="password" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!-- Remember Me Checkbox -->
    <div class="mb-4 flex items-center">
      <input type="checkbox" id="remember" name="remember" class="text-blue-500">
      <label for="remember" class="text-gray-600 ml-2">Remember Me</label>
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">Forgot Password?</a>
    </div>
    <!-- Login Button -->
    <button type="submit"
class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2 px-4 w-
full">Login</button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
   <RouterLink :to="{name: 'register'}" class="hover:underline">Sign up Here</
RouterLink>
  </div>
</div>
</template>
```

- Al RegisterPage le agregamos el campo name y cambiamos el botón de Login por el de Register
- También usamos un RouterLink para direccionar a LoginPage

```
<template>
  <div class="lg:p-36 md:p-52 sm:20 p-8 w-full lg:w-1/2">
  <h1 class="text-2xl font-semibold mb-4">Register</h1>
  <form action="#" method="POST">
    <!-- Username Input -->
```

```
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Username</label>
      <input type="text" id="username" name="username" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Password</label>
      <input type="password" id="password" name="password" class="w-full border border-
gray-300 rounded-md py-2 px-3 focus:outline-none focus:border-blue-500"
autocomplete="off">
    </div>
    <!--Name Input-->
    <div class="mb-4">
      <label for="name" class="block text-gray-600">Name</label>
      <input type="text" id="name" name="name" class="w-full border border-gray-300
rounded-md py-2 px-3 focus:outline-none focus:border-blue-500" autocomplete="off">
    </div>
    <!-- Remember Me Checkbox -->
    <div class="mb-4 flex items-center">
      <input type="checkbox" id="remember" name="remember" class="text-blue-500">
      <label for="remember" class="text-gray-600 ml-2">Remember Me</label>
    </div>

    <!-- Register Button -->
    <button type="submit"
class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2 px-4 w-
full">Register</button>
  </form>
  <!-- Sign in  Link -->
  <div class="mt-6 text-blue-500 text-center">
   <RouterLink :to="{name: 'login'}" class="hover:underline">Sign in Here</RouterLink>
  </div>
</div>
</template>
```

- El router queda así
- Puedo redirigir a login si alguien cae a /auth
- router/index.ts

```
import HomePage from '@/modules/landing/pages/HomePage.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
   {
    path: '/',
    name: 'landing',
    component: ()=>import("@/modules/landing/layouts/LandingLayout.vue"),
    children:[
       {
```

```
      path: '/',
      name: 'home', //opcional
      component: HomePage
    },
    {
      path: '/features',
      name: 'features', //opcional
      component: ()=>import('@/modules/landing/pages/FeaturesPage.vue')
    },
    {
      path: '/pricing',
      name: 'pricing', //opcional
      component: ()=>import('@/modules/landing/pages/PricingPage.vue')
    },
    {
      path: '/contact',
      name: 'contact', //opcional
      component: ()=>import('@/modules/landing/pages/ContactPage.vue')
    },
    ]
   },
  {
    path: '/auth',
    redirect:{name: 'login'}, //que /auth lleve al login
    name: 'authlayout',
    component: ()=>import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
       {
    path:'/login',
    name: 'login',
    component: ()=> import('@/modules/auth/pages/LoginPage.vue')
   },
   {
    path: '/register',
    name: 'register',
    component: ()=>import('@/modules/auth/pages/RegisterPage.vue')
   }
    ]
  }
  ],
})

export default router
```

- Si quieres que auth sea parte de la ruta **hay que uqitarle el / a login y register en el router**

```
  {
    path: '/auth',
    redirect:{name: 'login'}, //que /auth lleve al login
    name: 'authlayout',
    component: ()=>import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
       {
    path:'login',
```

```
    name: 'login',
    component: ()=> import('@/modules/auth/pages/LoginPage.vue')
   },
   {
    path: 'register',
    name: 'register',
    component: ()=>import('@/modules/auth/pages/RegisterPage.vue')
   }
    ]
  }
```

- Ahora apunta a auth/login y auth/register. Lo bueno de tener los links a través del v-bind con el nombre es que aunque haya cambios, los links permanecen funcionales
- Si alguien pone una ruta que no existe en mi app, la app no está devolviendo un 404. Vamos a cambiar eso

Página 404

- Cuando se introduce en la url una ruta que no existe podemos redireccionar a una ruta que ya existe o crear una página de 404
- Se usa pathMatch para indicar cualquier path que no exista
- Puedo usar redirect, pero usaremos una 404
- router/index.ts

```
import HomePage from '@/modules/landing/pages/HomePage.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
   {
    path: '/',
    name: 'landing',
    component: ()=>import("@/modules/landing/layouts/LandingLayout.vue"),
    children:[
       {
      path: '/',
      name: 'home', //opcional
      component: HomePage
    },
    {
      path: '/features',
      name: 'features', //opcional
      component: ()=>import('@/modules/landing/pages/FeaturesPage.vue')
    },
    {
      path: '/pricing',
      name: 'pricing', //opcional
      component: ()=>import('@/modules/landing/pages/PricingPage.vue')
```

```
    },
    {
      path: '/contact',
      name: 'contact', //opcional
      component: ()=>import('@/modules/landing/pages/ContactPage.vue')
    },
    ]
   },
  {
    path: '/auth',
    redirect:{name: 'login'}, //que /auth lleve al login
    name: 'authlayout',
    component: ()=>import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
       {
    path:'/login',
    name: 'login',
    component: ()=> import('@/modules/auth/pages/LoginPage.vue')
   },
   {
    path: '/register',
    name: 'register',
    component: ()=>import('@/modules/auth/pages/RegisterPage.vue')
   }
    ]
  },
  {
    path: '/:pathMatch(.*)*',
    //redirect: '/',
    name: 'notFound',
    component: ()=>import('@/modules/common/pages/404Page.vue')
  }
  ],
})

export default router
```

- El componente para modules/common/pages/404Page.vue lo he sacado de https://www.creative-tim.com/twcomponents/component/404-error-page

```
<template>
    <!-- component -->
<!-- This is an example component -->
<div class="h-screen w-screen bg-gray-50 flex items-center">
    <div class="container flex flex-col md:flex-row items-center justify-between px-5
text-gray-700">
            <div class="w-full lg:w-1/2 mx-8">
                <div class="text-7xl text-green-500 font-dark font-extrabold mb-8">
404</div>
            <p class="text-2xl md:text-3xl font-light leading-normal mb-8">
                Sorry we couldn't find the page you're looking for
            </p>
```

```
            <RouterLink :to="{name: 'home'}" class="px-5 inline py-3 text-sm font-
medium leading-5 shadow-2xl text-white transition-all duration-400 border border-
transparent rounded-lg focus:outline-none bg-green-600 active:bg-red-600 hover:bg-
red-700">back to homepage</RouterLink>
    </div>
        <div class="w-full lg:flex lg:justify-end lg:w-1/2 mx-5 my-12">
        <img src="https://user-images.githubusercontent.com/43953425/166269493-
acd08ccb-4df3-4474-95c7-ad1034d3c070.svg" class="" alt="Page not found">
        </div>

    </div>
</div>
</template>
```

- Si quiero manetener el footer y el menú de navegación, puedo hacer que este 404Page sea una ruta hija del landing
- **NOTA**: Si el componente a copiar está dentro de un body, copiar lo que hay dentro (sin la etiqueta body). Si es un div, meter dentro de un template y ya está
- Tenemos que ver como proteger rutas
- Añado el Login como submenú en el LandingLayout.vue
- Para ello meto los RouterLink que tenía en un div y en otro div el login
- El menú quedará en el extremo izquierd y el Login en el extremo derecho

```
<nav class="flex items-center space-x-4 w-full justify-between">
<!-- Contenedor de los enlaces principales a la izquierda -->
    <div class="flex space-x-4">
  <RouterLink :to="{name: 'home'}"> Home </RouterLink>
  <RouterLink :to="{name: 'features'}"> Features </RouterLink>
  <RouterLink :to="{name: 'pricing'}"> Pricing </RouterLink>
  <RouterLink :to="{name: 'contact'}"> Contact </RouterLink>
  </div>
<!-- Contenedor para Login, a la derecha -->
  <div class="flex flex-col items-center mt-2">
  <RouterLink :to="{name: 'login'}"> Login </RouterLink>
  </div>
</nav>
```

Argumentos por URL

- Creamos modules/pokemons/pages/PokemonPage.vue
- Para la imagen uso POSTMAN y hago un llamado a la PokeAPi en sprites/other/dream_world/front_default

```
<template>
    <section class="flex flex-col items-center justify-center">
        <h1>Pokemón <small class="text-blue-500">100</small></h1>
        <img src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/
```

```
pokemon/other/dream-world/1.svg"
        alt=""
        class="w-[200px] h-[200px] mt-5"
        >
    </section>
</template>
```

- Será una hija del Landing
- router/index.ts

```
//routes
{
    path: '/',
    name: 'landing',
    component: ()=>import("@/modules/landing/layouts/LandingLayout.vue"),
    children:[
       {
      path: '/',
      name: 'home', //opcional
      component: HomePage
    },
    {
      path: '/features',
      name: 'features', //opcional
      component: ()=>import('@/modules/landing/pages/FeaturesPage.vue')
    },
    {
      path: '/pricing',
      name: 'pricing', //opcional
      component: ()=>import('@/modules/landing/pages/PricingPage.vue')
    },
    {
      path: '/contact',
      name: 'contact', //opcional
      component: ()=>import('@/modules/landing/pages/ContactPage.vue')
    },
    {
      path: '/pokemon/:id',
      name: 'pokemons',
      component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
    }
    ]
   },

{...code}
```

- Agrego Pokemons al lado de Login en el nav LandingLayout
- Le pongo un 1 de id en duro
- Pongo un elemento al lado del otro con flex (sin flex-col), les pongo una separación con space-x-4

• LandingLayout.vue

```
<nav class="flex items-center space-x-4 w-full justify-between">
  <div class="flex space-x-4">
  <RouterLink :to="{name: 'home'}"> Home </RouterLink>
  <RouterLink :to="{name: 'features'}"> Features </RouterLink>
  <RouterLink :to="{name: 'pricing'}"> Pricing </RouterLink>
  <RouterLink :to="{name: 'contact'}"> Contact </RouterLink>
  </div>

  <div class="flex space-x-4 items-center mt-2">
  <RouterLink to="pokemon/1"> Pokemon </RouterLink>
  <RouterLink :to="{name: 'login'}"> Login </RouterLink>
  </div>
</nav>
```

• Hay varios lugares dónde voy a necesitar el id del pokemon

• ¿Cómo recibo esa property en PokemonPage que va a venir por el URL?

• Uso defineProps y el v-bind en el src de image, encierro entre back tics (dentro de las comilals) el string de la imagen y le coloco la prop

• Renderizo también el id en el small del h1

```
<template>
    <section class="flex flex-col items-center justify-center">
        <h1>Pokemón <small class="text-blue-500 text-lg">#{{ id }}</small></h1>
        <img
        :src="`https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/
pokemon/other/dream-world/${id}.svg`"
        alt=""
        class="w-[200px] h-[200px] mt-5"
        >
    </section>
</template>

<script setup lang="ts">
   interface Props{
    id: string //por el URL siempre recibimos strings
   }

   defineProps<Props>()
</script>
```

• Nos da error de “missing required prop”

• En el vue-router necesitamos poner una property que diga que necesitamos que lo convierta

• Se pueden hacer varias cosas, empecemos por poner props en true

```
{
  path: '/pokemon/:id',
```

```
  name: 'pokemon',
  props: true,
  component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
}
```

- Pongamos que quiero que el id sea un número para hacer cálculos con él
- Si quisiera crear un botón/link para el siguiente pokemon, bien podría usar en el to la ruta entre backticks (dentro de las comillas) tipo pokemon/{id}, pero **también pudo usar el nombre y pasarle el ide en el objeto params** así
- Recuerda que siempre que no sea pasarle solo un string usamos v-bind

```
<RouterLink
    :to="{name: 'pokemon', params: {id: id+1}}"
    class="bg-blue-500 text-white p-2 rounded mt-5 text-center"
>
    Siguiente
</RouterLink>
```

- Debo cambiar la interface de las props en PokemonPage.vue

```
<script setup lang="ts">
   interface Props{
    id: number
   }

   defineProps<Props>()
</script>
```

- Falta cambiar el string sacado de la url a número!

Procesar argumentos por URL

- En la ruta tenemos varios ciclos de vida, por ejemplo el **beforeEnter**
- Si a la propiedad props le añado una función de flecha donde voy a tener la route, si miro el tipo con el cursro encima me dice que es de tipo RouteLocationNormalizedGeneric
- Hago un console.log de ese route

```
{
  path: '/pokemon/:id',
  name: 'pokemon',
  props: (route)=>{
    console.log(route)
  },
```

```
  component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
}
```

- En consola veo un objeto como este

```
fullPath: "/pokemon/1"
hash: ""
href: "/pokemon/1"
matched: (2) [{…}, {…}] //rutas en las que hace match
meta: {}
name: "pokemon"
params: {id: '1'}
path: "/pokemon/1"
query: {}
redirectedFrom: undefined
[[Prototype]]: Object
```

- Lo que yo retorne en esta función es la prop que retorna

```
{
  path: '/pokemon/:id',
  name: 'pokemon',
  props: (route)=>{
    return {
      id:100 //retorna id 100
    }
  },
  component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
}
```

- Aquí es donde puedo extraer el id, hacer una validación de que lo que venga sea un número y retornar el id como número
- Si no es un número devuelvo 1, si no devuelvo el id (que he parseado con un + en route.params.id)

```
{
  path: '/pokemon/:id',
  name: 'pokemon',
  props: (route)=>{
    const id = +route.params.id!

    return isNaN(id) ? {id: 1}: {id}
  },
  component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
}
```

useRouterComposable

- Hay ocasiones en las que voya a querer navegar pero no con un botón/anchor tag/ RouterLink, sino basado en alguna acción de JS
- modules/auth/pages/LoginPage.vue
- Aquí tenemos un botón que hace el submit de Login de type submit
- Lo cambio a type button porque no quiero trabajar con la propagación del formulario, quiero que sea un botón normal y corriente
- Le paso la función onLogin (que creo en el script setup del componente) en el @click

```
<button
  type="button"
  class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2 px-4 w-
full"
  @click="onLogin"
>
  Login
</button>
```

- En el script setup del componente añado la navegación
- Tenemos el **useRoute** para poder obtener info sobre la ruta
- Y tenemos **useRouter** que nos permite obtener el objeto del router
- Con router. tengo varios métodos, el push es para cuando quiero que se cree un historial
- A replace le paso a que ruta quiero navegar
- LoginPage.vue

```
<script setup lang="ts">
import { useRouter } from 'vue-router';

  const router = useRouter()

  const onLogin=()=>{
    router.replace({name:'home'})
  }
</script>
```

- Vamos a almacenar un valor booleano ficticio en el localStorage
- Cuando le dé al botón de Login almacenará eel userId fake en el localStorage

```
<script setup lang="ts">
import { useRouter } from 'vue-router';

  const router = useRouter()
```

```
  const onLogin=()=>{
    localStorage.setItem('userId', 'ABC-123')
    router.replace({name:'home'})
  }
</script>
```

- Para verlo ir a la pestaña de Aplicación de la consola en el navegador, Almacenamiento local
- Se puede almacenar cualquier info siempre y cuando sean strings
- Si se quieren guardar onjetos hay que serializarlos con JSON.stringify
- Vamos a ver cómo solo puedo ver esta pantalla de pokemons si estoy autenticado
- Si no estoy autenticado no debería poder ni entrar en la pantalla de pokemons
- Para borrar la data en el LocalStorage seleccionar y pulsar delete
- En el LocalStorage la data es persistente, en la SessionStorage solo se mantendrá mientras la sesión del navegador esté abierta

Proteger rutas

- Los protectores de rutas, o **Guards**, se pueden colocar en varios lugares
- Se pueden tener tantos como se necesiten
- Todos se vana air ejecutando uno después del otro. Hasta que todos se cumplen, entonces llegamos a la ruta en particular
- Como va a ser un Guard estrechamente relacionado con auth, lo coloco en auth//guards/is-authenticated.guard.ts
- El guard recibe el to(a dónde voy), el from(de dónde vengo), y el next(la función que voy a llamar para navegar a la persona)
- Para tipar estos argumentos, si voy al router, la propiedad beforEnter tiene un array que va a esperar los guards
- Aquí puedo crear una función de flecha con el to, el from y el next, y de aquí puedo sacar el tipado ya que ya viene implícito, solo tengo que colocar el cursor encima de cada argumento

```
{
      path: '/pokemon/:id',
      name: 'pokemon',
      beforeEnter:[
        (to, from, next)=>{ //tipado implícito

          console.log(to,from,next)
          return next()
        }
      ],
      props: (route)=>{
        const id = +route.params.id!

        return isNaN(id) ? {id: 1}: {id}
```

```
      },
      component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
    }
```

- Ahora puedo tipar los argumentos

```
import type { NavigationGuardNext, RouteLocationNormalizedGeneric,
RouteLocationNormalizedLoadedGeneric } from "vue-router"


const isAuthenticatedGuard = (
    to:RouteLocationNormalizedGeneric,
    from:RouteLocationNormalizedLoadedGeneric,
    next:  NavigationGuardNext)=>{


    return next()
}

export default isAuthenticatedGuard
```

- Ahora puedo obtener el userId del LocalStorage
- Si no hay userId lo voy a mandar al login

```
import type { NavigationGuardNext, RouteLocationNormalizedGeneric,
RouteLocationNormalizedLoadedGeneric } from "vue-router"


const isAuthenticatedGuard = (
    to:RouteLocationNormalizedGeneric,
    from:RouteLocationNormalizedLoadedGeneric,
    next:  NavigationGuardNext)=>{

    const userId = localStorage.getItem('userId')

    if(!userId){
        return next({
            name: 'login'
        })
    }

    return next()
}

export default isAuthenticatedGuard
```

- Si intento navegar a la pantalla de pokemons y no tengo el userId en el localStorage me manda directamente a la pantalla de Login

- Si no está autenticado podemos determinar la ruta en la que nos encontramos (pongamos que estoy en la página pokemon)
- En el to, tenemos el path (to.path)
- Esta es la ruta a la que quería navegar. Puedo grabarlo en el LocalStorage
- Siempre se va a guardar la última ruta en la que estuvo (pokemon/1, luego pokemon/2 si le da al botón de siguiente,etc)

```
import type { NavigationGuardNext, RouteLocationNormalizedGeneric,
RouteLocationNormalizedLoadedGeneric } from "vue-router"


const isAuthenticatedGuard = (
    to:RouteLocationNormalizedGeneric,
    from:RouteLocationNormalizedLoadedGeneric,
    next:  NavigationGuardNext)=>{

    const userId = localStorage.getItem('userId')
    localStorage.setItem('lastPath', to.path)

    if(!userId){
        return next({
            name: 'login'
        })
    }
    return next()
}

export default isAuthenticatedGuard
```

- Si la persona no está autenticada, saco a la persona de ahí (la mando al login), pero sé a qué página iba la persona, la tengo en el localStorage, para que cuando se autentique, la lleve ahí
- En el LoginPage, en lugar de mandarlo al home con el onLogin (la función del botón de Login), vamos a usar la página del localStorage
- Si el valor es nulo lo mandamos al home

```
<script setup lang="ts">
import { useRouter } from 'vue-router';

  const router = useRouter()

  const onLogin=()=>{
    localStorage.setItem('userId', 'ABC-123')
    const lastPath= localStorage.getItem('lastPath') ?? '/'
    router.replace(lastPath)
```

```
  }
</script>
```

- Para que se guarde en el localStorage el lastPath (la página donde stoy de pokemon) debo recargar el navegador estando en la página
- Recuerda que tenemos el guard en el beforeEnter del router
- router/index.ts

```
{
  path: '/pokemon/:id',
  name: 'pokemon',
  beforeEnter:[isAuthenticatedGuard],
  props: (route)=>{
    const id = +route.params.id!

    return isNaN(id) ? {id: 1}: {id}
  },
  component: ()=> import('@/modules/pokemons/pages/PokemonPage.vue')
}
```

- Ahora si borro el userId del localStorage y vuelvo a hacer login, me lleva a la página dónde estaba (pokemon/10, pokemon/11, la que sea)
- Si no hay una página guardada en lastPath en el localStorage me lleva al home cuando hago Login
- En la vida real vamos a usar el is-authenticated.guard para verificar un jsonwebtoken, o el estado de autenticación, puede ser async (para cuando tengamos un backend con el que autenticarme)
- Lo importante de esta lección es que hemos visto **cómo proteger la ruta** (el backend tendría otra capa)

Ciclo de vida de los componentes

- Vamos a HomePage.vue
- El ciclo de vida no son más que una serie de funciones que se disparan automáticamente en un cierto punto del tiempo
- En la documentación de Vue veremos los Lifecycle Hooks del Composition API
- Es similar a React: onMounted, onUpdated, onUnmounted, onBeforeMount, onBeforeUpdate,onBeforeUnmount, onErrorCaptured, onRenderTracked, onRenderTriggered, onActivated, onDeactivated, onServerPrefetch (este último es para Server-Side)
- Se pueden usar directamente en el script setup

```
<template>
    <div class="text-center">
```

```
    <h1 class="text-4xl font-bold tracking-tighter sm:text-5xl">
        Bienvenido a nuestro sitio web
    </h1>
    <p class="mx-auto max-w-[600px] text-gray-500 md:text-xl">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    </p>
    </div>
</template>

<script setup lang="ts">
import { onMounted } from 'vue';

onMounted(()=>{
    console.log('onMounted')
})
</script>
```

- Vamos a usarlos todos
- Cuando tenemos más de diez líneas en el script setup es recomendable usar un archivo independiente
- En modules/landing/pages/HomePage.ts
- La función setup también es una parte del ciclo de vida de lso componentes, ya que se ejecuta antes de que se monte

```
import { defineComponent, onMounted } from "vue";

export default defineComponent({
    setup: ()=>{

        onMounted(()=>{
            console.log("onMounted")
        })

        console.log('setup') //el setup aparece primero porque setup se ejecuta antes
que todo

    }
})
```

- Debo hacer referencia en el script del HomePage al archivo HomePage.ts y quitar el setup

```
<template>
    <div class="text-center">
    <h1 class="text-4xl font-bold tracking-tighter sm:text-5xl">
        Bienvenido a nuestro sitio web
    </h1>
    <p class="mx-auto max-w-[600px] text-gray-500 md:text-xl">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    </p>
    </div>
```

```
</template>

<script lang="ts" src="./HomePage">
</script>
```

• Llamemos a todos los métodos

```
import {
    defineComponent,
    onActivated,
    onBeforeMount,
    onBeforeUnmount,
    onBeforeUpdate,
    onDeactivated,
    onErrorCaptured,
    onMounted,
    onRenderTracked,
    onRenderTriggered,
    onUnmounted,
    onUpdated,
    ref } from "vue";

export default defineComponent({
    setup: ()=>{
      //defino propiedad reactiva para el siguiente ejercicio
      const count = ref(0)

        onMounted(()=>{ console.log("onMounted")})
        onUpdated(()=>{console.log("onUpdated")})
        onUnmounted(()=>{console.log("onUnmounted")})
        onBeforeMount(()=>{console.log("onBeforeMount")})
        onBeforeUpdate(()=>{console.log("onBeforeUpdate")})
        onBeforeUnmount(()=>{console.log("onBeforeUnmount")})
        onErrorCaptured(()=>{console.log("onErrorCaptured")})
        onRenderTracked(()=>{console.log("onrenderTracked")})
        onRenderTriggered(()=>{console.log("onRenderTriggered")})
        onActivated(()=>{console.log("onActivated")})
        onDeactivated(()=>{console.log("onDeactivated")})
        console.log('setup')

      return{
        count
      }
    }
})
```

• En consola aparece el setup, el onBeforeMount y el onMounted

• Si me voy a otra pantalla dispara el onBeforeUnmount y el onUnmounted

• Esto es útil para limpiar subscripciones, websockets, etc

keep Alive - Activar y desactivar

- Renderizo el count en el template del Home
- Añado un botón de incremento

```
<template>
    <div class="text-center">
    <h1 class="text-4xl font-bold tracking-tighter sm:text-5xl">
        Bienvenido a nuestro sitio web
    </h1>
    <p>Counter: {{ count }} </p>
    <button @click="count++">Increment</button>
    </div>
</template>

<script lang="ts" src="./HomePage">
</script>
```

- Ahora aparece en consola (en este orden) el setup, onBeforeMount, onRenderTracked, onMounted
- onrenderTracked es cuando hay alguna dependencia que se acaba de colocar y nos permite poderlo seguir (solo en modo desarrollo)
- Cuando le doy al botón de Increment se dispara onRenderTriggered, onBeforeUpdate, onRenderTracked, onUpdate
- onRenderTrigged nos indica cuando empieza a realizar el proceso, cuando hay que empezar a hacer la renderización de un nuevo cambio (solo en dev)
- Luego se detcta el cambio antes de que se haga (onBeforeUpdate)
- Se dispara otro onRenderTracked y luego en onUpdated que se dispara después de que se actualiza
- Si salgo del Home tengo el onBeforeUnmount y el onUnmounted
- Si regreso al Home el counter vuelve a cero y se disparan los que se dispararon inicialmente (setup, onBeforeMount, onrenderTracked, onMounted)
- Si quiero manetener el valor de 3 **puedo usar un gestor de estado**
- Pongamos que no quiero usar uno usar un gestor de estado (o no puedo porque es de un acordeón, o unas tabs)
- Hay un componente en el vue-router que es el **KeepAlive** (que se puede mezclar con el Transition para hacer transiciones)
- Se usa para mantener con vida al componente, usando el componente keep-alive
- Lo uso en el router-view del LandingLayout

```
<main class="flex-1 flex items-center justify-center">
  <RouterView v-slot="{Component}">
    <keep-alive>
```

```
    <component :is="Component" />
    </keep-alive>
  </RouterView>
</main>
```

- Ahora en consola pareció también el onActivated
- Si incremento el contador, me voy a otra página y vuelvo al Home, el valor del contador se mantuvo
- Si estoy en el Home, el active está en el Home, si estoy en Features, el active está en Features
- Si recargo el navegador pierdo el valor del counter
- Cuando me muevo a otra ruta (dentro del keep-alive) no voy a llamar al onMounted, porque lo que estamos haciendo es desactivarlo, por lo que se llama al onDeactivated

Router Link Active

- Estaría bien remarcar dónde está el usuario en el menú de navegación
- Es sencillo. Puedo inspeccionar el elemento en consola dando click al primer icono de la esquina superior izquierda (que es de una flechita en una pantalla de linea discontinua) y después sobre el elemento en cuestión
- Cuando estoy en una ruta, vue-router va a poner en la clase del RouterLink donde estoy el router-link-active y el router-link-exact-active
- El router-link-active menciona que estamos en esa ruta, el router-link-exact-active dice que estamos exactamente en el path que coincide con lo que en el elemento inspeccionado es el href del anchor tag (‘/features’, por ejemplo)
- Empieza con el /. La ruta del Home también empieza con el /, por lo que también tiene la clase router-link-active, ya que esas dos rutas podrían coincidir con el path en el cual nos encontramos (‘/features’)
- Podemos usar estas clases para colorear el elemento
- Uso active-class, pero en el Home no lo coloco por lo que hemos comentado, si no siempre se marcaría el Home (al tener /) y donde sea que esté (Features, Pricing)
- Landinglayout.vue

```
<div class="flex space-x-4">
  <RouterLink :to="{name: 'home'}"> Home </RouterLink>
  <RouterLink
  active-class="underline font-semibold"
  :to="{name: 'features'}"> Features </RouterLink>
  <RouterLink :to="{name: 'pricing'}"
  active-class="underline font-semibold"
  > Pricing </RouterLink>
  <RouterLink :to="{name: 'contact'}"
  active-class="underline font-semibold"
```

```
  > Contact </RouterLink>
</div>
```

- Para evitar este error usaremos exact-active-class
- Puedo dejar el nav como estaba
- Pero en lugar de hacerlo a nivel de componente (que no está mal) puedo definirme una clase en styles.css que apunte a .router-link-exact-active
- styles.css

```
@import "tailwindcss";

html, body{
    background-color: #f1f1f1;
}

.router-link-exact-active{
    @apply font-semibold transition-all text-green-500
}
```

- Ha habido una cosa que ha quedado pendiente, que si yo clico dos veces en el menú a pokemon, como tengo en duro el pokemon/1, me añade al url otro pokemon/1 por lo que me lleva a la página de 404
- Una manera de solucionarlo sería con un v-if

```
<RouterLink :to="{ name: 'pokemon', params: { id: 1 } }"
> Pokemon
</RouterLink>
```

La variable global $route en Vue

¡Claro! Aquí tienes una lista completa de las propiedades de `$route` en Vue.js, con una breve descripción de cada una:

Propiedades de `$route`

1. `$route.path`:

- **Descripción**: La **ruta actual** (el valor completo del `path` en la URL, por ejemplo, `/pokemon/1`).

1. `$route.name`:

- **Descripción**: El **nombre de la ruta** definido en Vue Router. Es útil para referirse a rutas por su nombre en lugar de la ruta completa.

1. `$route.params`:

- **Descripción**: Un objeto que contiene los **parámetros dinámicos** de la ruta. Estos son los valores definidos en la URL con `:param` (por ejemplo, `/pokemon/:id`).

1. `$route.query`:

- **Descripción**: Un objeto que contiene los **parámetros de consulta** de la URL (lo que sigue al `?` en la URL, como `/search?query=vue`).

1. `$route.hash`:

- **Descripción**: El **fragmento de la URL** después del `#` (por ejemplo, en `/about#section`, la propiedad `hash` será `#section`).

1. `$route.fullPath`:

- **Descripción**: La **ruta completa** incluyendo la **cadena de consulta** y el **hash**. Esto es, por ejemplo, `/pokemon/1?search=true#info`.

1. `$route.matched`:

- **Descripción**: Un **array de rutas coincidentes** con la ruta actual. Es útil para obtener información sobre las rutas anidadas y los componentes coincidentes.

1. `$route.redirectedFrom`:

- **Descripción**: La **ruta de la cual** la navegación fue **redirigida** (si la ruta actual fue el resultado de una redirección). Si no hubo redirección, será `null`.

1. `$route.meta`:

- **Descripción**: Un objeto que contiene los **metadatos** definidos para la ruta en la configuración del enrutador. Este valor lo puedes personalizar para agregar información adicional a la ruta (como permisos, roles, etc.).

1. `$route.hash`:
    - **Descripción**: Parte de la URL después del `#` que puede ser utilizada para manejar fragmentos dentro de la página.

Resumen:

- `$route.path`: La ruta completa sin parámetros dinámicos.
- `$route.name`: El nombre de la ruta.
- `$route.params`: Parámetros dinámicos en la ruta.
- `$route.query`: Parámetros de consulta de la URL.
- `$route.hash`: Fragmento de la URL después del `#`.
- `$route.fullPath`: La ruta completa con parámetros de consulta y hash.
- `$route.matched`: Rutas coincidentes en la navegación.
- `$route.redirectedFrom`: La ruta original si fue redirigida.
- `$route.meta`: Metadatos asociados con la ruta (personalizable).

- Estas propiedades proporcionan acceso a diferentes partes de la URL y la ruta actual, lo que te permite hacer una navegación dinámica y gestionar parámetros en tu aplicación Vue.

Otras variables globales en Vue

Claro, aquí tienes un listado de las variables globales disponibles en **Vue.js** y **Vue Router**, con una breve explicación de cada una:

Variables globales en Vue

1. `$el`:

- **Descripción**: Hace referencia al **elemento DOM** en el que está montado el componente.
- **Ejemplo**: `this.$el` te da acceso al elemento DOM que contiene tu componente.

```
javascript
  mounted() {
```

```
    console.log(this.$el); // <div>...</div>
  }
```

1. **`$data`**:

- **Descripción**: Es un objeto que contiene los datos reactivos del componente. Es equivalente a la propiedad `data` del componente.
- **Ejemplo**: `this.$data` devuelve todos los datos del componente.

```
javascript
  mounted() {
    console.log(this.$data); // { count: 0 }
  }
```

1. **`$props`**:

- **Descripción**: Es un objeto que contiene todas las propiedades (`props`) pasadas al componente.
- **Ejemplo**: `this.$props` contiene los props que el componente ha recibido desde su componente padre.

```
javascript
  mounted() {
    console.log(this.$props); // { title: 'Hello' }
  }
```

1. **`$route`** (disponible solo con Vue Router):

- **Descripción**: Proporciona información sobre la ruta actual, como el `path`, los `params`, el `query`, etc. Es útil para obtener información sobre la URL y los parámetros de la ruta.
- **Ejemplo**:

```
javascript
  mounted() {
    console.log(this.$route.path);  // '/home'
    console.log(this.$route.params);  // { id: 1 }
  }
```

1. **`$router`** (disponible solo con Vue Router):

- **Descripción**: Es el objeto del enrutador de Vue Router. Permite realizar operaciones de navegación, como redirigir a otras rutas.
- **Ejemplo**:

```javascript
  methods: {
    navigateToHome() {
      this.$router.push({ name: 'home' });
    }
  }
```

1. **`$store`** (disponible solo si usas Vuex):

- **Descripción**: Es el objeto que representa el estado global de la aplicación cuando estás usando Vuex. Permite acceder y modificar el estado global.
- **Ejemplo**:

```javascript
  mounted() {
    console.log(this.$store.state); // Accede al estado global
  }
```

1. **`$refs`**:

- **Descripción**: Es un objeto que contiene todas las referencias a los elementos DOM o componentes hijos que han sido asignados con `ref` en la plantilla.
- **Ejemplo**:

```html

<button ref="myButton">Click me</button>

```

1. **`$nextTick`**:

- **Descripción**: Permite ejecutar una función después de que Vue haya actualizado el DOM. Es útil cuando necesitas hacer algo justo después de un cambio en el DOM.
- **Ejemplo**:

```javascript
  this.$nextTick(() => {
```

```
    console.log('DOM actualizado');
  });
```

1. **`$root`**:

- **Descripción**: Hace referencia al **componente raíz** de la aplicación. Puede usarse para acceder a propiedades o métodos del componente principal desde cualquier lugar de la aplicación.
- **Ejemplo**:

```
javascript
  mounted() {
    console.log(this.$root); // Accede al componente raíz
  }
```

1. **`$parent`**:

    - **Descripción**: Hace referencia al **componente padre** de un componente hijo. Es útil si necesitas acceder a datos o métodos del componente que te contiene.
    - **Ejemplo**:

```
javascript
mounted() {
  console.log(this.$parent); // Accede al componente padre
}
```

2. **`$children`**:

    - **Descripción**: Es un array de los componentes hijos directos de un componente. Te permite acceder a los componentes hijos de manera programática.
    - **Ejemplo**:

```
javascript
mounted() {
  console.log(this.$children); // Accede a los componentes hijos
}
```

3. **`$emit`**:

    - **Descripción**: Permite **enviar eventos** desde un componente hijo hacia su componente padre. Este método es utilizado para comunicar eventos personalizados hacia el componente padre.
    - **Ejemplo**:

```
javascript
this.$emit('customEvent', data);
```

4. **`$destroy`**:
    - **Descripción**: Elimina el componente actual, es decir, destruye el ciclo de vida de un componente. Es más comúnmente utilizado cuando se maneja la destrucción manual de un componente.
    - **Ejemplo**:

```javascript
this.$destroy();
```

5. **`$isServer`**:
    - **Descripción**: Es una propiedad que indica si el código se está ejecutando en el **lado del servidor** (cuando se usa **SSR** o **Server-Side Rendering**) o no.
    - **Ejemplo**:

```javascript
if (this.$isServer) {
  console.log('Estamos en el servidor');
} else {
  console.log('Estamos en el cliente');
}
```

6. **`$set`** (disponible en Vue 2.x, en Vue 3 se usa `reactive`):
    - **Descripción**: Permite establecer una propiedad reactiva en un objeto que previamente no existía. En Vue 2, esto era necesario para hacer que las propiedades reactivas se "agregaran" dinámicamente a los objetos.
    - **Ejemplo**:

```javascript
this.$set(this.someObject, 'newProp', value);
```

7. **`$delete`** (disponible en Vue 2.x, en Vue 3 se usa `reactive`):
    - **Descripción**: Permite eliminar propiedades reactivas de un objeto.
    - **Ejemplo**:

```javascript
this.$delete(this.someObject, 'propertyToRemove');
```

Variables adicionales cuando usas Vue Router:

1. `$route.params`:

- **Descripción**: Proporciona los parámetros de la ruta actual. Se utiliza principalmente con rutas dinámicas, como `/pokemon/:id`.

```javascript
  console.log(this.$route.params.id); // Ejemplo: 1
```

1. `$route.query`:

- **Descripción**: Proporciona los parámetros de consulta de la URL (todo lo que viene después del `?`, por ejemplo, `/search?query=vue`).

```javascript
  console.log(this.$route.query.query); // Ejemplo: 'vue'
```

1. `$route.hash`:

- **Descripción**: Contiene la parte del **fragmento** de la URL, es decir, todo lo que sigue al `#` en una URL.

```javascript
  console.log(this.$route.hash); // Ejemplo: '#about'
```

Resumen

Aquí tienes una lista de las variables globales más comunes en Vue:

- `$route`: Información sobre la ruta actual.
- `$router`: Permite interactuar con el enrutador, como redirigir a otras rutas.
- `$store`: El estado global de la aplicación usando Vuex.
- `$refs`: Acceso a los elementos DOM o componentes hijos referenciados con `ref`.
- `$nextTick`: Permite ejecutar una función después de que Vue actualice el DOM.
- `$el`, `$data`, `$props`: Proporcionan acceso al elemento DOM, datos reactivos y propiedades del componente, respectivamente.
- `$root`, `$parent`, `$children`: Para navegar entre componentes y acceder a componentes relacionados.

Estas variables globales te permiten interactuar con diferentes partes de tu aplicación Vue, desde el enrutamiento hasta el estado global y el DOM.

Vue Herrera - Slots y DaisyUI

- Nos vamos a enfocar en trabajar con los slotes y DaisyUI, aunque Daisy no sea la mejor tecnología
- Un slot nos permite desde un componente padre mandar componentes hijos dentro de la definición del componente padre
- Esto ayuda a crear componentes reutilizables
- Este módulo nos encamina para cuando lleguemos a trabajar con Pinia, poder explicarlo basado en esta aplicación
- Trabajaremos con modales
- Es un app de proyectos

ProjectsApp

- Creamos un proyecto con npm create vue@latest
- name: projects-app
- TypeScript: si
- JSX: no
- Vue Router: si
- Pinia: si
- Vitest: si
- E2E: no
- ESLint: si
- Prettier: si
- Instalo Tailwind
- En App.vue borro todo lo que hay y pego únicamente el RouterView

```
<template>
  <RouterView />
</template>
```

- Puedo colocar el semicollon en true y el trailingComa en all en el .prettierc.json, simplemente por el estilo. ASí cuando guarde lo formateará siempre igual

```
{
  "$schema": "https://json.schemastore.org/prettierrc",
  "semi": true,
  "singleQuote": true,
```

```
  "printWidth": 100,
  "trailingComma": "all"
}
```

- La estructura de directorios será
- modules
- modules/common
- modules/common/components/ (donde irá el navbar, sidebar, tendré todo lo que s compartido que no tiene una relación directa con los otros módulos)
- modules/projects (agregar proyectos, tareas, etc)
- modules/projects/layouts
- modules/projects/components
- modules/projects/views
- No hay carpeta de composables, porque el ejercicio está enfocado a Pinia y como maneja la data
    - Pinia nos permite agregar lógica en las acciones
- Con esta estructura podemos empezar, se irá ampliando a medida que avancemos

Librería de componentes DaisyUI

- Instalemos esta librería de componentes para Tailwind (se necesita tener Tailwind previamente instalado)

> npm i -D daisyui@latest

- Hay que añadir esta linea a styles.css (que está importado en el main)

```
@import "tailwindcss";
@plugin "daisyui";
```

- Para probar que la configuración ha ido bien, puedo copiar todos los botones de una muestra desde la web

> https://daisyui.com/components/button/

- En App.vue los renderizo

```
<template>
  <button class="btn btn-neutral">Neutral</button>
  <button class="btn btn-primary">Primary</button>
  <button class="btn btn-secondary">Secondary</button>
  <button class="btn btn-accent">Accent</button>
  <button class="btn btn-info">Info</button>
```

```
  <button class="btn btn-success">Success</button>
  <button class="btn btn-warning">Warning</button>
  <button class="btn btn-error">Error</button>
  <RouterView />
</template>
```

- Fácilmente se pueden crear, poniendo solo btn- (y Ctrl+space para ver las opciones)

Estructura de los componentes

- Vamos a ir a componentes (de la web de DaisyUI) constantemente

> https://daisyui.com/components/

- En modules/projects/layouts/ProjectsLayout.vue

```
<template>
  <div class="flex flex-col ">
    <!-- Top Menu-->
    <main>
        <!--Side Menu-->

        <RouterView />
    </main>
  </div>
</template>
```

- Vamos con el Top Menu, como es un elemento común que quiero reutilizar en toda mi aplicación lo colocaremos en modules/common/components

```
<template>
    <div class="flex items-center justify-between p-4 bg-base-200">
        <h2 class="text-lg font-bold text-blue-600">Notion</h2>
    </div>
</template>
```

- Lo renderizo en ProjectsLayout (donde estaba el comment de Top Menu)
- Puedo usar TopMenu o top-menu
- No vemos nada en pantalla porque hay que redireccionar el Landing en el Layout
- Podemos cargarlo de manera perezosa o si va a ser algo que siempre vamos a necesitar simplemente usar el componente
- router/index.ts

```
import ProjectsLayout from '@/modules/projects/layouts/ProjectsLayout.vue'
import { createRouter, createWebHistory } from 'vue-router'
```

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'projects',
      component: ProjectsLayout
    }
  ],
})

export default router
```

• Toca el SideMenu. Lo creo en modules/projects/components/SideMenu.vue

```
<template>
    <aside class="bg-base-200 w-72 min-h-screen">
        <h2 class="text-lg font-bold mx-4">Proyectos</h2>
        <p class="text-sm text-gray-500 mx-4">No hay proyectos</p>
    </aside>
</template>
```

• Lo coloco dentro del main del Layout (donde estaba el comment SideMenu)

• Se recomienda usar como una etiqueta semántica html (top-menu) antes que TopMenu

```
<template>
  <div class="flex flex-col ">
    <top-menu />
    <main>
        <side-menu />

        <router-view />
    </main>
  </div>
</template>

<script setup lang="ts">
import TopMenu from '@/modules/common/components/TopMenu.vue';
import SideMenu from '../components/SideMenu.vue';

</script>
```

• Usemos un menú llamado Collapsable submenu, lo pego en SideMenu

```
<template>
  <aside class="bg-base-200 w-72 min-h-screen">
    <h2 class="text-lg font-bold mx-4">Proyectos</h2>
    <p class="text-sm text-gray-500 mx-4">No hay proyectos</p>
```

```
    <!--Menu-->
     <ul class="menu rounded-box w-56">
        <li><a>Item 1</a></li>
        <li>
            <details open>
            <summary>Parent</summary>
            <ul>
                <li><a>Submenu 1</a></li>
                <li><a>Submenu 2</a></li>
                <li>
                <details open>
                    <summary>Parent</summary>
                    <ul>
                    <li><a>Submenu 1</a></li>
                    <li><a>Submenu 2</a></li>
                    </ul>
                </details>
                </li>
            </ul>
            </details>
        </li>
        <li><a>Item 3</a></li>
        </ul>
    </aside>
</template>
```

- DaisyUI hace que funcione sin JS
- Los elementos como summary que no existen, son como si fueran divs, solo que DaisyUI ya sabe que estilos aplicarle
- No hay que importarlos

Estructura de projects view

- Trabajemos con la pantalla que se verá en el centro que es un listado de todos nuestros proyectos
- Buscamos la table with a row that highlights on hover
- modules/projects/views/ProjectsView.vue
- Los views se incrustan dentro de los layouts
- Le coloco un w-full al div para que me ocupe toda la pantalla de ancho
- Solo dejo una fila, usaremos un v-for para listarlas
- ProjectsView.vue

```
<template>
    <div class="overflow-x-auto w-full">
  <table class="table">
    <!-- head -->
    <thead>
      <tr>
```

```
        <th></th>
        <th>Proyectos</th>
        <th>Tareas</th>
        <th>Avances</th>
      </tr>
    </thead>
    <tbody>
      <!-- row 1 -->
      <tr>
        <th>1</th>
        <td>Cy Ganderton</td>
        <td>Quality Control Specialist</td>
        <td>Blue</td>
      </tr>
    </tbody>
  </table>
</div>
</template>
```

- Será una ruta hija en el router
- router/index.ts

```
import ProjectsLayout from '@/modules/projects/layouts/ProjectsLayout.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      redirect: {name: 'projects'},
      component: ProjectsLayout,
      children:[
        {
          path:'projects',
          name: 'projects',
          component: ()=>import('@/modules/projects/views/ProjectsView.vue')
        }
      ]
    }
  ],
})

export default router
```

- Aparece la tabla abajo porque me faltan unas clases de tailwind en el main del Layout (flex flex-row)
- ProjectsLayout.vue

```
<template>
  <div class="flex flex-col">
    <top-menu />
    <main class="flex flex-row">
        <side-menu />

        <RouterView />
    </main>
  </div>
</template>

<script setup lang="ts">
import TopMenu from '@/modules/common/components/TopMenu.vue';
import SideMenu from '../components/SideMenu.vue';

</script>
```

- Voy a querer un botón flotante. Busco circle button. Lo coloco abajo de la tabla
- Añado la clase btn-secondary
- Busco en svgrepo.com otro svg, Add Circle SVG Vector (se puede editar el svg en la página, de edit svg puedo hacer un copy)
- Le coloco la clase fixed y uso bottom-10 right-10 para situarlo en la esquina inferior derecha

```
<template>
<div class="overflow-x-auto w-full">
  <table class="table">
    <!-- head -->
    <thead>
      <tr>
        <th></th>
        <th>Proyectos</th>
        <th>Tareas</th>
        <th>Avances</th>
      </tr>
    </thead>
    <tbody>
      <!-- row 1 -->
      <tr>
        <th>1</th>
        <td>Cy Ganderton</td>
        <td>Quality Control Specialist</td>
        <td>Blue</td>
      </tr>
    </tbody>
  </table>
  <button class="btn btn-circle btn-secondary fixed bottom-10 right-10">
    <svg viewBox="0 0 24 24"
    fill="none"
    xmlns="http://www.w3.org/2000/svg">
    <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
    <g id="SVGRepo_tracerCarrier"
    stroke-linecap="round"
```

```
    stroke-linejoin="round"></g>
    <g id="SVGRepo_iconCarrier">
        <path d="M15 12L12 12M12 12L9 12M12 12L12 9M12 12L12 15"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
        <path d="M7 3.33782C8.47087 2.48697 10.1786 2 12 2C17.5228 2 22 6.47715 22
12C22 17.5228 17.5228 22 12 22C6.47715 22 2 17.5228 2 12C2 10.1786 2.48697 8.47087
3.33782 7"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
    </g>
    </svg>
</button>
</div>
</template>
```

• Ahora si, vamos con el código para que funcione

Botón flotante con Slots

• Separemos el código del botón flotante en un componente independiente

• modules/common/components/FabButton.vue

```
<template>
      <button class="btn btn-circle btn-secondary fixed bottom-10 right-10">
    <svg viewBox="0 0 24 24"
    fill="none"
    xmlns="http://www.w3.org/2000/svg">
    <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
    <g id="SVGRepo_tracerCarrier"
    stroke-linecap="round"
    stroke-linejoin="round"></g>
    <g id="SVGRepo_iconCarrier">
        <path d="M15 12L12 12M12 12L9 12M12 12L12 9M12 12L12 15"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
        <path d="M7 3.33782C8.47087 2.48697 10.1786 2 12 2C17.5228 2 22 6.47715 22
12C22 17.5228 17.5228 22 12 22C6.47715 22 2 17.5228 2 12C2 10.1786 2.48697 8.47087
3.33782 7"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
    </g>
    </svg>
</button>
</template>
```

• Lo renderizo en ProjectsView, debajo de la tabla, (después del div)

- El FabButton va a requerir cierta personalización, la emisión de los eventos, la posición quiero darla basada en unas properties
- Definamos el script setup de FabButton
- Defino la interfaz de las props. Si quiero un valor por defecto uso la función with defaults, le paso el defineProps tipado y en un objeto seteo la propiedad por defecto
- Como le voy a poner una propiedad por defecto marco como opcional position en la interfaz
- Creo las clases dentro de la etiqueta style con la palabra scoped para que solo afecte a este componente
- Para usar las clases de manera condicional uso v-bind con class y encierro las clases dentro de corchetes
- Fuera de las clases por defecto le paso la prop

```
<template>
      <button :class="['btn btn-circle btn-secondary fixed', position]">
    <svg viewBox="0 0 24 24"
    fill="none"
    xmlns="http://www.w3.org/2000/svg">
    <g id="SVGRepo_bgCarrier" stroke-width="0"></g>
    <g id="SVGRepo_tracerCarrier"
    stroke-linecap="round"
    stroke-linejoin="round"></g>
    <g id="SVGRepo_iconCarrier">
        <path d="M15 12L12 12M12 12L9 12M12 12L12 9M12 12L12 15"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
        <path d="M7 3.33782C8.47087 2.48697 10.1786 2 12 2C17.5228 2 22 6.47715 22
12C22 17.5228 17.5228 22 12 22C6.47715 22 2 17.5228 2 12C2 10.1786 2.48697 8.47087
3.33782 7"
        stroke="#1C274C"
        stroke-width="1.5"
        stroke-linecap="round"></path>
    </g>
    </svg>
</button>
</template>

<script setup lang="ts">

interface Props{
    position?: 'top-left' | 'top-right' | 'bottom-left' | 'bottom-right'
}

withDefaults(defineProps<Props>(), {
    position: 'bottom-right'
})
</script>

<style scoped>
@reference 'tailwindcss'
```

```
.top-left{
    @apply top-10 left-10;
}
.top-right{
    @apply top-10 right-10;
}
.bottom-left{
    @apply bottom-10 left-10;
}
.bottom-right{
    @apply bottom-10 right-10;
}

</style>
```

- El componente no me pide la property, porque la he marcado como opcional y he setado bottom-right por defecto con withDefaults - - Puedo cambiar la posición para ver si funciona

```
<fab-button position="bottom-left"/>
```

- Si quiero personalizar el icono (para hacerlo más reutilizable) ¿cómo lo hago?
- Cómo recibo el children del componente? es decir, cómo recibo este Hola mundo? (el children)

```
<fab-button>
  <span>Hola mundo</span>
</fab-button>
```

- Usando el componente global slot
- Borro el svg de FabButton y coloco el slot
- Se puede usar con autocierre o como etiqueta con cierre normal

```
<template>
  <button :class="['btn btn-circle btn-secondary fixed', position]">
    <slot />
  </button>
</template>
```

- Ahora veo Hola mundo en el botón
- Le puedo pasar el svg, para hacerlo más bonito en modules/common/icons/AddCircle.vue

```
<template>
  <svg
     viewBox="0 0 24 24"
```

```
      fill="none"
      xmlns="http://www.w3.org/2000/svg">
      <g id="SVGRepo_bgCarrier"
      stroke-width="0"></g>
      <g id="SVGRepo_tracerCarrier"
      stroke-linecap="round"
      stroke-linejoin="round"></g>
      <g id="SVGRepo_iconCarrier">
         <path d="M15 12L12 12M12 12L9 12M12 12L12 9M12 12L12 15"
         stroke="#1C274C" stroke-width="1.5" stroke-linecap="round">
         </path>
         <path d="M7 3.33782C8.47087 2.48697 10.1786 2 12 2C17.5228 2 22 6.47715 22
12C22 17.5228 17.5228 22 12 22C6.47715 22 2 17.5228 2 12C2 10.1786 2.48697 8.47087
3.33782 7"
         stroke="#1C274C"
         stroke-width="1.5"
         stroke-linecap="round">
         </path>
    </g>
  </svg>
</template>
```

- Ahora puedo pasarle el componente a fab-button

```
<fab-button><add-circle /></fab-button>
```

- También podríamos usar una property para hacer esto
- Yo voy a querer saber que se hizo clici en el botón, uso defineEmits (va a emitir un 'click')
- Cuando haga click (@click es el on-click) emitirá el evento 'click' con $emit('click')

```
<template>
  <button
  :class="['btn btn-circle btn-secondary fixed', position]"
  @click="$emit('click')"
  >
    <slot />
  </button>
</template>

<script setup lang="ts">

interface Props{
    position?: 'top-left' | 'top-right' | 'bottom-left' | 'bottom-right'
}

withDefaults(defineProps<Props>(), {
    position: 'bottom-right'
})

defineEmits(['click'])
</script>
```

```
<style scoped>
@reference 'tailwindcss'

.top-left{
    @apply top-10 left-10;
}
.top-right{
    @apply top-10 right-10;
}
.bottom-left{
    @apply bottom-10 left-10;
}
.bottom-right{
    @apply bottom-10 right-10;
}

</style>
```

- Ahora en el fab-button tengo el evento @click

```
<fab-button @click="console.log('clicked')">
      <add-circle />
</fab-button>
```

Modals sin slots

- Al tocar el fab-button vamos a mostrar un modal que me sirva para capturar el valor del nuevo poroyecto para empezar a crearlo
- Busco en DaisyUI el modal Dialog modal
- Copio a partir del dialog (no desde el button) para implementar la lógica con Vue
- modules/common/components/InputModal.vue

```
<template>
    <dialog class="modal"> <!--si le añado open se muestra el modal-->
  <div class="modal-box">
    <h3 class="text-lg font-bold">Hello!</h3>
    <p class="py-4">Press ESC key or click the button below to close</p>
    <div class="modal-action">
      <form method="dialog">
        <!-- if there is a button in form, it will close the modal -->
        <button class="btn">Close</button>
      </form>
    </div>
  </div>
```

```
</dialog>
</template>
```

- Lo renderizo encima del fab-button en ProjectsView.vue
- Para que se vea el modal, tengo que añadirle el atributo open a dialog
- Le coloco un v-bind al open y le asigno un la prop open, de la que depende el modal para estar visible o no
- Si el button es de type button no cierra el modal, si es de tipo submit (para la propagación del formulario), si cierra
- Si no se especifica, por defecto es de tipo submit
- Creo un div de toda la pantalla para bloquear el fondo
- En las props necesito el open como boolean pues es la variable de control
- El modal también necesita emitir la data del input, por lo que uso defineEmits, y emitir un close para cerrar
- Creo la variable reactiva inputValue y se la paso con v-model al input
- En el form usamos @submit y le pasamos el valor del formulario
- Uso .prevent para evitar la propagación del formulario (para que no haga un refresh el navegador)
- InputModal.vue

```
<template>
    <dialog class="modal" :open="open">
  <div class="modal-box">
    <h3 class="text-lg font-bold">Hello!</h3>
    <p class="py-4">Press ESC key or click the button below to close</p>
    <div class="modal-action flex flex-col">
      <form method="dialog" @submit.prevent="submitValue">
        <input
        v-model="inputValue"
        type="text"
        placeholder="Nombre del proyecto"
        class="input input-bordered input-primary w-full flex-1"
        >
        <div class="flex justify-end mt-5 mr-4">
            <button class="btn">Close</button>
            <button type="submit" class="btn btn-primary">Aceptar</button>
        </div>
      </form>
    </div>
  </div>
</dialog>
<div class="modal-backdrop fixed top-0 left-0 z-10 bg-black opacity-40 w-screen h-
screen">
</div>
</template>

<script setup lang="ts">
```

```
import { ref } from 'vue';


    interface Props{
        open: boolean //siempre la necesito porque es la variable de control
    }

    defineProps<Props>()

    const emits= defineEmits<{
        close: [void], //close no emite nada
        value: [text: string] //value emite un text de tipo string
    }>()

    const inputValue= ref('')

    const submitValue=()=>{
        if(!inputValue.value){
            //foco en el input
        }

        emits('value', inputValue.value.trim()) //emito el valor
        emits('close')

        inputValue.value='' //reseteo el inputValue
    }
</script>
```

Funcionalidad del modal

- Ahora el modal no se cierra porque no hay una propagación del formulario
- Quiero que el foco del elemento aparezca cuando le doy a Aceptar para que se ilumine la caja de texto
- Creo inputRef, lo inicio en null y lo tipo

```
const inputRef= ref<HTMLInputElement | null>(null)
```

- Se lo coloco al input con ref="inputRef". No es necesario ponerle los : a ref (:ref). Es el único elemento que no lo necesita
- Cuando se monta el componente ya tenemos acceso a su referencia (el input)
- Lo usaré para poner el foco en la caja de texto cuando no haya un valor

```
const inputRef= ref<HTMLInputElement | null>(null)

const submitValue=()=>{
    if(!inputValue.value){
        inputRef.value?.focus()
        return
```

```
    }

    emits('value', inputValue.value.trim()) //emito el valor
    emits('close')

    inputValue.value='' //reseteo el inputValue
}
```

- La funcionalidad de abrir y cerrar el modal depende de nuestras props (con open)
- En el componente padre (ProjectsView.vue) definimos una variable reactiva modalOpen con false por defecto
- Se la paso al open con un v-bind para indicarle que no es un string sino una variable
- Cuando clicamos el FabButton, cambiamos el modalOpen a true

```
<template>
{...code}
  <input-modal :open="openModal" />
    <fab-button @click="openModal = true">
      <add-circle />
    </fab-button>
</template>

<script setup lang="ts">
import FabButton from '@/modules/common/components/FabButton.vue';
import InputModal from '@/modules/common/components/InputModal.vue';
import AddCircle from '@/modules/common/icons/AddCircle.vue';
import { ref } from 'vue';

const openModal= ref(false)
</script>
```

- En el InputModal usamos el v-if para mostrar el div con el background oscuro translúcido usando la prop open
- Cuando clico el botón de close, debo llamar al close que tiene que cambiar el valor de openModal para cerrar el modal y el fondo
- Usando $emit podría bastar

```
<button @click="$emit('close')" class="btn">Close</button>
```

- Y recibir el evento close desde el padre ProjectView

```
<input-modal :open="openModal" @close="openModal = false"/>
  <fab-button @click="openModal = true">
```

```
    <add-circle />
  </fab-button>
```

- Ahora necesitamos recibir el valor del input, creo una función onNewValue en ProjectsView.vue
- Se lo paso al evento value que definí con defineEmits
- Si el evento que emite el value simplemento lo pasamos como argumento a la función, podemos mandar la función solo por referencia

```
<template>
{...code}

  <input-modal
  :open="openModal"
  @close="openModal = false"
  @value="onNewValue"
  />
    <fab-button @click="openModal = true">
      <add-circle />
    </fab-button>
</template>

<script setup lang="ts">
import FabButton from '@/modules/common/components/FabButton.vue';
import InputModal from '@/modules/common/components/InputModal.vue';
import AddCircle from '@/modules/common/icons/AddCircle.vue';
import { ref } from 'vue';

const openModal= ref(false)

const onNewValue=(projectName: string)=>{
  console.log(projectName)
}
</script>
```

- En el InputModal.vue recuerda que emito el inputValue.value con el evento value y que el input tiene el inputValue en el v-model

```
//uso el v-model para guardar el valor del input en una variable
<input
    ref="inputRef"
    v-model="inputValue"
    type="text"
    placeholder="Nombre del proyecto"
    class="input input-bordered input-primary w-full flex-1"
    >

// emito el evento que he definido antes con defineEmits pasandole el valor

  const emits= defineEmits<{
```

```
        close: [void], //close no emite nada
        value: [text: string] //value emite un text de tipo string
    }>()

  emits('value', inputValue.value.trim()) //emito el valor (inputValue es el text)
```

- Ahora si le doy al botón abre el modal, pone el resto de la pantalla en negro bloqueándolo, si escribo en la caja de texto Hola mundo, lo imprime en consola, cierra el modal y cierra el fondo. Si aprieto el botón de close también cierra el modal
- Puedo personalizar el placeholder del modal agregando la prop en la interfaz, no hace falta usar un v-bind (:placeholder) porque le paso un string

```
interface Props{
      open: boolean, //siempre la necesito porque es la variable de control
      placeholder?: string
  }

//solo tengo que pasarle la prop al modal
 <input-modal
  :open="openModal"
  @close="openModal = false"
  @value="onNewValue"
  placeholder="Escribe aquí tu apellido"
  />
```

- Hagamos lo mismo con el título y subtítulo del modal
- Puedo usar v-if para hacer el subtitle opcional
- InputModal.vue

```
<template>
    <dialog id="my_modal_1" class="modal" :open="open">
  <div class="modal-box">
    <h3 class="text-lg font-bold">{{ title }}</h3>
    <p v-if="subTitle" class="py-4">{{ subtitle }}</p>
    <div class="modal-action flex flex-col">
      <form method="dialog" @submit.prevent="submitValue">
        <input
        ref="inputRef"
        v-model="inputValue"
        type="text"
        placeholder="Nombre del proyecto"
        class="input input-bordered input-primary w-full flex-1"
        >
        <div class="flex justify-end mt-5 mr-4">
            <button @click="$emit('close')" class="btn">Close</button>
            <button type="submit" class="btn btn-primary">Aceptar</button>
        </div>
      </form>
    </div>
```

```
  </div>
</dialog>
<div
v-if="open"
class="modal-backdrop fixed top-0 left-0 z-10 bg-black opacity-40 w-screen h-screen">
</div>
</template>

<script setup lang="ts">
import { ref } from 'vue';


    interface Props{
        open: boolean, //siempre la necesito porque es la variable de control
        placeholder?: string,
        title: string,
        subtitle?: string
    }

    defineProps<Props>()

    const emits= defineEmits<{
        close: [void], //close no emite nada
        value: [text: string] //value emite un text de tipo string
    }>()

    const inputValue= ref('')
    const inputRef= ref<HTMLInputElement | null>(null)

    const submitValue=()=>{
        if(!inputValue.value){
            inputRef.value?.focus()
            return
        }

        emits('value', inputValue.value.trim()) //emito el valor
        emits('close')

        inputValue.value='' //reseteo el inputValue
    }
</script>
```

• Ahora le paso el titulo y el subtitulo como string en las props desde el padre ProjectsView

```
<input-modal
  :open="openModal"
  @close="openModal = false"
  @value="onNewValue"
  title="Añade aquí tu proyecto"
  subtitle="Desde aquí puedes añadir tu proyecto"
  placeholder="Nombre del proyecto"
  />
```

named Slots - Slots con nombre

- Un caso típico de Slots con nombre es un modal, donde tenemos un header, un cuerpo y un footer con acciones (los botones)
- Creo modules/common/components/CustomModal.vue
- Le pego el código del modal de DiasyUI sin el button (a partir del dialog)
- open es lo que hace que esté visible el componente, uso las props con el withDefaults para setear open como false por defecto
- Uso el v-bind en open para pasarle la prop
- Le coloco el componente que creé para dejar el fondo de la pantall opaco. Lo coloco fuera del dialog para que no afecte el modal

```
<template>
 <dialog class="modal" :open="open">
  <div class="modal-box">
    <!--Header-->
    <h3 class="text-lg font-bold">Hello!</h3>
    <p class="py-4">Press ESC key or click the button below to close</p>
    <!--Body-->
    <!--Footer/Actions-->
    <div class="modal-action">
      <form method="dialog">
        <button class="btn">Close</button>
      </form>
    </div>
  </div>
</dialog>
  <div
    v-if="open"
    class="modal-backdrop fixed top-0 left-0 z-10 bg-black opacity-40 w-screen h-
screen">
  </div>
</template>


<script setup lang="ts">
    interface Props{
        open: boolean
    }
    withDefaults(defineProps<Props>(),{
        open: false
    })
</script>
```

- Lo llamos en ProjectView, añado un nuevo botón que usa la variable reactiva customOpenModal en el on-click

```
<template>
{...code}
<custom-modal :open="customOpenModal"/>

<fab-button @click="customOpenModal = true" position="bottom-left">
    <add-circle />
  </fab-button>
</template>

<script>
const customOpenModal=ref(false)
{...code}
</script>
```

- Hay que quitar el fondo opaco cuando cierro. No lo vamos a controlar con un emit, sino de otra manera
- Mandaremos las piezas desde ProjectsView a nuestro componente CustomModal
- Dividamos el componente CustomModal en header, body, y footer actions
- Puedo especificarle un nombre al slot

```
<template>
 <dialog class="modal" :open="open">
  <div class="modal-box">
    <!--Header-->
    <div class="border-b border-b-blue-500">
         <h1 class="text-lg font-bold">Header</h1>
         <slot name="header" />
    </div>
    <!--Body-->
    <div class="my-5">
        <h1>Body</h1>
        <slot name="body" />
    </div>
    <!--Footer/Actions-->
    <div class="modal-action">
        <div class="border-t border-t-blue-500 pt-2">
            <h1>Footer actions</h1>
            <slot name="footer" />
        </div>
    </div>
  </div>
</dialog>
<div
  v-if="open"
  class="modal-backdrop fixed top-0 left-0 z-10 bg-black opacity-40 w-screen h-screen">
</div>
</template>


<script setup lang="ts">
    interface Props{
        open: boolean
```

```
    }
    withDefaults(defineProps<Props>(),{
        open: false
    })
</script>
```

- Solo tengo que mandarle el contenido desde el padre usando el template numeral y el nombre del slot

```
<custom-modal :open="customOpenModal">
  <template #header>
    <h1 class="text-3xl">Titulo del modal</h1>
  </template>
</custom-modal>
```

10 Vue Herrera - Pinia (Gestor de estado)

- Trabajar con el store en Pinia no es distinto que trabajar con un composable. con la excepción que la función watch no está
- Imagina una piña, la piel está compuesta por hexágonos. Esos hexágonos serían los stores de manera independiente
- Desde las devtools se puede ver el store (desde el icono de piña)
- Desde las devtools se pueden borrar estados
- En cada proyecto yo puedo añadir infinitas tareas a completar
- Tendremos una barra de progreso (Avance)
- En este ejercicio no uniremos varios stores, pero se pueden unir

Instalación de Pinia

- Pinia es básciamente un composable, solo que la función de inicialización es un poco diferente
- No se hace con un composable sino con Pinia porque nos da ciertos superpoderes

> npm i pinia

- main.ts

```
import './assets/styles.css'
import { createApp } from 'vue'
import { createPinia } from 'pinia'

import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(createPinia())
app.use(router)

app.mount('#app')
```

- Pinia es perezoso por defecto (no lo carga si no se utiliza)

Projects Store

- Se aconseja ponerle use al principio y Store al final del store

- Prefiero tener los stores acoplados a los módulos
- Si regreso projects como computed podría hacer la propiedad de solo lectura, ya que usando solo ref se puede modificar
- src/modules/projects/stores//projects.store.ts

```
import { defineStore } from "pinia";
import { ref } from "vue";
import type { Project } from "../interface/project.interface";

export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref<Project[]>([])

    return {projects}
})
```

- La interfaz de proyectos en modules/projects/interface/projects.interface.ts

```
export interface Project{
    id: string
    name: string
    tasks: Task[]
}

export interface Task{
    id: string
    name: string
    completedAt?: Date
}
```

- ¿Cómo uso el store?
- Me creo un estado incial (initialLoad)

```
import { defineStore } from "pinia";
import { ref } from "vue";
import type { Project } from "../interface/project.interface";

const initialLoad = (): Project[]=>{
    return [
        {
            id: '1',
            name: 'project 1',
            tasks:[]
        },
        {
            id: '2',
            name: 'project 2',
            tasks:[]
        },
    ]
```

```
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref<Project[]>(initialLoad())

    return {projects}
})
```

- Para usarlo voy al script setup de modules/projects/views/ProjectView.vue

```
const projectStore = useProjectsStore()
```

- Con eso ya me aparece el estado en las vueDevtools
- Una vez hago referencia al store ya se inicializa y luego hace referencia a la data que hay (es como un singleton)

Usando el store

- Hay varias maneras de consumir nuestro store
- Si uso la desestructuración se pierde la reactividad del store

```
//se pierde la reactividad
const {projects} = useProjectsStore()
```

- Para aplicar desestructuración se debe usar storeToRefs

```
const {projects} = storeToRefs(useProjectsStore())
```

- Pero también se puede usar tal cual lo teníamos tal que así
- Uso el componente progress de DaisyUI para la barra de avance, de momento con el avance en duro (value=40)
- ProjectView.vue

```
//script setup
const projectStore = useProjectsStore()

//template
 <tbody>
    <!-- row 1 -->
    <tr v-for="(project,index) in projectStore.projects" :key="project.id">
    <th>{{ index+1 }}</th>
    <td>{{ project.name }}</td>
```

```
    <td>{{ project.tasks.length }}</td>
    <td>
        <progress class="progress progress-primary w-56" value="40" max="100"></
progress>
    </td>
    </tr>
</tbody>
```

- Desde las devTools puedo cambiarle el nombre al proyecto, añadir una task, etc
- Con el tasks.length, si añado una tarea aparece 1 en Tareas
- El estado es volátil. Para hacerlo persistente lo leeriamos de un backend o grabar en el localStorage
- Podemos crear una propiedad computada para evitar que desde cualquier otro lugar se puedan realizar modificaciones del store
- En el ProjecstView script setup esto funciona

```
<template>
{...code}
  <input-modal
  :open="openModal"
  @close="openModal = false"
  @value="onNewValue"
  title="Añade aquí tu proyecto"
  subtitle="Desde aquí puedes añadir tu proyecto"
  placeholder="Nombre del proyecto"
  />
</template>

<script>
const openModal= ref(false)
const customOpenModal=ref(false)

const projectStore = useProjectsStore()

const onNewValue=(projectName: string)=>{
  projectStore.projects.push({
    id: '3',
    name: projectName,
    tasks: []
  })
}
</script>
```

- Hay varios problemas en hacerlo de esta manera
- Puede ser que el id sea una función UUID, o se tiene que crear basado en cierta estructura
- Yo no quiero que cualquiera pueda llegar a mi store y modificarlo de esa manera
- Podemos trabajar con una propiedad computada

- Ya no ocupo regresar los proyectos, así me aseguro de que los proyectos solo puedan ser accedidos y modifcados dentro del store
- Es como crearse una propiedad privada de solo lectura (en teoría. voy a poder insertar)
- projects.store.ts

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { Project } from "../interface/project.interface";

const initialLoad = (): Project[]=>{
    return [
        {
            id: '1',
            name: 'project 1',
            tasks:[]
        },
        {
            id: '2',
            name: 'project 2',
            tasks:[]
        },
    ]
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref<Project[]>(initialLoad())

    return {
        // Properties
        //projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value])

        // Actions
    }
})
```

- En ProjectView.vue se está quejando porque ya no tengo .projects sino .projectList
- Ahora si escribo en el input Nuevo Proyecto sigue publicándolo, no debería poder usar el .push
- Esta es la manera en la que vamos a crear nuestros getters (con propiedades computadas)
- Podemos crear la función para agregar proyectos en el store
- Como todo esto está pasando por referencia gracias al computed, vamos a tener la actualización
- Instalo uuid para empezar a trabajar con UUID

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { Project } from "../interface/project.interface";

const initialLoad = (): Project[]=>{
    return [
        {
            id: '1',
            name: 'project 1',
            tasks:[]
        },
        {
            id: '2',
            name: 'project 2',
            tasks:[]
        },
    ]
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref<Project[]>(initialLoad())

    return {
        // Properties
        //projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value])
        // Actions
    }
})
```

- Esta acción es la que voy a usar en lugar de onNewValue
- ProjectsView.vue

```
 <input-modal
  :open="openModal"
  @close="openModal = false"
  @value="projectStore.addProject"
  title="Añade aquí tu proyecto"
  subtitle="Desde aquí puedes añadir tu proyecto"
  placeholder="Nombre del proyecto"
  />
```

- Como el value está emitiendo un string y lo que espera es un string, puedo passarle la función por referencia
- Trabajamos con .push porque no tenemos un backend, sino serían tareas asíncronas

Colocar foco en el modal

- Quiero que al abrir el modal se ponga automáticamente el foco en la caja de texto
- Quiero poner el foco tan pronto cambie el openModal a true (el componente ya está montado de antes, por lo que las funciones del ciclo de vida no me servirían)
- watch es para estar pendiente de una propiedad en concreto (en este caso las props que es donde tengo open)
- En el callback tengo las nuevas props, puedo desestructurar el open
- watchEffect sirve para estar pendiente de las propiedades reactivas que tiene el componente
- Uso nextTick de Vue para darle tiempo a la referencia a tomar el componente, ya que Vue va muy rápido (!)
- modules/common/components/InputModal.vue

```
//script setup
   const props= defineProps<Props>()

    watch(props, async ({ open }) => {
      if (open) {
      await nextTick();
      inputRef.value?.focus();
    }
});
```

Store to LocalStorage

- Usaremos el LocalStorage porque no tenemos backend+
- Podemos hacerlo todo en el initialLoad del store para obtener el estado del localStorage y retornarlo, luego usar $subscribe con el store para guardar el estado incial
- Pero hay un paquete de npm llamado **pinia-plugin-persistedstate**
- Solo hay que poner en la definición del store, después de las llaves, el persist: true
- Usaremos **@vueuse/core**, es más multiuso

> https://vueuse.org/

- Hago uso de la función useLocalStorage, le pongo el tipado (se lo quito a ref)
- projects.store.ts

```
export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref( useLocalStorage<Project[]>('projects', initialLoad()))

    const addProject =(name: string) =>{
        if(name.length === 0) return
```

```
        projects.value.push({
            id: uuidv4(),
            name,
            tasks: []
        })
    }
    return {
        // Properties
        //projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),

        // Actions
        addProject
    }
})
```

- Puedo inicializarlo como un arreglo vacío, es totalmente válido

```
const projects = ref( useLocalStorage<Project[]>('projects', []))
```

- Ahora si recargo el navegador la información es persistente
- VueUse tiene muchas utilidades muy útiles, que mezclado con los composables lo hacen muy potente

Menú lateral

- Si no hay proyectos debería aparecer “No hay proyectos”
- Y si hay proyectos debería aparecer el menú
- Para facilitar el ejercicio retorno projects del store

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { Project } from "../interface/project.interface";
import {v4 as uuidv4} from 'uuid'
import { useLocalStorage } from "@vueuse/core";

const initialLoad = (): Project[]=>{
    return [
        {
            id: uuidv4(),
            name: 'project 1',
            tasks:[]
        },
        {
            id: uuidv4(),
            name: 'project 2',
```

```
            tasks:[]
        },
    ]
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref( useLocalStorage<Project[]>('projects', initialLoad()))

    const addProject =(name: string) =>{
        if(name.length === 0) return

        projects.value.push({
            id: uuidv4(),
            name,
            tasks: []
        })
    }
    return {
        // Properties
        projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),

        // Actions
        addProject
    }
})
```

- Buscamos modules/projects/components/SideMenu.vue
- Importo el store en el script
- A este componente solo le va a importar saber como luce este store
- Uso el length de los proyectos con un v-if para mostrar el párrafo “No hay proyectos”
- En el ul un v-else

```
<template>
  <aside class="bg-base-200 w-72 min-h-screen">
    <h2 class="text-lg font-bold mx-4">Proyectos</h2>
    <p v-if="projectsStore.projects.length == 0" class="text-sm text-gray-500 mx-4">No
hay proyectos</p>
    <!--Menu-->
     <ul v-else class="menu rounded-box w-56">
        <li><a>Item 1</a></li>
        <li>
            <details open>
            <summary>Parent</summary>
            <ul>
                <li><a>Submenu 1</a></li>
                <li><a>Submenu 2</a></li>
                <li>
```

```
                <details open>
                    <summary>Parent</summary>
                    <ul>
                    <li><a>Submenu 1</a></li>
                    <li><a>Submenu 2</a></li>
                    </ul>
                </details>
                </li>
            </ul>
            </details>
        </li>
        <li><a>Item 3</a></li>
        </ul>
    </aside>
</template>

<script setup lang="ts">
import { useProjectsStore } from '../stores/projects.store';

    const projectsStore = useProjectsStore()
</script>
```

- Bien podría crear un getter para el length de los proyectos
- No hace falta que use un ternario porque ya devuelve una expresión booleana
- projects.store.ts

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { Project } from "../interface/project.interface";
import {v4 as uuidv4} from 'uuid'
import { useLocalStorage } from "@vueuse/core";

const initialLoad = (): Project[]=>{
    return [
        {
            id: uuidv4(),
            name: 'project 1',
            tasks:[]
        },
        {
            id: uuidv4(),
            name: 'project 2',
            tasks:[]
        },
    ]
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref( useLocalStorage<Project[]>('projects', initialLoad()))

    const addProject =(name: string) =>{
```

```
        if(name.length === 0) return

        projects.value.push({
            id: uuidv4(),
            name,
            tasks: []
        })
    }
    return {
        // Properties
        projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),
        noProjects: computed(()=>projects.value.length == 0),

        // Actions
        addProject
    }
})
```

- Ahora puedo usar el getter directamente en lugar del .length

```
<p v-if="projectsStore.noProjects" class="text-sm text-gray-500 mx-4">No hay
proyectos</p>
```

- details solo aparece si el proyecto tiene tareas, sino solo debería aparecer el enlace directo
- El template se puede usar como un Fragments de React, no solo con los slots
- Lo que sea que ponga dentro del template, si está dentro de un li, va a ser hijo directo del li
- Coloco lo que había en details dentro del template, le quito el atributo open para que se tenga que clicar para mostrarlo
- Uso RouterLink para navegar al proyecto (todavía no está montado en el router)
- Uso otro template con un v-else para mostrar solo el enlace al proyecto con el nombre (sin las tareas que no hay)

```
<template>
  <aside class="bg-base-200 w-72 min-h-screen">
    <h2 class="text-lg font-bold mx-4">Proyectos</h2>
    <p v-if="projectsStore.noProjects" class="text-sm text-gray-500 mx-4">No hay
proyectos</p>
    <!--Menu-->
     <ul v-else class="menu rounded-box w-56">
        <li v-for="project in projectsStore.projectList" :key="project.id">
        <template v-if="project.tasks.length > 0">
         <details>
            <summary>
                <router-link :to="`/project/${project.id}`">{{ project.name }}</router-
link>
            </summary>
```

```
            <ul>
                <li v-for="task in project.tasks">
                  <router-link :to="`/project/${project.id}`">{{ task }}</router-
link>
                </li>
            </ul>
         </details>
        </template>

        <template v-else>
            <summary>
                <router-link :to="`/project/${project.id}`">{{ project.name }}</router-
link>
            </summary>
        </template>
        </li>
        </ul>
    </aside>
</template>

<script setup lang="ts">
import { useProjectsStore } from '../stores/projects.store';

    const projectsStore = useProjectsStore()
</script>
```

- Todavía no tengo nada apuntando a esos paths de los RouterLink
- Como no tienen tareas, no aparece la flecha para clicar y desplegar el menú
- falta la pantalla donde recibir el argumento (el id), validar que existe, si no hay que salir de ahi, etc

ProjectView - pantalla para ver el proyecto

- Creo en modules/projects/views/ProjectView.vue (el general es ProjectsView.vue)
- Digamos que por ahora lo único que quiero hacer es tomar el nombre del proyecto
- ProjectView.vue

```
<template>
    <div>
        <h1 class="text-3xl">{{ 'hola mundo' }}</h1>
    </div>
</template>
```

- Coloquemos la página en una ruta del router

```
import ProjectsLayout from '@/modules/projects/layouts/ProjectsLayout.vue'
import { createRouter, createWebHistory } from 'vue-router'
```

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      redirect: {name: 'projects'},
      component: ProjectsLayout,
      children:[
        {
          path:'projects',
          name: 'projects',
          component: ()=>import('@/modules/projects/views/ProjectsView.vue')
        },
        {
          path:'project/:id',
          name: 'project',
          component: ()=>import('@/modules/projects/views/ProjectView.vue')
        }
      ]
    }
  ],
})

export default router
```

- Si ahora clico en alguno de los proyectos del menú lateral me muestra el hola mundo
- En la url aparece algo como http://localhost:5173/project/f2d88cbb-3554-48fc-992e-c295590262ad
- Una duda: que pasa si la persona recarga el navegador estando en esta página, y dado que en nuestro Pinia Store tiene que cargarse, porque está basado en el localStorage, qué pasa en ese caso?
- Lo primero, tenemos que ser capaces de obtener el id del url. Hay varias maneras, veamos una que no hemos visto
- Usando useRoute, puedo desestructurar params que es donde tengo el id
- ProjectView.vue

```
//script setup
const {params} = useRoute()

console.log(params.id)
```

- También es bien común mandar las props en true desde el router
- router/index.ts

```
{
path:'project/:id',
props: true,
```

```
name: 'project',
component: ()=>import('@/modules/projects/views/ProjectView.vue')
}
```

- Para obtenerlas tengo que definirlas (las tipo)
- Ahora ya puedo usar el id
- ProjectView.vue

```
<template>
    <div>
        <h1 class="text-3xl">{{ id }}</h1>
    </div>
</template>

<script setup lang="ts">
import { useRoute } from 'vue-router';

//const {params} = useRoute()

interface Props{
    id: string
}

defineProps<Props>()

</script>

<style scoped>

</style>
```

- Vamos a usar el componente Breadcrumbs de DaisyUI para ir creando el Path tipo Home>Documents>Add Document
- Selecciono Breadcrumbs with icons
- Creo modules/common/components/BreadCrumbs.vue
- Pego el código dentro de un template

```
<template>
    <div class="breadcrumbs text-sm">
  <ul>
    <li>
      <a>
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          class="h-4 w-4 stroke-current">
          <path
            stroke-linecap="round"
```

```
            stroke-linejoin="round"
            stroke-width="2"
            d="M3 7v10a2 2 0 002 2h14a2 2 0 002-2V9a2 2 0 00-2-2h-6l-2-2H5a2 2 0 00-2
2z"></path>
        </svg>
        Home
      </a>
    </li>
    <li>
      <a>
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          class="h-4 w-4 stroke-current">
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M3 7v10a2 2 0 002 2h14a2 2 0 002-2V9a2 2 0 00-2-2h-6l-2-2H5a2 2 0 00-2
2z"></path>
        </svg>
        Documents
      </a>
    </li>
    <li>
      <span class="inline-flex items-center gap-2">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          class="h-4 w-4 stroke-current">
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M9 13h6m-3-3v6m5 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"></path>
        </svg>
        Add Document
      </span>
    </li>
  </ul>
</div>
</template>
```

- Hacemos algunos cambios, el del medio no lo necesito (dejo solo el primero y el tercero), usamos un RouterLink en lugar de un anchor tag, uso las props para colocarle el name al el último elemento (no tengo más subrutas)
- Coloco un   al Home, que es colocar un espacio porque queda muy pegado
- BreadCrumbs.vue

```
<template>
    <div class="breadcrumbs text-sm">
  <ul>
    <li>
      <router-link to="/projects">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          class="h-4 w-4 stroke-current">
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M3 7v10a2 2 0 002 2h14a2 2 0 002-2V9a2 2 0 00-2-2h-6l-2-2H5a2 2 0 00-2
2z"></path>
        </svg>
         
        Home
    </router-link>
    </li>
    <li>
      <span class="inline-flex items-center gap-2">
        <svg
          xmlns="http://www.w3.org/2000/svg"
          fill="none"
          viewBox="0 0 24 24"
          class="h-4 w-4 stroke-current">
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M9 13h6m-3-3v6m5 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"></path>
        </svg>
        {{ name }}
      </span>
    </li>
  </ul>
</div>
</template>

<script lang="ts" setup>
interface Props{
    name: string
}
defineProps<Props>()
</script>
```

- Usemos el BreadCrumbs en ProjectView.vue
- De name le paso project en duro

```vue
<template>
    <div>
        <bread-crumbs name="project"></bread-crumbs>
    </div>
</template>

<script setup lang="ts">
import BreadCrumbs from '@/modules/common/components/BreadCrumbs.vue';

interface Props{
    id: string
}

defineProps<Props>()

</script>

<style scoped>

</style>
```

- Si estoy en ProjectsView (el home) no aparecen los BreadCrumbs, pero si voy a project 1 en el menú lateral izquierdo, aparece Home (con un icono) / project (con otro icono)
- Quiero que en vez de poner project en duro aparezca el nombre del proyecto
- Puedo tomar el id en ProjectView, ir al store y tomar ese proyecto
- Hagámoslo mal para luego hacerlo bien
- En el script de ProjectView.vue

```vue
interface Props{
    id: string
}

const props = defineProps<Props>()
const projectStore = useProjectsStore()             //puede ser undefined
const project= projectStore.projectList.find(project => project.id== props.id)
```

- Uso project en el template de ProjectView.vue
- Si no hay un project le paso un string con no-name
- Uso v-bind en el name para poder añadir lógica y no solo un string

```vue
<bread-crumbs :name="project?.name ?? 'no-name'"></bread-crumbs>
```

- Parece funcionar bien, pero cuando estoy en la página de un proyecto debo volver al Home para ir a la de otro proyecto
- Cambia la url pero no se está destruyendo el componente y volviendo a crear
- Si me quedo en otro proyecto y recargo el navegador, ahí si cambia

- Tengo que estar pendiente de si el id cambia
- Ya hemos visto el watch
- Podemos hacerlo con computed
- ProjectView.vue

```
//script
interface Props{
    id: string
}

const props = defineProps<Props>()
const projectStore = useProjectsStore()

const project = computed(() => {
  return projectStore.projectList.find((x) => x.id === props.id);
});
```

- De esta manera si puedo cambiar entre proyectos
- Otra manera usando el watch, pero no tenemos el valor (aparece no-name) cuando se recarga
    - Es por como funciona el watch
    - La primera vez que se va a la página no cambia, por eso no se ejecuta
    - Para que se dispare nada más montar el componente puedo usar inmediate en true

```
//script
const project = ref<Project|null>()

watch(()=>props.id, ()=>{
    project.value= projectStore.projectList.find((x) => x.id === props.id);
},
{
    immediate: true
})
```

- **Explicación de computed**:
- Se utiliza cuando quieres derivar un valor de otras reactividades, como en este caso, projectStore.projectList y props.id.
- Es un valor reactivo que se actualiza automáticamente cada vez que sus dependencias cambian.
- La idea detrás de computed es que estás creando una propiedad calculada que depende de otras propiedades reactivas, y Vue lo optimiza para recalcular solo cuando alguna de esas dependencias cambia. No es necesario hacer nada explícito para “guardar” el resultado, ya que computed retorna un valor que siempre estará actualizado.
- **El watch**:

- Se utiliza cuando quieres ejecutar un efecto secundario (side effect) en respuesta a un cambio en los valores reactivos.
- Usualmente se usa para ejecutar código que no necesariamente debe ser calculado, sino que debe ejecutar alguna acción cuando los datos cambian. En tu ejemplo, estás buscando un proyecto y luego actualizando el valor de project.
- watch no retorna un valor; más bien, actúa sobre una acción. Aquí estás usando project.value directamente dentro de un watch, lo que hace que sea más adecuado si tienes efectos secundarios.
- Optimización: computed es más eficiente en términos de rendimiento en comparación con watch cuando se trata de derivar valores, porque solo se recalcula cuando alguna de sus dependencias cambia, y Vue lo maneja automáticamente. Si usas watch, cada vez que props.id cambie, el efecto se ejecutará y se actualizará manualmente el valor, lo que puede ser innecesario si solo quieres un valor calculado.
- ¿Cuándo usar watch en vez de computed?
- Si necesitas ejecutar acciones secundarias como llamar a una API, actualizar el estado de otros elementos, o hacer cosas que no sean directamente valores derivados.
- Si el cálculo no es simplemente un valor derivado sino que implica lógica compleja o efectos secundarios.

Redireccionar si no existe el proyecto

- Si navegamos a un URL que no existe (con un id fake) aparece Home / no-name
- Hagamos la redirección, en ProjectView.vue hago udo del useRouter
- En ProjectView.vue

```
//script
interface Props{
    id: string
}

const router = useRouter()
const props = defineProps<Props>()
const projectStore = useProjectsStore()

const project = ref<Project|undefined>()

watch(()=> props.id, ()=>{
    project.value= projectStore.projectList.find((x) => x.id === props.id);
    if(!project.value){
        router.replace('/')
    }
}, {
```

```
    immediate: true
})
```

- Pongo el breadcrumb dentro de un section dentro de un div con w-full (en ProjectView.vue)
- Hago algunos cambios en el template, agrego otro section y una tabla (que busco en DaisyUI), Table with a row that highlights on hover
- El componente luce así

```
    <div class="overflow-x-auto">
  <table class="table">
    <!-- head -->
    <thead>
      <tr>
        <th></th>
        <th>Name</th>
        <th>Job</th>
        <th>Favorite Color</th>
      </tr>
    </thead>
    <tbody>
      <!-- row 1 -->
      <tr>
        <th>1</th>
        <td>Cy Ganderton</td>
        <td>Quality Control Specialist</td>
        <td>Blue</td>
      </tr>
      <!-- row 2 -->
      <tr class="hover:bg-base-300">
        <th>2</th>
        <td>Hart Hagerty</td>
        <td>Desktop Support Technician</td>
        <td>Purple</td>
      </tr>
      <!-- row 3 -->
      <tr>
        <th>3</th>
        <td>Brice Swyre</td>
        <td>Tax Accountant</td>
        <td>Red</td>
      </tr>
    </tbody>
  </table>
</div>
```

- Borro todos los rows exceptuando el que tiene el hover y customizo el header. Meto la tabla en el segundo section
- Pongo un m-2 en el section para que no quede todo tan pegado a los bordes
- ProjectView.vue

```vue
<template>
    <div class="w-full">
        <section class="m-2">
            <bread-crumbs :name="project?.name ?? 'no-name'"></bread-crumbs>
        </section>

        <section class="m-2">
            <template>
                <div class="overflow-x-auto">
            <table class="table">
                <!-- head -->
                <thead>
                <tr>
                    <th class=""w-14>Completada</th>
                    <th>Tarea</th>
                    <th>Completada en</th>
                </tr>
                </thead>
                <tbody>
                <tr class="hover:bg-base-300">
                    <th>2</th>
                    <td>Hart Hagerty</td>
                    <td>Desktop Support Technician</td>
                    <td>Purple</td>
                </tr>
                </tbody>
            </table>
            </div>
            </template>
        </section>
    </div>
</template>

<script setup lang="ts">
import BreadCrumbs from '@/modules/common/components/BreadCrumbs.vue';
import { useProjectsStore } from '../stores/projects.store';
import { computed, ref, watch } from 'vue';
import type { Project } from '../interface/project.interface';
import { useRouter } from 'vue-router';

interface Props{
    id: string
}

const router = useRouter()
const props = defineProps<Props>()
const projectStore = useProjectsStore()

const project = ref<Project|undefined>()

watch(()=> props.id, ()=>{
    project.value= projectStore.projectList.find((x) => x.id === props.id);
    if(!project.value){
        router.replace('/')
    }
```

```
}, {
    immediate: true
})
</script>
```

• Creo otro row en la tabla con un input

```
<section class="m-2">
    <div class="overflow-x-auto">
<table class="table">
    <!-- head -->
    <thead>
    <tr>
        <th class="w-14">Completada</th>
        <th>Tarea</th>
        <th>Completada en</th>
    </tr>
    </thead>
    <tbody>
    <tr class="hover:bg-base-300">
        <th>2</th>
        <td>Hart Hagerty</td>
        <td>Desktop Support Technician</td>
        <td>Purple</td>
    </tr>
    <tr class="hover:bg-base-300">
        <th></th>
        <td>
            <input type="text"
            class="input input-primary w-full opacity-60 tranisition-all"
            placeholder="Nueva tarea"
            >
        </td>
    </tr>
    </tbody>
</table>
</div>
</section>
```

• Cuando el foco esté encima quiero que se vea fuerte (sin la opacidad)

• Fácil con tailwind

```
 <input type="text"
  class="input input-primary w-full opacity-60 tranisition-all hover:opacity-100
focus:opacity-100"
  placeholder="Nueva tarea"
  >
```

Agregar tareas al proyecto

- Creo la función en el store para agregar tarea. La retorno para poder usarla
- projects.store.ts

```
//store

const addTaskToProject =(projectId: string, taskName: string)=>{

    if(taskName.trim().length === 0) return

    const project= projects.value.find(p => p.id === projectId)

    if(!project) return

    project.tasks.push({
    id: uuidv4(),
    name: taskName,
    })

}
```

- Uso el v-model y el @keypress.enter para insertar la tarea, me creo una función para poder borrar la tarea del input una vez insertada
- No ocupo el evento porque todo lo tengo a la mano
- Uso un v-for para renderizar la tarea

```
<template>
  <tbody>
    <tr v-for="task in project?.tasks" :key="task.id" class="hover:bg-base-300">
        <th>2</th>
        <td>{{ task.name }}</td>
        <td>Desktop Support Technician</td>
        <td>Purple</td>
    </tr>
    <tr class="hover:bg-base-300">
        <th></th>
        <td>
            <input type="text"
            class="input input-primary w-full opacity-60 tranisition-all
hover:opacity-100 focus:opacity-100"
            placeholder="Nueva tarea"
            v-model="newTask"
            @keypress.enter="addTask"
            >
        </td>
    </tr>
  </tbody>
{...code}
</template>
```

```
<script setup lang="ts">
//imports

interface Props{
    id: string
}

const router = useRouter()
const props = defineProps<Props>()
const projectStore = useProjectsStore()
const project = ref<Project|undefined>()
const newTask = ref('')

watch(()=> props.id, ()=>{
    project.value= projectStore.projectList.find((x) => x.id === props.id);
    if(!project.value){
        router.replace('/')
    }
}, {
    immediate: true
})

const addTask=()=>{
    if(!project.value) return
    projectStore.addTaskToProject(project.value.id, newTask.value)
    newTask.value=""
}
</script>
```

Completar tareas y progreso

- Donde tenemos ese número 2 en la tabla de ProjectView.vue colocaremos un chekcbox para marcar la tarea completada o no
- En DaisyUI, un checkbox no es más que un input que tiene el type checkbox, con la propiedad checked en true por defecto
- Uso la doble negación para el cheked con el completedAt. Con una negación, estoy diciendo “si no existe una fecha”, con doble negación lo transformo en un valor booleano. Si tiene un valor será true (con la doble negación)

```
<tr v-for="task in project?.tasks" :key="task.id" class="hover:bg-base-300">
    <th>
        <input type="checkbox"
            :checked="!!task.completedAt"
            class="checkbox checkbox-primary"
        >
    </th>
    <td>{{ task.name }}</td>
    <td>Desktop Support Technician</td>
```

```
    <td>Purple</td>
</tr>
```

- Tenemos que insertar una fecha en completedAt para que al marcar el checkbox se ponga en true
- Creo toggletAsk en projects.store.ts

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { Project } from "../interface/project.interface";
import {v4 as uuidv4} from 'uuid'
import { useLocalStorage } from "@vueuse/core";

const initialLoad = (): Project[]=>{
    return [
        {
            id: uuidv4(),
            name: 'project 1',
            tasks:[]
        },
        {
            id: uuidv4(),
            name: 'project 2',
            tasks:[]
        },
    ]
}



export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref( useLocalStorage<Project[]>('projects', initialLoad()))

    const addProject =(name: string) =>{
        if(name.length === 0) return

        projects.value.push({
            id: uuidv4(),
            name,
            tasks: []
        })
    }

    const addTaskToProject =(projectId: string, taskName: string)=>{

        if(taskName.trim().length === 0) return

       const project= projects.value.find(p => p.id === projectId)

       if(!project) return

       project.tasks.push({
        id: uuidv4(),
```

```
            name: taskName,
           })

    }

    const toggleTask = (projectId: string, taskId: string)=>{
       const project= projects.value.find(p => p.id === projectId)
        if(!project) return

        const task= project.tasks.find(t=> t.id === taskId)
        if(!task) return

        task.completedAt = task.completedAt ? undefined : new Date()
    }

    return {
        // Properties
        projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),
        noProjects: computed(()=>projects.value.length == 0),

        // Actions
        addProject,
        addTaskToProject,
        toggleTask
    }
})
```

- Usamos toggletAsk en ProjectView.vue
- Uso el on-change en el input y le paso los argumentos a la función. Uso ! en project para que no se queje, ya que si no hay proyecto no podría entrar en la página, por lo que seguro que hay proyecto

```
<input type="checkbox"
        :checked="!!task.completedAt"
        class="checkbox checkbox-primary"
        @change="projectStore.toggleTask(project!.id, task.id)"
>
```

- Renderizo el completedAT

```
<tr v-for="task in project?.tasks" :key="task.id" class="hover:bg-base-300">
    <th>
        <input type="checkbox"
            :checked="!!task.completedAt"
            class="checkbox checkbox-primary"
            @change="projectStore.toggleTask(project!.id, task.id)"
        >
    </th>
```

```
    <td>{{ task.name }}</td>
    <td>{{ task.completedAt }}</td>
</tr>
```

- Si le doy al checkbox aparece la fecha completa en la columna de la derecha (con el header Completada en)
- Tenemos que indicar la barra de progreso según las tareas completadas o no
- Tenemos que calcular el value (del componente progress) basado en las tareas completadas
- Hagamos la función en el store
- Creo una propiedad computada llamada projectsWithCompletion
- Necesito regresar un nuevo arreglo con el id del proyecto, el nombre, el númerod e tareas y el completion
- .map me va a permitir regresar otro arreglo basado en un arreglo
- projects.store.ts

```
export const useProjectsStore= defineStore('projects', ()=>{
    const projects = ref( useLocalStorage<Project[]>('projects', initialLoad()))

    const addProject =(name: string) =>{
        if(name.length === 0) return

        projects.value.push({
            id: uuidv4(),
            name,
            tasks: []
        })
    }

    const addTaskToProject =(projectId: string, taskName: string)=>{

        if(taskName.trim().length === 0) return

       const project= projects.value.find(p => p.id === projectId)

       if(!project) return

       project.tasks.push({
        id: uuidv4(),
        name: taskName,
       })

    }

    const toggleTask = (projectId: string, taskId: string)=>{
       const project= projects.value.find(p => p.id === projectId)
        if(!project) return

        const task= project.tasks.find(t=> t.id === taskId)
        if(!task) return
```

```
        task.completedAt = task.completedAt ? undefined : new Date()
    }



    return {
        // Properties
        projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),
        noProjects: computed(()=>projects.value.length == 0),
        projectsWithCompletion: computed(()=>{
            return projects.value.map(project=>{
                const total = project.tasks.length

                const completed = project.tasks.filter(t=> t.completedAt).length
                const completion = total === 0 ? 0 : (completed/total) * 100

                return{
                    id: project.id,
                    name: project.name,
                    taskCount: total,
                    completion: completion
                }
            })
        }),

        // Actions
        addProject,
        addTaskToProject,
        toggleTask
    }
})
```

- En ProjecstView.vue (el plano general, donde está la barra de progress)
- En lugar del projectList en el for usaré el projectWithCompletion
- Uso el .taskCount en lugar de tasks.length

```
<template>
<div class="overflow-x-auto w-full">
  <table class="table">
    <!-- head -->
    <thead>
      <tr>
        <th></th>
        <th>Proyectos</th>
        <th>Tareas</th>
        <th>Avances</th>
      </tr>
    </thead>
    <tbody>
      <!-- row 1 -->
```

```
      <tr v-for="(project,index) in
projectStore.projectsWithCompletion" :key="project.id">
        <th>{{ index+1 }}</th>
        <td>{{ project.name }}</td>
        <td>{{ project.taskCount }}</td>
        <td>
          <progress class="progress progress-primary w-56" :value="project.completion"
max="100"></progress>
        </td>
      </tr>
    </tbody>
  </table>
 </div>

  <input-modal
  :open="openModal"
  @close="openModal = false"
  @value="projectStore.addProject"
  title="Añade aquí tu proyecto"
  subtitle="Desde aquí puedes añadir tu proyecto"
  placeholder="Nombre del proyecto"
  />

  <custom-modal :open="customOpenModal">
    <template #header>
      <h1>Esto es el header</h1>
    </template>
  </custom-modal>

  <fab-button @click="openModal = true">
    <add-circle />
  </fab-button>

  <fab-button @click="customOpenModal = true" position="bottom-left">
    <add-circle />
  </fab-button>

</template>

<script setup lang="ts">
import CustomModal from '@/modules/common/components/CustomModal.vue';

import FabButton from '@/modules/common/components/FabButton.vue';
import InputModal from '@/modules/common/components/InputModal.vue';
import AddCircle from '@/modules/common/icons/AddCircle.vue';
import { ref } from 'vue';
import { useProjectsStore } from '../stores/projects.store';

const openModal= ref(false)
const customOpenModal=ref(false)

const projectStore = useProjectsStore()

const onNewValue=(projectName: string)=>{
  projectStore.projectList.push({
    id: '3',
```

```
    name: projectName,
    tasks: []
  })
}
</script>
```

- Si quiero mostrar el porcentaje puedo redondear el completion con un Math.round
- projects.store.ts

```
return {
        // Properties
        projects,

        // Getters (propiedades computadas)
        projectList: computed(()=> [...projects.value]),
        noProjects: computed(()=>projects.value.length == 0),
        projectsWithCompletion: computed(()=>{
            return projects.value.map(project=>{
                const total = project.tasks.length

                const completed = project.tasks.filter(t=> t.completedAt).length
                const completion = total === 0 ? 0 : (completed/total) * 100

                return{
                    id: project.id,
                    name: project.name,
                    taskCount: total,
                    completion: Math.round(completion)
                }
            })
        }),
}
```

- Y usar el project.completion en el td de la barra de progreso en ProjectsView.vue

```
<td>
    <progress class="progress progress-primary w-56" :value="project.completion"
max="100"></progress>
    {{ project.completion }}
</td>
```

- El ProjectView.vue queda así

```
<template>
    <div class="w-full">
        <section class="m-2">
            <bread-crumbs :name="project?.name ?? 'no-name'"></bread-crumbs>
        </section>
        <section class="m-2">
```

```
                <div class="overflow-x-auto">
            <table class="table">
                <!-- head -->
                <thead>
                <tr>
                    <th class="w-14">Completada</th>
                    <th>Tarea</th>
                    <th>Completada en</th>
                </tr>
                </thead>
                <tbody>
                <tr v-for="task in project?.tasks" :key="task.id" class="hover:bg-
base-300">
                    <th>
                        <input type="checkbox"
                            :checked="!!task.completedAt"
                            class="checkbox checkbox-primary"
                            @change="projectStore.toggleTask(project!.id, task.id)"
                        >
                    </th>
                    <td>{{ task.name }}</td>
                    <td>{{ task.completedAt }}</td>
                </tr>
                <tr class="hover:bg-base-300">
                    <th></th>
                    <td>
                        <input type="text"
                        class="input input-primary w-full opacity-60 tranisition-all
hover:opacity-100 focus:opacity-100"
                        placeholder="Nueva tarea"
                        v-model="newTask"
                        @keypress.enter="addTask"
                        >
                    </td>
                </tr>
                </tbody>
            </table>
            </div>
        </section>
    </div>
</template>

<script setup lang="ts">
import BreadCrumbs from '@/modules/common/components/BreadCrumbs.vue';
import { useProjectsStore } from '../stores/projects.store';
import { computed, ref, watch } from 'vue';
import type { Project } from '../interface/project.interface';
import { useRouter } from 'vue-router';

interface Props{
    id: string
}

const router = useRouter()
const props = defineProps<Props>()
const projectStore = useProjectsStore()
```

```
const project = ref<Project|undefined>()
const newTask = ref('')

watch(()=> props.id, ()=>{
    project.value= projectStore.projectList.find((x) => x.id === props.id);
    if(!project.value){
        router.replace('/')
    }
}, {
    immediate: true
})

const addTask=()=>{
    if(!project.value) return
    projectStore.addTaskToProject(project.value.id, newTask.value)
    newTask.value=""
}
</script>
```

- El SideMenu.vue queda así (añado un router-link al h2 de Proyectos para poder clicar para ir al Home)

```
<template>
  <aside class="bg-base-200 w-72 min-h-screen">
    <h2 class="text-lg font-bold mx-4">
        <router-link to="/">Proyectos</router-link>
    </h2>
    <p v-if="projectsStore.noProjects" class="text-sm text-gray-500 mx-4">No hay
proyectos</p>
    <!--Menu-->
     <ul v-else class="menu rounded-box w-56">
        <li v-for="project in projectsStore.projectList" :key="project.id">
        <template v-if="project.tasks.length > 0">
         <details>
            <summary>
                <router-link :to="`/project/${project.id}`">{{ project.name }}</router-
link>
            </summary>
            <ul>
                <li v-for="task in project.tasks">
                  <router-link :to="`/project/${project.id}`">{{ task.name }}</router-
link>
                </li>
            </ul>
         </details>
        </template>

        <template v-else>
            <summary>
                <router-link :to="`/project/${project.id}`">{{ project.name }}</router-
link>
            </summary>
        </template>
```

```
        </li>
        </ul>
    </aside>
</template>

<script setup lang="ts">
import { useProjectsStore } from '../stores/projects.store';

    const projectsStore = useProjectsStore()
</script>
```

• El router queda así

```
import ProjectsLayout from '@/modules/projects/layouts/ProjectsLayout.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      redirect: {name: 'projects'},
      component: ProjectsLayout,
      children:[
        {
          path:'projects',
          name: 'projects',
          component: ()=>import('@/modules/projects/views/ProjectsView.vue')
        },
        {
          path:'project/:id',
          props: true,
          name: 'project',
          component: ()=>import('@/modules/projects/views/ProjectView.vue')
        }
      ]
    }
  ],
})

export default router
```

- Faltaría poder editar las tareas las tareas clicando encima de la tarea

12 Vue Herrera - Administración de productos

- Cuando refresco el navegador, Tanstack Query carga la página 1 y la 2, para dar la impresión de ser la web más rápida
- Cuando estoy en la 2 precarga la 3
- Crearemos un dashboard administrativo y autenticación basada en JWT

Inicio

> npm create vue@latest

- name: admins-shop
- TypeScript: si
- JSX: no
- Vue Router: si
- Pinia: si
- Vitest: si
- E2E: no
- ESLint: si
- Prettier: si

- Configuramos tailwind

> npm install tailwindcss @tailwindcss/vite

- vite.config.ts

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
    tailwindcss(),
  ],
  resolve: {
    alias: {
```

```
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
  },
})
```

- En styles.css

```
@import "tailwindcss";
```

- Debo importar styles.css en el main

```
import './assets/styles.css'
import { createApp } from 'vue'
import { createPinia } from 'pinia'

import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(createPinia())
app.use(router)

app.mount('#app')
```

- Coloco el RouterView en el App.vue

```
<template>
    <h1 class="text-3xl text-red-500">Hola mundo</h1>
    <router-view />
</template>
```

ShopLayout - Diseño de la tienda

- src/modules/auth para todo lo relacionado con autenticación
- src/modules/admin para lo relacionado con el panel administrativo
- src/modules/common para módulos que no dependen de otros
- src/modules/products todo lo relacionado a productos
- src/modules/shop lo que vamos a usar para mostrar de cara a los usuarios que entren a la aplicación
- Creo la subcarpeta modules/shop/layouts/ShopLayout.ts
- Usaremos TailwindComponents

> https://www.creative-tim.com/twcomponents

- Usaremos este componente

> https://www.creative-tim.com/twcomponents/component/sopping-cart

- Borro el body, corrijo algunos errores (hay una etiqueta de cierre del footer duplicada, y un div sin cerrar)
- Dejo solo un article dentro del ProductsList

```
<template>
    <!-- component -->
<!-- Create By Joker Banny -->
  <!-- Header Navbar -->
<nav class="fixed top-0 left-0 z-20 w-full border-b border-gray-200 bg-white py-2.5
px-6 sm:px-4">
  <div class="container mx-auto flex max-w-6xl flex-wrap items-center justify-between">
    <a href="#" class="flex items-center">
      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-
width="1.5" stroke="currentColor" class="mr-3 h-6 text-blue-500 sm:h-9">
        <path stroke-linecap="round" stroke-linejoin="round" d="M21 7.5l-9-5.25L3
7.5m18 0l-9 5.25m9-5.25v9l-9 5.25M3 7.5l9 5.25M3 7.5v9l9 5.25m0-9v9" />
      </svg>

      <span class="self-center whitespace-nowrap text-xl font-semibold">Termcode</span>
    </a>
    <div class="mt-2 sm:mt-0 sm:flex md:order-2">
      <!-- Login Button -->
      <button type="button" class="rounde mr-3 hidden border border-blue-700 py-1.5
px-6 text-center text-sm font-medium text-blue-700 focus:outline-none focus:ring-4
focus:ring-blue-300 md:inline-block rounded-lg">Login</button>
      <button type="button" class="rounde mr-3 hidden bg-blue-700 py-1.5 px-6 text-
center text-sm font-medium text-white hover:bg-blue-800 focus:outline-none focus:ring-4
focus:ring-blue-300 md:mr-0 md:inline-block rounded-lg">Register</button>
      <!-- Register Button -->
      <button data-collapse-toggle="navbar-sticky" type="button" class="inline-flex
items-center rounded-lg p-2 text-sm text-gray-500 hover:bg-gray-100 focus:outline-none
focus:ring-2 focus:ring-gray-200 md:hidden" aria-controls="navbar-sticky" aria-
expanded="false">
        <span class="sr-only">Open main menu</span>
        <svg class="h-6 w-6" aria-hidden="true" fill="currentColor" viewBox="0 0 20
20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M3 5a1 1 0
011-1h12a1 1 0 110 2H4a1 1 0 01-1-1zM3 10a1 1 0 011-1h12a1 1 0 110 2H4a1 1 0 01-1-1zM3
15a1 1 0 011-1h12a1 1 0 110 2H4a1 1 0 01-1-1z" clip-rule="evenodd"></path></svg>
      </button>
    </div>
    <div class="hidden w-full items-center justify-between md:order-1 md:flex md:w-
auto" id="navbar-sticky">
      <ul class="mt-4 flex flex-col rounded-lg border border-gray-100 bg-gray-50 p-4
md:mt-0 md:flex-row md:space-x-8 md:border-0 md:bg-white md:text-sm md:font-medium">
        <li>
          <a href="#" class="block rounded bg-blue-700 py-2 pl-3 pr-4 text-white md:bg-
transparent md:p-0 md:text-blue-700" aria-current="page">Home</a>
```

```
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">About</a>
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">Services</a>
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">Contact</a>
        </li>
      </ul>
    </div>
  </div>
</nav>


<!-- Title -->
<div class="pt-32  bg-white">
<h1 class="text-center text-2xl font-bold text-gray-800">All Products</h1>
</div>

<!-- Tab Menu -->
<div class="flex flex-wrap items-center  overflow-x-auto overflow-y-hidden py-10
justify-center   bg-white text-gray-800">
    <a rel="noopener noreferrer" href="#" class="flex items-center flex-shrink-0 px-5
py-3 space-x-2text-gray-600">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"
class="w-4 h-4">
            <path d="M19 21l-7-5-7 5V5a2 2 0 0 1 2-2h10a2 2 0 0 1 2 2z"></path>
        </svg>
        <span>Architecto</span>
    </a>
    <a rel="noopener noreferrer" href="#" class="flex items-center flex-shrink-0 px-5
py-3 space-x-2 rounded-t-lg text-gray-900">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"
class="w-4 h-4">
            <path d="M2 3h6a4 4 0 0 1 4 4v14a3 3 0 0 0-3-3H2z"></path>
            <path d="M22 3h-6a4 4 0 0 0-4 4v14a3 3 0 0 1 3-3h7z"></path>
        </svg>
        <span>Corrupti</span>
    </a>
    <a rel="noopener noreferrer" href="#" class="flex items-center flex-shrink-0 px-5
py-3 space-x-2  text-gray-600">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"
class="w-4 h-4">
            <polygon
points="12 2 15.09 8.26 22 9.27 17 14.14 18.18 21.02 12 17.77 5.82 21.02 7 14.14 2 9.27
8.91 8.26 12 2"></polygon>
        </svg>
        <span>Excepturi</span>
```

```
    </a>
    <a rel="noopener noreferrer" href="#" class="flex items-center flex-shrink-0 px-5
py-3 space-x-2  text-gray-600">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"
class="w-4 h-4">
            <circle cx="12" cy="12" r="10"></circle>
            <polygon points="16.24 7.76 14.12 14.12 7.76 16.24 9.88 9.88 16.24 7.76"></
polygon>
        </svg>
        <span>Consectetur</span>
    </a>
</div>

<!-- Product List -->
<section class="py-10 bg-gray-100">
  <div class="mx-auto grid max-w-6xl  grid-cols-1 gap-6 p-6 sm:grid-cols-2 md:grid-
cols-3 lg:grid-cols-4">
    <article class="rounded-xl bg-white p-3 shadow-lg hover:shadow-xl hover:transform
hover:scale-105 duration-300 ">
      <a href="#">
        <div class="relative flex items-end overflow-hidden rounded-xl">
          <img src="https://images.unsplash.com/flagged/photo-1556637640-2c80d3201be8?
ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w
=1170&q=80" alt="Hotel Photo" />
          <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4 py-1.5
text-white duration-100 hover:bg-blue-600">
            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
              <path stroke-linecap="round" stroke-linejoin="round"
d="M2.25 3h1.386c.51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
            </svg>

            <button class="text-sm">Add to cart</button>
          </div>
        </div>

        <div class="mt-1 p-2">
          <h2 class="text-slate-700">The Hilton Hotel</h2>
          <p class="mt-1 text-sm text-slate-400">Lisbon, Portugal</p>

          <div class="mt-3 flex items-end justify-between">
              <p class="text-lg font-bold text-blue-500">$450</p>
            <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4
py-1.5 text-white duration-100 hover:bg-blue-600">
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
                <path stroke-linecap="round" stroke-linejoin="round" d="M2.25 3h1.386c.
51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
              </svg>
```

```
              <button class="text-sm">Add to cart</button>
            </div>
          </div>
        </div>
      </a>
    </article>
    </div>
</section>

<!-- Footer -->
<footer class="py-6  bg-gray-200 text-gray-900">
    <div
class="container px-6 mx-auto space-y-6 divide-y divide-gray-400 md:space-y-12 divide-
opacity-50">
        <div class="grid justify-center  lg:justify-between">
            <div class="flex flex-col self-center text-sm text-center md:block lg:col-
start-1 md:space-x-6">
                <span>Copy rgight © 2023 by codemix team </span>
                <a rel="noopener noreferrer" href="#">
                    <span>Privacy policy</span>
                </a>
                <a rel="noopener noreferrer" href="#">
                    <span>Terms of service</span>
                </a>
            </div>
            <div class="flex justify-center pt-4 space-x-4 lg:pt-0 lg:col-end-13">
                <a rel="noopener noreferrer" href="#" title="Email" class="flex items-
center justify-center w-10 h-10 rounded-full bg-blue-500 hover:bg-blue-600 duration-150
text-gray-50">
                    <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20"
fill="currentColor" class="w-5 h-5">
                        <path
d="M2.003 5.884L10 9.882l7.997-3.998A2 2 0 0016 4H4a2 2 0 00-1.997 1.884z"></path>
                        <path d="M18 8.118l-8 4-8-4V14a2 2 0 002 2h12a2 2 0
002-2V8.118z"></path>
                    </svg>
                </a>
                <a rel="noopener noreferrer" href="#" title="Twitter" class="flex
items-center justify-center w-10 h-10 rounded-full bg-blue-500 hover:bg-blue-600
duration-150 text-gray-50">
                    <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 50 50"
fill="currentColor" class="w-5 h-5">
                        <path d="M 50.0625 10.4375 C 48.214844 11.257813 46.234375
11.808594 44.152344 12.058594 C 46.277344 10.785156 47.910156 8.769531 48.675781
6.371094 C 46.691406 7.546875 44.484375 8.402344 42.144531 8.863281 C 40.269531
6.863281 37.597656 5.617188 34.640625 5.617188 C 28.960938 5.617188 24.355469 10.21875
24.355469 15.898438 C 24.355469 16.703125 24.449219 17.488281 24.625 18.242188 C
16.078125 17.8125 8.503906 13.71875 3.429688 7.496094 C 2.542969 9.019531 2.039063
10.785156 2.039063 12.667969 C 2.039063 16.234375 3.851563 19.382813 6.613281 21.230469
C 4.925781 21.175781 3.339844 20.710938 1.953125 19.941406 C 1.953125 19.984375
1.953125 20.027344 1.953125 20.070313 C 1.953125 25.054688 5.5 29.207031 10.199219
30.15625 C 9.339844 30.390625 8.429688 30.515625 7.492188 30.515625 C 6.828125
30.515625 6.183594 30.453125 5.554688 30.328125 C 6.867188 34.410156 10.664063
37.390625 15.160156 37.472656 C 11.644531 40.230469 7.210938 41.871094 2.390625
41.871094 C 1.558594 41.871094 0.742188 41.824219 -0.0585938 41.726563 C 4.488281
```

```
44.648438 9.894531 46.347656 15.703125 46.347656 C 34.617188 46.347656 44.960938
30.679688 44.960938 17.09375 C 44.960938 16.648438 44.949219 16.199219 44.933594
15.761719 C 46.941406 14.3125 48.683594 12.5 50.0625 10.4375 Z"></path>
                    </svg>
                </a>
                <a rel="noopener noreferrer" href="#" title="GitHub" class="flex items-
center justify-center w-10 h-10 rounded-full bg-blue-500 hover:bg-blue-600 duration-150
text-gray-50">
                    <svg xmlns="http://www.w3.org/2000/svg" fill="currentColor"
viewBox="0 0 24 24" class="w-5 h-5">
                        <path
d="M10.9,2.1c-4.6,0.5-8.3,4.2-8.8,8.7c-0.5,4.7,2.2,8.9,6.3,10.5C8.7,21.4,9,21.2,9,20.8v-1.
c-1.4,0-2-1.2-2.1-1.9c-0.1-0.4-0.3-0.7-0.6-1C5.1,16.3,5,16.3,5,16.2C5,16,5.3,16,5.4,16c
0.6,0,1.1,0.7,1.3,1c0.5,0.8,1.1,1,1.4,1
c0.4,0,0.7-0.1,0.9-0.2c0.1-0.7,0.4-1.4,1-1.8c-2.3-0.5-4-1.8-4-4c0-1.1,0.5-2.2,1.2-3C7.1
,8.8,7,8.3,7,7.6C7,7.2,7,6.6,7.3,6
c0,0,1.4,0,2.8,1.3C10.6,7.1,11.3,7,12,7s1.4,0.1,2,0.3C15.3,6,16.8,6,16.8,6C17,6.6,17,7.
2,17,7.6c0,0.8-0.1,1.2-0.2,1.4
c0.7,0.8,1.2,1.8,1.2,3c0,2.2-1.7,3.5-4,4c0.6,0.5,1,1.4,1,2.3v2.6c0,0.3,0.3,0.6,0.7,0.5c
3.7-1.5,6.3-5.1,6.3-9.3 C22,6.1,16.9,1.4,10.9,2.1z"></path>
                    </svg>
                </a>
            </div>
        </div>
    </div>
</footer>
</template>
```

• Coloco el componente en el router

```
import ShopLayout from '@/modules/shop/layouts/ShopLayout.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'shop',
      component: ShopLayout
    }
  ],
})

export default router
```

• Vamos a querer separar el Layout en diferentes componentes (el navbar, por ejemplo)

• modules/shop/components/TopMenu.vue

• Compacto el nav y lo pego dentro de un template en el TopMenu.vue

```
<template>
  <nav class="fixed top-0 left-0 z-20 w-full border-b border-gray-200 bg-white py-2.5
px-6 sm:px-4">
  <div class="container mx-auto flex max-w-6xl flex-wrap items-center justify-between">
    <a href="#" class="flex items-center">
      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-
width="1.5" stroke="currentColor" class="mr-3 h-6 text-blue-500 sm:h-9">
        <path stroke-linecap="round" stroke-linejoin="round" d="M21 7.5l-9-5.25L3
7.5m18 0l-9 5.25m9-5.25v9l-9 5.25M3 7.5l9 5.25M3 7.5v9l9 5.25m0-9v9" />
      </svg>

      <span class="self-center whitespace-nowrap text-xl font-semibold">Termcode</span>
    </a>
    <div class="mt-2 sm:mt-0 sm:flex md:order-2">
      <!-- Login Button -->
      <button type="button" class="rounde mr-3 hidden border border-blue-700 py-1.5
px-6 text-center text-sm font-medium text-blue-700 focus:outline-none focus:ring-4
focus:ring-blue-300 md:inline-block rounded-lg">Login</button>
      <button type="button" class="rounde mr-3 hidden bg-blue-700 py-1.5 px-6 text-
center text-sm font-medium text-white hover:bg-blue-800 focus:outline-none focus:ring-4
focus:ring-blue-300 md:mr-0 md:inline-block rounded-lg">Register</button>
      <!-- Register Button -->
      <button data-collapse-toggle="navbar-sticky" type="button" class="inline-flex
items-center rounded-lg p-2 text-sm text-gray-500 hover:bg-gray-100 focus:outline-none
focus:ring-2 focus:ring-gray-200 md:hidden" aria-controls="navbar-sticky" aria-
expanded="false">
        <span class="sr-only">Open main menu</span>
        <svg class="h-6 w-6" aria-hidden="true" fill="currentColor" viewBox="0 0 20
20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M3 5a1 1 0
011-1h12a1 1 0 110 2H4a1 1 0 01-1-1zM3 10a1 1 0 011-1h12a1 1 0 110 2H4a1 1 0 01-1-1zM3
15a1 1 0 011-1h12a1 1 0 110 2H4a1 1 0 01-1-1z" clip-rule="evenodd"></path></svg>
      </button>
    </div>
    <div class="hidden w-full items-center justify-between md:order-1 md:flex md:w-
auto" id="navbar-sticky">
      <ul class="mt-4 flex flex-col rounded-lg border border-gray-100 bg-gray-50 p-4
md:mt-0 md:flex-row md:space-x-8 md:border-0 md:bg-white md:text-sm md:font-medium">
        <li>
          <a href="#" class="block rounded bg-blue-700 py-2 pl-3 pr-4 text-white md:bg-
transparent md:p-0 md:text-blue-700" aria-current="page">Home</a>
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">About</a>
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">Services</a>
        </li>
        <li>
          <a href="#" class="block rounded py-2 pl-3 pr-4 text-gray-700 hover:bg-
gray-100 md:p-0 md:hover:bg-transparent md:hover:text-blue-700">Contact</a>
        </li>
      </ul>
    </div>
  </div>
```

```
</nav>
</template>
```

- Lo renderizo en el layout
- Hago lo mismo con el footer
- Añado el año actual en el copyright

```
<span>Copyright © {{ new Date().getFullYear() }} by Meikakuzen team </span>
```

- El TabMenu y el ProductList debería de ser nuestra pantalla del home (el título también)
- Los compacto, los copio y los pego en el HomeView.vue
- modules/shop/views/HomeView.vue
- Uso el RouterView en el ShopLayout.vue para mostrar el HomeView

```
<template>
<top-menu />

<router-view />

<custom-footer />
</template>


<script setup lang="ts">
import CustomFooter from '../components/CustomFooter.vue';
import TopMenu from '../components/TopMenu.vue';

</script>
```

- En el router uso children
- La ruta tiene el path vacío para que se apunte al shop

```
import ShopLayout from '@/modules/shop/layouts/ShopLayout.vue'
import HomeView from '@/modules/shop/views/HomeView.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'shop',
      component: ShopLayout,
      children: [
        {
          path: '',
          name: 'home',
```

```
          component: ()=>import('@/modules/shop/views/HomeView.vue')
        }
      ]
    }
  ],
})

export default router
```

- Cada componente que se muestra en el home (los articulos) también deben ser independientes

Levantar nuestro backend

- URL del backend

> https://github.com/Klerith/nest-teslo-shop/tree/complete-backend

- Descargo el código, lo pego en la carpeta backend dentro de la aplicación, hago npm i
- Falta la DB. Copio el .env.template en un .env, abro Docker y uso docker compose up -d en la terminal
- Llamo al seed para recrear la DB, apunto a localhost:3000/api/seed
- Hago una prueba con un GET a localhost:3000/api/products en POSTMAN
- Para ver la DB puedo usar Table Plus, creo una nueva DB de Postgres, con el puerto definido en el archivo de Docker (5432)
- User es postgres, database es TesloDB
- Los passwords de los usuarios test1 y test2 es Abc123
- El .env

```
STAGE=dev

DB_PASSWORD=MySecr3tPassWord@as2
DB_NAME=TesloDB
DB_HOST=localhost
DB_PORT=5432
DB_USERNAME=postgres

PORT=3000
HOST_API=http://localhost:3000/api

JWT_SECRET=Est3EsMISE3Dsecreto32s
```

Variables de entorno - Axios

- Creo src/api/tesloApi-ts

- Lo creo en src porque todos los módulos pasan por este mismo api. Si tuviera uno para store, lo crearía en el módulo store
- Instalo axios con npm
- Exporto tesloApi como un objeto porque haremos uso de interceptores
- tesloApi.ts

```
import axios from 'axios'


const tesloApi = axios.create({
    baseURL: import.meta.env.VITE_TESLO_API_URL
})

//Interceptors
export {tesloApi}
```

- Creo el .env en el frontend
- Hay que ponerle VITE delante a la variable para poder usarla

```
VITE_STAGE=dev

VITE_TESLO_API_URL=http://localhost:3000/api
```

- Es conveniente dejar un README.md

```
# Admin Shop


- Pasos para dev

1. Clonar el repositorio
2. Instalar dependencias
3. Crear un archivo .env basado en el .env.template
4. Correr el proyecto con `npm run dev`
```

Obtener productis paginados

- Primero los extraemos, luego los conectamos a un gestor de estado, manejaremos caché, etc
- Creo en products/actions/get-products.action.ts
- Creo un archivo de barril para exportar las diferentes actios
- get-products.action.ts

```

```

```
import { tesloApi } from "@/api/tesloApi"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get('/products')
        console.log(data)
        return (data)
    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

• En el script de shop/views/Homeview.vue

```
<script lang="ts" setup>
import { getProductsAction } from '@/modules/products/actions/get-products.action';


  getProductsAction()
</script>
```

• En la consola del navegador me devuelve un array con 10 objetos, cada objeto luce así

```
description:
"Introducing the Tesla Chill Collection. The Women's Chill Half Zip Cropped Hoodie has
a premium, soft fleece exterior and cropped silhouette for comfort in everyday
lifestyle. The hoodie features an elastic hem that gathers at the waist, subtle
thermoplastic polyurethane Tesla logos along the hood and on the sleeve, a double layer
single seam hood and a custom ring zipper pull. Made from 60% cotton and 40% recycled
polyester."
gender: "women"
id: "0b2d7456-2f7f-4d63-adcf-d0a7be0bda46"
images: (2) ['1740226-00-A_0_2000.jpg', '1740226-00-A_1.jpg']
price: 130
sizes: (4) ['XS', 'S', 'M', 'XXL']
slug: "women_chill_half_zip_cropped_hoodie"
stock: 10
tags: ['hoodie']
title: "Women's Chill Half Zip Cropped Hoodie"
user:{id: '119e2450-1428-4b16-bec8-a2d28ba199fe', email: 'test1@google.com', fullName:
'Test One', isActive: true, roles: Array(1)}
```

• Con esta estructura, si trabajaramos con un backend de terceros, podríamos pensar en crear un mapper (traducir este objeto a nuestra estructura de datos en la aplicación) para evitar errores con cambios inesperados en el backend

• Quiero tipar la respuesta, por lo que la copio de POSTMAN y uso PasteJSONasCode

• En products/interfaces/product.interface.ts

• Hago algunas correcciones

```
export interface Product{
    id:          string;
    title:       string;
    price:       number;
    description: string;
    slug:        string;
    stock:       number;
    sizes:       Size[];
    gender:      Gender;
    tags:        string[]; //arreglo de strings
    images:      string[];
    user:        User;
}

export enum Gender {
    Kid = "kid",
    Men = "men",
    Women = "women",
}

export enum Size {
    L = "L",
    M = "M",
    S = "S",
    Xl = "XL",
    Xs = "XS",
    Xxl = "XXL",
}

export enum Tag {
    Hoodie = "hoodie",
    Shirt = "shirt",
    Sweatshirt = "sweatshirt",
}

//este User debe de estar en el módulo de auth
export interface User {
    id:       string;
    email:    string;
    fullName: string;
    isActive: boolean;
    roles:   string[];
}

{/*export enum Email {
    Test1GoogleCOM = "test1@google.com",
}

export enum FullName {
    TestOne = "Test One",
}

export enum Role {
```

```
    Admin = "admin",
}*/}
```

- auth/interfaces/user.interface.ts

```
export interface User {
    id:       string;
    email:    string;
    fullName: string;
    isActive: boolean;
    roles:   string[];
}
```

- Ahora ya puedo tipar la respuesta de getProductsAction

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>('/products')
        console.log(data[0]?.gender) //ya tengo el tipado
        return (data)
    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

- Puedo mandarle params a la url de products, por ejemplo que me de dos registros y empiece despues de los siguientes 10 registros

> localhost:3000/api/products?limit=2&offset=10

- Uso template literal para incluir el limit y el offset

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)
        console.log(data[0]?.gender)
        return (data)
    } catch (error) {
        throw new Error('Error getting products')
```

```
    }
}
```

- Tener la petición directamente en el script del HomeView no luce bien
- Además quiero manejarlo en caché

TanStack Query - useQuery

- Llamando a la action desde el script de HomeView hacemos la petición http tan pronto entramos al componente
- Instalo TanStack Query (también es un gestor de estado para peticiones http)

> npm i @tanstack/vue-query

- Hay que hacer uso del VueQueryPlugin
- main.ts

```
import './assets/styles.css'
import { createApp } from 'vue'
import { createPinia } from 'pinia'

import App from './App.vue'
import router from './router'
import { VueQueryPlugin } from '@tanstack/vue-query'

const app = createApp(App)

app.use(createPinia())
app.use(VueQueryPlugin)
app.use(router)

app.mount('#app')
```

- Ya se pueden empezar a hacer peticiones http y usar useQuery
- Hay otra instalación interesante para tanstack que son las devtools

> npm i @tanstack/vue-query-devtools

- Se necesita colocar este componente en el nivel más alto de la aplicación (App.vue)

```
<template>
<router-view />
<vue-query-devtools />
</template>

<script lang="ts" setup>
```

```
  import { VueQueryDevtools} from '@tanstack/vue-query-devtools';
</script>
```

- Esto es todo. Abajo, en el bottom right de la pantalla del navegador aparece como un icono de una playa tropical
- Son las devtools
- En modules/shop/views/HomeView.vue no ocupo hacer la petición http de esta forma (llamando simplemente a la action)
- Usemos useQery, en la propiedad queryKey (es un arreglo) le paso como primer argumento 'products' y le indico la página 1
- Lo que pongo entre las llaves del queryKey crea una llave, un identificador único
- Este queryKey puedo usarlo como caché, puedo ponerle un tiempo de duración
- El queryFn es la función que voy a llamar cuando se llame a este composable
- Pudo desestructurar varias cosas del useQuery, la data que renombro a products y el isLoading, entre otros
- En el script del HomeView.vue

```
<script lang="ts" setup>
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import { useQuery } from '@tanstack/vue-query';

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page:1}],
  queryFn: ()=> getProductsAction()
})

</script>
```

- Puedo colocar estos products en un div dentro del template

```
<div>
  loading: {{ isLoading }}
</div>
<div>
  {{ products }}
</div>
```

- Aparece el loading en false y un arreglo con los objetos de los productos
- Si miro las devtools de TansTack podemos acceder a la data que se encuentra en el caché de la llave
- La llave es

```
```

```
[
  "products",
  {
    "page":1
  }
]
```

- La data es el arreglo con los products
- Si alguien vuelve a hacer una petición a este queryKey y este queryKey ya existía en el cliente (el VueQueryPlugin del main) va a brindar esta info primero en caché y luego la actualiza
- Puedo usar staleTime para indicar cuanto tiempo quiero que dure n caché

```
const {data:products, isLoading} = useQuery({
  queryKey:['products', {page:1}],
  queryFn: ()=> getProductsAction(),
  staleTime: 1000 *60 //60 segundos
})
```

Obtener el url de las imágenes

- Ya casi tenemos todo listo para empezar a graficar los productos en pantalla
- En el objeto Product las imágenes están en un array llamado images
- Lo que se ve es “175165-81667_1.jpg”, pero necesitamos todo el url completo
- Podemos usar una función que reconstruya el url cada vez que muestra un producto o reconstruimos el url en la petición y lo mantenemos
- Para ver una imagen están en localhost:3000/api/files/product/082732873_1.jpg
- Este es el URL que necesito, pero en producción el dominio va a variar, no será localhost, por lo que usaremos una variable de entorno
- Creo una nueva acción
- En products/action/get-product-image.action.ts

```
export const getProductImageAction =(imageName: string): string=> {

    return imageName.includes('http')
    ? imageName
    : `${import.meta.env.VITE_TESLO_API_URL}/files/product/${imageName}`
}
```

- Creo un archivo de barril para exportar las actions
- index.ts

```
export * from './get-products.action'
export * from './get-product-image.action'
```

- En el get-products.action uso esta función
- Esparzo el product con el spread y en el map le paso la función pro referencia

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)

        return data.map(product=>{
            return{
                ...product,
                images: product.images.map(getProductImageAction)
            }
        })
    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

- Se puede formatear sin el return usando el return implicito con los paréntesis

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)

        return data.map(product=>({
            ...product,
            images: product.images.map(getProductImageAction)}))

    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

- Puedo usar las devtools de Tanstack para ver que la url está formateada correctamente

Mostrar productos en pantalla

- En modules/shop/views/HomeView.vue tengo el article que será el ProductCard
- Vamos a dividir la vista de productos en 2 componentes, ProductList y ProductCard
- En el ProductList vamos a recibir un arreglo de productos y en el ProductCard el producto
- No importa mucho como lo hagamos porque luego lo vamos a administrar a través de un panel administrativo
- En modules/products/components/ProductList.vue pego todo el section que tengo en HomeView.vue
- Borrando todos los article de dentro me queda esto

```
<template>
  <section class="py-10 bg-gray-100">
    <div class="mx-auto grid max-w-6xl  grid-cols-1 gap-6 p-6 sm:grid-cols-2 md:grid-
cols-3 lg:grid-cols-4">
      <!--Aqui va el ProductCard (el article con la data)-->
    </div>
  </section>
</template>
```

- En el ProductCard tengo el article sin tunear que había dentro del ProductList.vue

```
<template>
    <article class="rounded-xl bg-white p-3 shadow-lg hover:shadow-xl hover:transform
hover:scale-105 duration-300 ">
      <a href="#">
        <div class="relative flex items-end overflow-hidden rounded-xl">
          <img src="https://images.unsplash.com/photo-1520256862855-398228c41684?
ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w
=1169&q=80" alt="Hotel Photo" />
          <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4 py-1.5
text-white duration-100 hover:bg-blue-600">
            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
              <path stroke-linecap="round" stroke-linejoin="round"
d="M2.25 3h1.386c.51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
            </svg>

            <button class="text-sm">Add to cart</button>
          </div>
        </div>

        <div class="mt-1 p-2">
          <h2 class="text-slate-700">The Hilton Hotel</h2>
          <p class="mt-1 text-sm text-slate-400">Lisbon, Portugal</p>
```

```
          <div class="mt-3 flex items-end justify-between">
              <p class="text-lg font-bold text-blue-500">$450</p>
            <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4
py-1.5 text-white duration-100 hover:bg-blue-600">
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
                <path stroke-linecap="round" stroke-linejoin="round" d="M2.25 3h1.386c.
51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
              </svg>

              <button class="text-sm">Add to cart</button>
            </div>
          </div>
        </div>
      </a>
    </article>
</template>
```

- En el ProductList.vue renderizo el ProductCard, y en el HomeView.vue renderizo el ProductList
- Para manejar la data por las props, se recomienda empezar por el componente que realmente necesita la info
- Por el nieto, luego el padre y luego el abuelo
- Creo las interfaces y uso defineProps
- ProductList.vue

```
<template>
    <section class="py-10 bg-gray-100">
  <div class="mx-auto grid max-w-6xl  grid-cols-1 gap-6 p-6 sm:grid-cols-2 md:grid-
cols-3 lg:grid-cols-4">
        <ProductCard
        v-for="product in products" :key="product.id"
        :product="product"
        />
    </div>
</section>
</template>


<script setup lang="ts">
import type { Product } from '../interfaces/products-response.interface';
import ProductCard from './ProductCard.vue';

interface Props{
    products: Product[]
}

defineProps<Props>()
```

```
</script>
```

- En el HomeView.vue donde muestro el ProductList, si le paso el arreglo de products me da error, **porque en algún punto del tiempo es undefined**
- Podemos crear un loading usando v-if y v-else para los products, negando products en el v-if

```
<template>
{...code}

<div v-if="!products" class="text-center h-[500px]">
  <h1 class="text-xl">Cargando productos...</h1>
  <p>Espere por favor</p>
</div>
<!-- Product List -->
  <ProductList v-else :products="products"/>
</template>


<script lang="ts" setup>
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery } from '@tanstack/vue-query';

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page:1}],
  queryFn: ()=> getProductsAction(),
})

</script>
```

- Ahora si carga los 10 resultados en duro en pantalla
- Falta la carpinteria del ProductCard, ya tengo el product listo para consumir
- ProductCard.vue

```
<template>
    <article class="rounded-xl bg-white p-3 shadow-lg hover:shadow-xl hover:transform
hover:scale-105 duration-300 ">
      <a href="#">
        <div class="relative flex items-end overflow-hidden rounded-xl">
          <img :src="product.images[0]" :alt="product.title" />
        </div>

        <div class="mt-1 p-2">
          <h2 class="text-slate-700 capitalize">{{ product.title }}</h2>
          <p class="mt-1 text-sm text-slate-400">{{ product.gender }}</p>

          <div class="mt-3 flex items-end justify-between">
              <p class="text-lg font-bold text-blue-500">{{ product.price }}</p>
            <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4
```

```
py-1.5 text-white duration-100 hover:bg-blue-600">
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
                <path stroke-linecap="round" stroke-linejoin="round" d="M2.25 3h1.386c.
51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
              </svg>

              <button class="text-sm">Add to cart</button>
            </div>
          </div>
        </div>
      </a>
    </article>
</template>


<script lang="ts" setup>
import type { Product } from '../interfaces/products-response.interface';

    interface Props{
        product: Product
    }

    defineProps<Props>()

</script>
```

Componente para la paginación

- modules/common/components/ButtonPagination.vue

```
<template>
    <div class="flex justify-center py-10 bg-gray-100 space-x-3">
        <button
class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5 bg-blue-500 disabled:bg-
gray-300">
            <span class="text-white text-xl">Anteriores</span>
        </button>

        <button
class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5 bg-blue-500 disabled:bg-
gray-300">
            <span class="text-white text-xl">Siguientes</span>
        </button>
```

```
    </div>
</template>
```

- Coloco el componente debajo del ProductList en HomeView.vue
- Para los iconos puedo crear src/icons en el filesystem
- Busco los iconos en Vector Collections

> https://www.svgrepo.com/collections/all/

- Uso chevron right y chevron left. Si le doy a Edit Vector puedo copiar el SVG
- Les añado un width y un height de unos 25 pixeles
- ChevronRight.vue

```
<template>
    <svg width="25px" height="25px"
    width="25px" heigth="25px"
    viewBox="-5.5 0 26 26" version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:sketch="http://www.bohemiancoding.com/sketch/ns" fill="#000000"><g
id="SVGRepo_bgCarrier"
    stroke-width="0"></g><g id="SVGRepo_tracerCarrier"
    stroke-linecap="round" stroke-linejoin="round"></g>
    <g id="SVGRepo_iconCarrier"> <title>chevron-right</title>
        <desc>Created with Sketch Beta.</desc>
        <defs> </defs>
        <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd"
sketch:type="MSPage">
            <g id="Icon-Set-Filled" sketch:type="MSLayerGroup"
transform="translate(-474.000000, -1196.000000)" fill="#000000">
        <path d="M488.404,1207.36 L477.637,1197.6 C476.806,1196.76 475.459,1196.76
474.629,1197.6 C473.798,1198.43 473.798,1199.77 474.629,1200.6 L483.885,1209
L474.629,1217.4 C473.798,1218.23 473.798,1219.57 474.629,1220.4 C475.459,1221.24
476.806,1221.24 477.637,1220.4 L488.404,1210.64 C488.854,1210.19 489.052,1209.59
489.015,1209 C489.052,1208.41 488.854,1207.81 488.404,1207.36"
        id="chevron-right"
        sketch:type="MSShapeGroup">

    </path>
    </g>
    </g> </g>
    </svg>
</template>
```

- ChevronLeft.vue

```
<template>
    <svg width="25px" height="25px"
    viewBox="-5.5 0 26 26" version="1.1" xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:sketch="http://
```

```
www.bohemiancoding.com/sketch/ns"
    fill="#000000"><g id="SVGRepo_bgCarrier" stroke-width="0"></g><g
id="SVGRepo_tracerCarrier"
    stroke-linecap="round" stroke-linejoin="round"></g><g id="SVGRepo_iconCarrier">
        <title>chevron-left</title>
        <desc>Created with Sketch Beta.</desc>
        <defs> </defs>
        <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd"
sketch:type="MSPage">
        <g id="Icon-Set-Filled" sketch:type="MSLayerGroup"
transform="translate(-423.000000, -1196.000000)" fill="#ffffff">
        <path d="M428.115,1209 L437.371,1200.6 C438.202,1199.77 438.202,1198.43
437.371,1197.6 C436.541,1196.76 435.194,1196.76 434.363,1197.6 L423.596,1207.36
C423.146,1207.81 422.948,1208.41 422.985,1209 C422.948,1209.59 423.146,1210.19
423.596,1210.64 L434.363,1220.4 C435.194,1221.24 436.541,1221.24 437.371,1220.4
C438.202,1219.57 438.202,1218.23 437.371,1217.4 L428.115,1209" id="chevron-left"
sketch:type="MSShapeGroup">
        </path>
        </g> </g>
        </g></svg>
</template>
```

• Para usarlos, los coloco en los botones

```
<template>
    <div class="flex justify-center py-10 bg-gray-100 space-x-3">
        <button class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600">
            <ChevronLeft />
            <span class="text-white text-lg">Anteriores</span>
        </button>

        <button class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600">
            <span class="text-white text-lg">Siguientes</span>
            <ChevronRight />
        </button>

    </div>
</template>

<script setup lang="ts">
import ChevronLeft from '@/icons/ChevronLeft.vue';
import ChevronRight from '@/icons/ChevronRight.vue';

</script>
```

• Para editar el color es más fácil hacerlo desde la web

Funcionalidad de la paginación

- Hay varias maneras de manejar la paginación
- Una de ellas es manejar una variable en JS para saber en qué página estamos, cual es la siguiente, etc, un simple contador
- Lo malo de esta manera es que el usuario no va a poder moverse usando la barra de navegación (adelante y atrás)
- Se recomienda que la paginación venga como argumento en el URL
- ButtonPagination.vue

```
<script setup lang="ts">
import ChevronLeft from '@/icons/ChevronLeft.vue';
import ChevronRight from '@/icons/ChevronRight.vue';

interface Props{
    page: number
    firstPage: boolean
    hasMoreData: boolean
}

defineProps<Props>()

</script>
```

- En el HomeView.vue que es donde está el componente, tengo que pasarle las props

```
<ButtonPagination  :has-more-data="true" :first-page="true" :page="1"/>
```

- Cuando los argumentos son opcionales con el interrogante “localhost:3000/api/products?page=1” no necesitamos definirlos en el router
- Usamos useRoute en el HomeView.vue para extraer la página de la ruta

```
<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue';
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery } from '@tanstack/vue-query';
import { useRoute } from 'vue-router';

const route = useRoute()

const page = route.query.page

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page:1}],
  queryFn: ()=> getProductsAction(),
```

```
})

</script>
```

- Si hago un console.log me devuelve page 1, pero porque en la URL está ?page=1
- Si no hay page me devuelve undefined, además estoy esperando que la página sea un número
- Si no tenemos la página va a ser un 1, lo parseo a número
- Dependiendo de qué página tengo necesito mandárselo al getProductsAction
- Podría hacerse así
- El problema es que cuando el URL cambie, page no es una variable reactiva
- Envolvemos page en un ref
- HomeView.vue

```
<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue';
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery } from '@tanstack/vue-query';
import { ref } from 'vue';
import { useRoute } from 'vue-router';

const route = useRoute()

const page = ref(Number(route.query.page || 1))

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page: page.value}],
  queryFn: ()=> getProductsAction(page.value),
})

</script>
```

- Pasemosle las props al componente (seguimos en HomeView.vue, ahora en el template)
- La doble negación es un valor booleano de si hay productos

```
<ButtonPagination
          :has-more-data="!!products && products.length < 10"
                :first-page="page === 1"
                    :page="page"/>
```

- En el ButtonPagination.vue, cuando hago click quiero navegar a la siguiente pantalla
- Si estamos en la primera página, el botón de anterior está deshabilitado
- Con $router tengo acceso al router, uso .push para añadir una nueva ruta al historial

```vue
<template>
    <div class="flex justify-center py-10 bg-gray-100 space-x-3">
        <button
        class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600"
        :disabled="firstPage"
        @click="$router.push({query: {page: page-1}})"
        >
            <ChevronLeft />
            <span class="text-white text-lg">Anteriores</span>
        </button>

        <button class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600"
        :disabled="hasMoreData"
        @click="$router.push({query: {page: page +1}})"
        >
            <span class="text-white text-lg">Siguientes</span>
            <ChevronRight />
        </button>

    </div>
</template>

<script setup lang="ts">
import ChevronLeft from '@/icons/ChevronLeft.vue';
import ChevronRight from '@/icons/ChevronRight.vue';

interface Props{
    page: number
    firstPage: boolean
    hasMoreData: boolean
}

defineProps<Props>()

</script>
```

- De esta manera no funciona de la manera que espero, se cambia el query parameter pero lo que pasa es que el HomeView no se vuelve a reconstruir, por lo que no me muestra la nueva data
- La página que es una variable reactiva, no se está volviendo a recalcular con el cambio
- Necesito estar pendiente de los cambios
- Para ello puedo usar watch o watcheffect, pero el watch me va a servir para especificar de lo que quiero estar pendiente
- HomeView.vue

```vue
<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue';
import { getProductsAction } from '@/modules/products/actions/get-products.action';
```

```
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery } from '@tanstack/vue-query';
import { ref, watch } from 'vue';
import { useRoute } from 'vue-router';

const route = useRoute()

const page = ref(Number(route.query.page || 1))

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page: page}], //le paso la variable reactiva page
  queryFn: ()=> getProductsAction(page.value),
})

watch(
    ()=>route.query.page,
    (newPage)=>{
        page.value = Number(newPage || 1) //será uno si devuelve undefined
    }
)

</script>
```

• Ya no ocupamos el firstPage, porque ya tenemos el page reactivo, lo podemos borrar

```
<template>
    <div class="flex justify-center py-10 bg-gray-100 space-x-3">
        <button
        class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600"
        :disabled="page == 1"
        @click="$router.push({query: {page: page-1}})"
        >
            <ChevronLeft />
            <span class="text-white text-lg">Anteriores</span>
        </button>

        <button class="flex items-center space-x-1.5 rounded-lg px-4 py-1.5
        bg-blue-500 disabled:bg-gray-300 hover:bg-blue-600"
        :disabled="hasMoreData"
        @click="$router.push({query: {page: page +1}})"
        >
            <span class="text-white text-lg">Siguientes</span>
            <ChevronRight />
        </button>

    </div>
</template>

<script setup lang="ts">
import ChevronLeft from '@/icons/ChevronLeft.vue';
import ChevronRight from '@/icons/ChevronRight.vue';

interface Props{
```

```
    page: number
    hasMoreData: boolean
}

definePropsProps>()

</script>
```

- Faltaría hacer que la pantalla cuando recarga se vea desde arriba

Carga adelantada y scroll

- Además de que la pantalla del navegador vaya arriba cuando cambio de página, quiero precargar la data de la página siguiente, para una mejor experiencia de usuario
- Hago el scroll en el watch
- HomeView.vue

```
<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue';
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery } from '@tanstack/vue-query';
import { ref, watch } from 'vue';
import { useRoute } from 'vue-router';

const route = useRoute()

const page = ref(Number(route.query.page || 1))

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page: page}],
  queryFn: ()=> getProductsAction(page.value),
})

watch(
    ()=>route.query.page,
    (newPage)=>{
        page.value = Number(newPage || 1)

        window.scrollTo({top:0, behavior: 'smooth'})
    }
)

</script>
```

- Hagamos una carga automática basada en cuando la página cambia
- Se recomienda que cada watch tenga una tarea específica
- Uso watchEffect para que se dispare cuando alguna variable reactiva cambie

- Tomo el cliente de TanstackQuery, que hace referencia a el query que tenemos (el queryPlugin, que es el acceso al store)
- Por lo tanto, apuntamos al store
- Le paso el queryKey y le añado al page.value +1
- La queryFn es la misma, también le añado +1

```
<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue';
import { getProductsAction } from '@/modules/products/actions/get-products.action';
import ProductList from '@/modules/products/components/ProductList.vue';
import { useQuery, useQueryClient } from '@tanstack/vue-query';
import { ref, watch, watchEffect } from 'vue';
import { useRoute } from 'vue-router';

const route = useRoute()

const page = ref(Number(route.query.page || 1))
const queryClient = useQueryClient()

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page: page}],
  queryFn: ()=> getProductsAction(page.value),
})

watch(
    ()=>route.query.page,
    (newPage)=>{
        page.value = Number(newPage || 1)

        window.scrollTo({top:0, behavior: 'smooth'})
    }
)

watchEffect(()=>{
    queryClient.prefetchQuery({
        queryKey: ['products', {page: page.value +1}],
        queryFn: ()=> getProductsAction(page.value),
    })
})

</script>
```

- En las devtools de tanstack puedo ver las dos peticiones
- Hay un brinco porque las imágenes no tienen un tamaño fijo
- Les pongo un height de 250 y le añado object-cover
- ProductCard.vue

```
<div class="relative flex items-end overflow-hidden rounded-xl">
    <img
```

```
    class="h-[250px] object-cover"
    :src="product.images[0]" :alt="product.title" />
</div>
```

13 Vue Herrera - Autenticación y autorización

- El backend ya tiene un login tradicional
- Nos traeremos del backend el usuario autenticado, los roles, el jsonwebtoken de autenticación
- Eso nos va a permitir usar los guards por la parte de Vue que permitan entrar a la persona donde quiere entrar, vamos a saber quien es, y sobretodo saber si es administrador. Solo los admin van a poder entrar a ciertos lugares
- Usaremos Pinia. En el state tendremos el user, el token y el authStatus (authenticated/unauthenticated)
- Tendremos varios getters: isChecking, isAuthenticated, isAdmin, username
    - Si no tenemos sesión de usuario, el estado va a ser checking
    - Luego vamos a tomar el token y con este verificamos el estado en el backend
    - Si el usuario tiene el rol de admin, aparecerá la posibilidad de ir a la pantalla de admin (lo veremos en el siguiente módulo)
- Una vez la persona ingresa, no puede regresar al Login (ni hacia atrás en el navegador ni poniendo la ruta de login)
- Habrá un loader global
- Descargar la carpeta auth adjunta
- Tengo guards, layouts y pages. Son de un proyecto anterior (SPA)
- auth/guards/is-authenticated.gurad.ts

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';

const isAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {
  const userId = localStorage.getItem('userId');
  localStorage.setItem('lastPath', to.path);

  if (!userId) {
    return next({
      name: 'login',
    });
  }

  return next();
};
```

```
export default isAuthenticatedGuard;
```

• auth/layouts/AuthLayout.vue

```
<template>
  <!-- component -->
  <div class="bg-gray-100 flex justify-center items-center h-screen">
    <!-- Left: Image -->
    <div class="w-1/2 h-screen hidden lg:block bg-blue-500">

    </div>
    <!-- Right: Login Form -->
    <div class="lg:p-36 md:p-52 sm:20 p-8 w-full lg:w-1/2">
      <RouterView />
    </div>
  </div>
</template>
```

• auth/views/LoginPage.vue

```
<template>
  <h1 class="text-2xl font-semibold mb-4">Login</h1>
  <form action="#" method="POST">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Username</label>
      <input
        type="text"
        id="username"
        name="username"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Password</label>
      <input
        type="password"
        id="password"
        name="password"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Remember Me Checkbox -->
    <div class="mb-4 flex items-center">
      <input type="checkbox" id="remember" name="remember" class="text-blue-500" />
      <label for="remember" class="text-gray-600 ml-2">Remember Me</label>
```

```
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">Forgot Password?</a>
    </div>
    <!-- Login Button -->
    <button
      @click="onLogin"
      type="button"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2
px-4 w-full"
    >
      Login
    </button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
    <RouterLink :to="{ name: 'register' }" class="hover:underline">Sign up Here</
RouterLink>
  </div>
</template>

<script lang="ts" setup>
import { useRouter } from 'vue-router';

const router = useRouter();

const onLogin = () => {
  localStorage.setItem('userId', 'ABC-123');

  const lastPath = localStorage.getItem('lastPath') ?? '/';

  // router.replace({
  //   // name: 'home',
  // });
  router.replace(lastPath);
};
</script>
```

• RegisterPage.vue

```
<template>
  <h1 class="text-2xl font-semibold mb-4">Register</h1>
  <form action="#" method="POST">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="name" class="block text-gray-600">Name</label>
      <input
        type="text"
        id="name"
        name="name"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
```

```
      />
    </div>

    <!-- Username Input -->
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Username</label>
      <input
        type="text"
        id="username"
        name="username"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Password</label>
      <input
        type="password"
        id="password"
        name="password"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Remember Me Checkbox -->
    <div class="mb-4 flex items-center">
      <input type="checkbox" id="remember" name="remember" class="text-blue-500" />
      <label for="remember" class="text-gray-600 ml-2">Remember Me</label>
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">Forgot Password?</a>
    </div>
    <!-- Login Button -->
    <button
      type="submit"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2
px-4 w-full"
    >
      Login
    </button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
    <RouterLink :to="{ name: 'login' }" class="hover:underline">Login Here</RouterLink>
  </div>
</template>
```

- Creo auth/routes/index.ts

- Guardará las rutas relacionadas a mi autenticación. La idea es que sea un objeto que yo pueda colocar directamente en el router

```
import type { RouteRecordRaw } from "vue-router";

export const authRoutes: RouteRecordRaw = {
    path: '/auth',
    name: 'auth',
    component: ()=> import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
        {
        path: 'login',
        name: 'login',
        component: ()=> import('@/modules/auth/views/LoginPage.vue')
        },
        {
        path: 'register',
        name: 'register',
        component: ()=> import('@/modules/auth/views/RegisterPage.vue')
        }

    ]
}
```

- Ahora solo tengo que importarlo en el router dentro de routes
- router/index.ts

```
import { authRoutes } from '@/modules/auth/routes'
import ShopLayout from '@/modules/shop/layouts/ShopLayout.vue'
import HomeView from '@/modules/shop/views/HomeView.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'shop',
      component: ShopLayout,
      children: [
        {
          path: '',
          name: 'home',
          component: ()=>import('@/modules/shop/views/HomeView.vue')
        }
      ]
    },
    authRoutes
  ],
})
```

```
export default router
```

- Ahora si en el navegador apunto a localhost:3000/auth/login voy a la pantalla de login

Login Action - Autenticación

- El endpoint para hacer login es /api/auth/login
- Si no se cumple con los parámetros del backend (verifica email, longitud de password, etc)
- Si el login es exitoso (user: test1@google.com, password:Abc123) me devuelve el id, el email, el fullName, el isActive, un arreglo de roles y el token
- Creo en modules/auth/actions/login.action.ts
- Coloco un try y un catch, hay dos tipos de errores posibles: que el email sea incorrecto o que se caiga la conexión, no tengamos acceso, etc (errores controlados y no controlados)
- Copio la respuesta del login exitoso y uso PasteJsonAsCode para tipar la respuesta de axios
- La pego en auth/interfaces/auth.response.ts

```
import type { User } from "./user.interface";

export interface AuthResponse {
    user:  User;
    token: string;
}

//es el mismo User que el de user.interface.ts

{/*export interface User {
    id:       string;
    email:    string;
    fullName: string;
    isActive: boolean;
    roles:    string[];
}*/}
```

- En el loginAction ya puedo tipar la respuesta de axios y también la de la propia loginAction
- En el catch **uso && con la condición para no usar un if anidado**

```
import { tesloApi } from "@/api/tesloApi"
import type { AuthResponse } from "../interfaces/auth.response"
import { isAxiosError } from "axios"
import type { User } from "../interfaces/user.interface"

interface LoginError {
    ok: false,
    message: string
}
```

```
interface LoginSuccess{
    ok: true,
    user: User,
    token: string
}


export const loginAction = async(
  email: string,
  password: string
): Promise<LoginError|LoginSuccess>=>{
    try {

        const {data} = await tesloApi.post<AuthResponse>('/auth/login', {
            email,
            password
        })

        return {
            ok: true,
            user: data.user,
            token: data.token
        }

    } catch (error) {
        if(isAxiosError(error) && error.response?.status == 401){
                return{
                    ok: false,
                    message: "Usuario o contraseña incorrectos"
                }
        }

        //cualquiewr otro error
        throw new Error("No se pudo realizar la petición")
    }
}
```

Pinia - Auth Store

• Creo el store en auth/stores/auth.store.ts

```
import { defineStore } from "pinia";

export const useAuthStore = defineStore('auth', ()=>{


    return {}
})
```

• Voy a ocupar la interfaz de user y un enum para el status de la autenticación

• Creo varios getters

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { User } from "../interfaces/user.interface";
import { AuthStatus } from "../interfaces/auth-status.enum";

export const useAuthStore = defineStore('auth', ()=>{

    const authStatus = ref(AuthStatus.Checking)
    const user= ref<User | undefined>()
    const token = ref<string>('') //lo voy a tebner que grabar en el localStorage

    return {
        authStatus,
        user,
        token,

        //getters
        isChecking: computed(()=>authStatus.value === AuthStatus.Checking ),
        isAuthenticated: computed(()=>authStatus.value === AuthStatus.Authenticated),
        isUnauthenticated: computed(()=>authStatus.value ===
AuthStatus.Unauthenticated),
        username : computed(()=>user.value?.fullName)

        //Todo: getter para saber si es admin o no
    }
})
```

• En auth/interfaces/auth-status.enum.ts guardo el enum de los estados de la autenticación

```
export enum AuthStatus{
    Authenticated ="Authenticated",
    Unauthenticated= 'Unauthenticated',
    Checking="Checking"
}
```

• Las acciones estan basadas en el gestor de estado, porque el gestor de estado basado en la respuesta de nuestra acción (loginAction) va a determinar los valores de user, token y authStatus

• Creo la función login para llamar a la acción, darle los valores a las propiedades reactivas y mapear la respuesta

• Creo una función logout para devolver la respuesta en caso de error (queda más limpio y es reutilizable)

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { User } from "../interfaces/user.interface";
```

```
import { AuthStatus } from "../interfaces/auth-status.enum";
import { loginAction } from "../actions/login.action";

export const useAuthStore = defineStore('auth', ()=>{

    const authStatus = ref(AuthStatus.Checking)
    const user= ref<User | undefined>()
    const token = ref<string>('') //lo voy a tebner que grabar en el localStorage

    const login = async(email: string, password: string)=>{
        try {
            const loginResp = await loginAction(email, password)
            if(!loginResp.ok){
                return false
            }

            user.value = loginResp.user
            token.value = loginResp.token
            authStatus.value = AuthStatus.Authenticated

            return true

        } catch (error) {
          return logout()
        }
    }

    const logout = ()=>{
          authStatus.value = AuthStatus.Unauthenticated
          user.value = undefined
          token.value = ''
          return false
    }

    return {
        authStatus,
        user,
        token,

        //getters
        isChecking: computed(()=>authStatus.value === AuthStatus.Checking ),
        isAuthenticated: computed(()=>authStatus.value === AuthStatus.Authenticated),
        isUnauthenticated: computed(()=>authStatus.value ===
AuthStatus.Unauthenticated),
        username : computed(()=>user.value?.fullName),
        //Todo: getter para saber si es admin o no

        //Actions
        login
    }
})
```

Realizar proceso de autenticación

- Tenemos que mandar a llamar a nuestra acción de login desde el gestor de Pinia en la pantalla de LoginPage.vue
- Borro este onLogin que tenía

```
const onLogin = () => {
  localStorage.setItem('userId', 'ABC-123');

  const lastPath = localStorage.getItem('lastPath') ?? '/';

  // router.replace({
  //   // name: 'home',
  // });
  router.replace(lastPath);
};
```

- Hagámoslo de nuevo
- onLogin debería llamarse en el submit del formulario, no tanto en el botón
- Por lo tanto, borro el @click del bitton, y en vez de tipo button será de tipo submit
- LoginPage.vue

```
<button
      type="submit"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2
px-4 w-full"
    >
```

- En el form puedo usar @sumbmit con el modificador preventDefault para que no recargue el navegador

```
<!--<form action="#" method="POST">-->
<form @submit.prevent="onLogin">
```

- Si quiero asegurarme que funciona pongo un console.log en el onLogin y al paretar el botón debería aparecer en consola
- Vamos a usar otro tipo de propiedad reactiva. Se podría hacer mediante ref de todas maneras
- Usaremos reactive
- Yo voy a querer trabajar todo con ref hasta donde sea posible (es la recomendación)
- Con ref es más fácil trabajar, sobretodo los arreglos (.push, etc)
- reactive se puede usar cuando se van a almacenar objetos

- Si llamo al objeto MyForm, accedo con Myform.loquesea, no hace falta poner el .value
- script LoginPage.vue

```
const MyForm = reactive({
  email: '',
  password: '',
  rememberMe: false
})
```

- Ahora, en el input del email, ya puedo usar MyForm.email con un v-model
- LoginPage.vue

```
<form @submit.prevent="onLogin">
<!-- Username Input -->
<div class="mb-4">
  <label for="email" class="block text-gray-600">Correo</label>
  <input
    v-model="MyForm.email"
    type="email"
    id="email"
    name="email"
    class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
    autocomplete="off"
  />
</div>
```

- Lo mismo con el otro input y el checkbox de rememberMe
- Coloco un console.log del myForm en el onLogin del script setup para ver el objeto reactivo
- Me devuelve un Proxy(Object)
- Tiene un Handler con un MutablereactiveHandler
- Tiene un Target (con el email, password y rememberMe con los valores por defecto que le puse si no rellené el formulario)
- Tiene un Prototype
- Tiene un IsRevoked en false
- Hay que hacer ciertas validaciones en el onLogin
- Hagamos uso del store y la función principal primero

```
<script lang="ts" setup>
import { reactive } from 'vue';
import { useRouter } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';

const router = useRouter();
const authStore = useAuthStore()
```

```
const MyForm = reactive({
  email: '',
  password: '',
  rememberMe: false
})

const onLogin = async() => {

  if(MyForm.email === ''){
    return emailInputRef.value?.focus()
  }

  if(MyForm.password.length < 6){
    return passwordInputRef.value?.focus()
  }

  if(MyForm.rememberMe){
    localStorage.setItem('email', MyForm.email )
  }else{
    localStorage.removeItem('email')
  }

  const ok = await authStore.login(MyForm.email, MyForm.password)

  if(ok){
    const lastPath = localStorage.getItem('lastPath') ?? '/';

    router.replace(lastPath);
    return
  }

  //toast.error('Usuario/Contraseña no son correctos')

  watchEffect(()=>{
    const email = localStorage.getItem('email')
    if(email){
      MyForm.email = email
      MyForm.rememberMe= true
    }
  })
};
</script>
```

- Si le doy al botón de Login sin poner datos me da un bad request y el ok en false
- Si hago el login con test1@google y el pass Abc123 me devuelve el ok en true
- Si en las devTools voy a Pinia/store, seleccion auth tengo el user que es un objeto con id, email, fullName, isActive, los roles, el token, el authStatus Athenticated, y en los getters tengo el isChecking en false, el isAuthenticated en true y el username Test One
- Si recargo el navegador debería haber purgado la información del store pero no lo hizo
- Llamo al logout si el ok está en false

```ts
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { User } from "../interfaces/user.interface";
import { AuthStatus } from "../interfaces/auth-status.enum";
import { loginAction } from "../actions/login.action";

export const useAuthStore = defineStore('auth', ()=>{

    const authStatus = ref(AuthStatus.Checking)
    const user= ref<User | undefined>()
    const token = ref<string>('')

    const login = async(email: string, password: string)=>{
        try {
            const loginResp = await loginAction(email, password)
            if(!loginResp.ok){
                logout() //AQUI!
                return false
            }

            user.value = loginResp.user
            token.value = loginResp.token
            authStatus.value = AuthStatus.Authenticated

            return true

        } catch (error) {
          logout()
          return false //lo necesito para el vue toastification
        }
    }

    const logout = ()=>{
          authStatus.value = AuthStatus.Unauthenticated
          user.value = undefined
          token.value = ''
          return false
    }

    return {
        authStatus,
        user,
        token,

        //getters
        isChecking: computed(()=>authStatus.value === AuthStatus.Checking ),
        isAuthenticated: computed(()=>authStatus.value === AuthStatus.Authenticated),
        isUnauthenticated: computed(()=>authStatus.value ===
AuthStatus.Unauthenticated),
        username : computed(()=>user.value?.fullName),
        //Todo: getter para saber si es admin o no

        //Actions
        login
```

```
    }
})
```

- Ahora si toco el botón lanza la excepción (badRequest), borra la información del store, y aparece unauthenticated
- Recordemos el loginAction

```
import { tesloApi } from "@/api/tesloApi"
import type { AuthResponse } from "../interfaces/auth.response"
import { isAxiosError } from "axios"
import type { User } from "../interfaces/user.interface"

interface LoginError {
    ok: false,
    message: string
}

interface LoginSuccess{
    ok: true,
    user: User,
    token: string
}


export const loginAction = async(
    email: string,
    password: string
): Promise<LoginError|LoginSuccess>=>{
    try {

        const {data} = await tesloApi.post<AuthResponse>('/auth/login', {
            email,
            password
        })

        return {
            ok: true,
            user: data.user,
            token: data.token
        }

    } catch (error) {
        if(isAxiosError(error) && error.response?.status == 401){
                return{
                    ok: false,
                    message: "Usuario o contraseña incorrectos"
                }
        }

        //cualquiewr otro error
        throw new Error("No se pudo realizar la petición")
```

```
    }
}
```

- Vamos a mejorar la experiencia de usuario
- Quizá noe s tan necesario validar aqui en el frontend, pero si ocuparemos mostrar mensajes de error

Terminar pantalla de login

- Creo dos propiedades reactivas en LoginPage.vue, emailInputRef y passwordInputRef
- A la hora de hacer un login debemos verificar que estas dos variables tienen información
- Si no la tienen debo colocar el foco en los respectivos campos
- Uso el ref de los input para colocar las variables reactivas. **Ref es el único que no lleva el : del bind!**
- En el onLogin puedo hacer una evaluación rápida (luego validaremos que sea un correo electrónico, usaremos esquemas de validación)

```
<template>
  <h1 class="text-2xl font-semibold mb-4">Login</h1>
  <form @submit.prevent="onLogin">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="email" class="block text-gray-600">Correo</label>
      <input
        ref="emailInputRef"
        v-model="MyForm.email"
        type="email"
        id="email"
        name="email"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Contraseña</label>
      <input
        ref="passwordInputRef"
        v-model="MyForm.password"
        type="password"
        id="password"
        name="password"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Remember Me Checkbox -->
```

```
    <div class="mb-4 flex items-center">
      <input
      v-model="MyForm.rememberMe"
      type="checkbox" id="remember" name="remember" class="text-blue-500" />
      <label for="remember" class="text-gray-600 ml-2">Recordar Usuario</label>
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">¿Olvidaste la contraseña?</a>
    </div>
    <!-- Login Button -->
    <button
      type="submit"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2
px-4 w-full"
    >
      Login
    </button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
    <RouterLink :to="{ name: 'register' }" class="hover:underline">Regístrate aquí</
RouterLink>
  </div>
</template>

<script lang="ts" setup>
import { reactive } from 'vue';
import { useRouter } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';
import { ref } from 'vue';

const router = useRouter();
const authStore = useAuthStore()
const emailInputRef = ref<HTMLInputElement| null>(null)
const passwordInputRef = ref<HTMLInputElement| null>(null)


const MyForm = reactive({
  email: '',
  password: '',
  rememberMe: false
})

const onLogin = async() => {

  if(MyForm.email === ''){
    return emailInputRef.value?.focus()
  }

  if(MyForm.password.length < 6){
    return passwordInputRef.value?.focus()
  }

  if(MyForm.rememberMe){
    localStorage.setItem('email', MyForm.email )
```

```
  }else{
    localStorage.removeItem('email')
  }

  const ok = await authStore.login(MyForm.email, MyForm.password)

  if(ok){
    const lastPath = localStorage.getItem('lastPath') ?? '/';

    router.replace(lastPath);
    return
  }

  //toast.error('Usuario/Contraseña no son correctos')

  watchEffect(()=>{
    const email = localStorage.getItem('email')
    if(email){
      MyForm.email = email
      MyForm.rememberMe= true
    }
  })
};
</script>
```

• Para mostrar los errores usaremos el paquete (algo antiguo ya) Vue toastification

> npm i vue-toastification@next

• Configuro el main

```
import './assets/styles.css'
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import Toast from 'vue-toastification'
import 'vue-toastification/dist/index.css'
import App from './App.vue'
import router from './router'
import { VueQueryPlugin } from '@tanstack/vue-query'

const app = createApp(App)

app.use(createPinia())
app.use(VueQueryPlugin)
app.use(router)
app.use(Toast)

app.mount('#app')
```

• Ya puedo usar el toast en el login

```
const onLogin = async() => {

  if(MyForm.email === ''){
    return emailInputRef.value?.focus()
  }

  if(MyForm.password.length < 6){
    return passwordInputRef.value?.focus()
  }

  if(MyForm.rememberMe){
    localStorage.setItem('email', MyForm.email )
  }else{
    localStorage.removeItem('email')
  }

  const ok = await authStore.login(MyForm.email, MyForm.password)

  if(ok){
    const lastPath = localStorage.getItem('lastPath') ?? '/';

    router.replace(lastPath);
    return
  }

  toast.error('Usuario/Contraseña no son correctos') //AQUI

  watchEffect(()=>{
    const email = localStorage.getItem('email')
    if(email){
      MyForm.email = email
      MyForm.rememberMe= true
    }
  })
};
```

- Vamos a usar VueUse para manejar el token en el LocalStorage

> npm i @vueuse/core

- Manejamos el token con useLocalStorage en el auth.store
- useLocalStorage ya devuelve un ref, por lo que no hay que usar ref

```
const token = useLocalStorage('token', '')
```

- Si recargo el navegador, aunque no ponga nada se graba el token (con un string vacío)
- Si hago el login correcto se guarda el token
- Si recargo el navegador, la información de pantalla se pierde pero la del localstorage se mantiene

- Eso es algo que vamos a cambiar, vamos a hacer que la info permanezca en pantalla en los inputs
- Podríamos crear un watch para que cargue esa información
- LoginPage.vue

```
const onLogin = async() => {

  if(MyForm.email === ''){
    return emailInputRef.value?.focus()
  }

  if(MyForm.password.length < 6){
    return passwordInputRef.value?.focus()
  }

  if(MyForm.rememberMe){
    localStorage.setItem('email', MyForm.email )
  }else{
    localStorage.removeItem('email')
  }

  const ok = await authStore.login(MyForm.email, MyForm.password)

  if(ok) return

  toast.error('Usuario/Contraseña no son correctos')

  watchEffect(()=>{
    const email = localStorage.getItem('email')
    if(email){
      MyForm.email = email
      MyForm.rememberMe= true
    }
  })
};
```

- Ahora falta el registro
- **NOTA**: cambio LoginPage y RegisterPage por LoginView RegisterView
- Lo cambio en el auth.router

```
import type { RouteRecordRaw } from "vue-router";

export const authRoutes: RouteRecordRaw = {
    path: '/auth',
    name: 'auth',
    component: ()=> import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
        {
        path: 'login',
        name: 'login',
```

```
        component: ()=> import('@/modules/auth/views/LoginView.vue')
        },
        {
        path: 'register',
        name: 'register',
        component: ()=> import('@/modules/auth/views/RegisterView.vue')
        }

    ]
}
```

Registro de usuarios

- Creo una nueva acción en auth/actions/register.action.ts que recibe el fullName, el email y el password

```
import { tesloApi } from "@/api/tesloApi"
import type { AuthResponse } from "../interfaces/auth.response"
import { isAxiosError } from "axios"
import type { LoginError, LoginSuccess } from "./login.action"

export const registerAction =
async (fullName: string, email: string, password: string)
:Promise<LoginError | LoginSuccess> =>{

  try {
    const {data} = await tesloApi.post<AuthResponse>('/auth/register',{
        fullName,
        email,
        password
    })

    return {
        ok: true,
        user: data.user,
        token: data.token
    }

    } catch (error) {
        if(isAxiosError(error)){
            return {
                ok: false,
                message: error.message
            }
        }

        throw new Error('No se ha podido realizar la petición')
    }
}
```

- La uso en el auth.store en la función register

- Es idéntico al login solo que con un campo más (el fullName)
- Cuando hay un error devuelvo un objeto con el ok y el message (debería hacerlo en el login también por consistencia)
- Con el true retorno el message vacío

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { User } from "../interfaces/user.interface";
import { AuthStatus } from "../interfaces/auth-status.enum";
import { loginAction } from "../actions/login.action";
import { useLocalStorage } from "@vueuse/core";
import { registerAction } from "../actions/register.action";

export const useAuthStore = defineStore('auth', ()=>{

    const authStatus = ref(AuthStatus.Checking)
    const user= ref<User | undefined>()
    const token = useLocalStorage('token', '')

    const login = async(email: string, password: string)=>{
        try {
            const loginResp = await loginAction(email, password)
            if(!loginResp.ok){
                logout()
                return false

            }

            user.value = loginResp.user
            token.value = loginResp.token
            authStatus.value = AuthStatus.Authenticated

            return true

        } catch (error) {
          logout()
          return false
        }
    }

    const register =async (fullName: string, email: string, password: string)=>{
        try {
            const registerResp = await registerAction(fullName, email, password)
            if(!registerResp.ok){
                logout()
                return {ok:false, message: registerResp.message}
            }

            user.value = registerResp.user
            token.value = registerResp.token
            authStatus.value = AuthStatus.Authenticated

            return {ok: true, message: ''}
```

```
        } catch (error) {
            logout()
            return {ok:false, message: 'Error al registrar el usuario'}
        }


    }

    const logout = ()=>{
          authStatus.value = AuthStatus.Unauthenticated
          user.value = undefined
          token.value = ''
          return false
    }

    return {
        authStatus,
        user,
        token,

        //getters
        isChecking: computed(()=>authStatus.value === AuthStatus.Checking ),
        isAuthenticated: computed(()=>authStatus.value === AuthStatus.Authenticated),
        isUnauthenticated: computed(()=>authStatus.value ===
AuthStatus.Unauthenticated),
        username : computed(()=>user.value?.fullName),
        //Todo: getter para saber si es admin o no

        //Actions
        login,
        register
    }
})
```

- En el RegisterView.vue cambio lo que hay en el form y uso @submit.prevent, le paso la función onRegister
- Uso v-model para guardar los valores en MyForm
- Uso el ref para ponerle el foco a los elementos si no hay texto
- Extraigo el ok y el message de la acción. Si el ok está en true, return. Si no, imprime el mensaje en el toast

```
<template>
  <h1 class="text-2xl font-semibold mb-4">Register</h1>
  <form @submit.prevent="onRegister">
    <!-- Username Input -->
    <div class="mb-4">
      <label for="username" class="block text-gray-600">Usuario</label>
      <input
        v-model="MyForm.fullName"
        ref="fullNameInputRef"
        type="text"
```

```
        id="username"
        name="username"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!--email-->
    <div class="mb-4">
      <label for="email" class="block text-gray-600">Correo</label>
      <input
        v-model="MyForm.email"
        ref="emailInputRef"
        type="email"
        id="email"
        name="email"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Password Input -->
    <div class="mb-4">
      <label for="password" class="block text-gray-600">Contraseña</label>
      <input
        v-model="MyForm.password"
        ref="passwordInputRef"
        type="password"
        id="password"
        name="password"
        class="w-full border border-gray-300 rounded-md py-2 px-3 focus:outline-none
focus:border-blue-500"
        autocomplete="off"
      />
    </div>
    <!-- Forgot Password Link -->
    <div class="mb-6 text-blue-500">
      <a href="#" class="hover:underline">¿Olvidaste la contraseña?</a>
    </div>
    <!-- Register Button -->
    <button
      type="submit"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold rounded-md py-2
px-4 w-full"
    >
      Ingresar
    </button>
  </form>
  <!-- Sign up  Link -->
  <div class="mt-6 text-blue-500 text-center">
    <RouterLink :to="{ name: 'login' }" class="hover:underline">Login</RouterLink>
  </div>
</template>

<script lang="ts" setup>
import { reactive } from 'vue';
```

```
import { useAuthStore } from '../stores/auth.store';
import { ref } from 'vue';
import { useToast } from 'vue-toastification';

const emailInputRef = ref<HTMLInputElement| null>(null)
const passwordInputRef = ref<HTMLInputElement| null>(null)
const fullNameInputRef = ref<HTMLInputElement | null>(null)
const toast = useToast()

const MyForm= reactive({
  fullName: '',
  email: '',
  password: ''
})

  const authStore = useAuthStore()

  const onRegister = async ()=>{
      if(MyForm.email === ''){
    return emailInputRef.value?.focus()
  }

  if(MyForm.password.length < 6){
    return passwordInputRef.value?.focus()
  }

  if(MyForm.fullName === ''){
    return fullNameInputRef.value?.focus()
  }

  const {ok, message} = await authStore.register(MyForm.fullName, MyForm.email,
MyForm.password)

  if(ok) return

  toast.error(message)

  }

</script>
```

- Para comprobar que se ha hecho el registro usar TablePlus con la data del .env (del backend)

```
STAGE=dev

DB_PASSWORD=MySecr3tPassWord@as2 ----> La contraseña
DB_NAME=TesloDB ---> El nombre de la DB
DB_HOST=localhost ---> host
DB_PORT=5432 ---> El puerto
DB_USERNAME=postgres ---> El usuario

PORT=3000
HOST_API=http://localhost:3000/api
```

```
JWT_SECRET=Est3EsMISE3Dsecreto32s
```

Interceptores de Axios

- Tenemos guardado el token en el localStorage cuando hacemos el Login
- Cuando recargamos el navegador tenemos que verificar si el token sigue siendo vigente, renovarlo, etc
- En el backend tengo un endpoint que es auth/check-status
- En POSTMAN, debo pegar en authorizations/Bearer Token/ Token el token que me devuelve el login (sin comillas)
- La respuesta, en caso de ser exitosa, es la misma que la de LoginSuccess (intefaz)
- Si no regresa “Unauthorized”
- Para hacer esta tarea vamos a usar los interceptorres de Axios
- Se puede hacer con el fecthAPI tambien
- En api/tesloApi.ts hago uso del interceptor en la request (también hay en la response)
- En config tengo la configuración de axios
- Siempre debe retornar la config, es lo mínimo que me pide para ser un interceptor
- Autenticación es poder identificar un usuario y autorización es saber si ese usuario tiene acceso a algo
- En este caso, aunque es autenticación, la palabra en headres en Authorization

```
import axios from 'axios'


const tesloApi = axios.create({
    baseURL: import.meta.env.VITE_TESLO_API_URL
})

tesloApi.interceptors.request.use((config)=>{
    const token = localStorage.getItem('token')

  //para que no envie el token en estos llamados
    const isAuthRoute =
        config.url?.includes('/auth/login') ||
        config.url?.includes('/auth/register')

    if(token && !isAuthRoute){
        config.headers.Authorization = `Bearer ${token}`
    }
    return config
})
```

```
export {tesloApi}
```

- POSTMAN tiene una opción que es en el icono del tag y puedo seleccionar NodeJs- Axios y me dice cómo tengo que establecer mi configuración
- Hay que mandar en los headers un objeto con 'Authorization': 'Bearer oiua87s9ab976s897as6b_token'

```
//icono
</>
```

- Creo la nueva acción modules/auth/actions/check-status.action.ts que llamará cada vez que se recargue el navegador

```
import { tesloApi } from "@/api/tesloApi"
import type { User } from "../interfaces/user.interface"
import type { AuthResponse } from "../interfaces/auth.response"
import { isAxiosError } from "axios"

interface CheckError{
    ok: false, //coloco explicitamente false para que TypScript pueda inferir
}

interface CheckSuccess{
    ok: true,
    user: User,
    token: string
}

export const checkStatusAction =async (): Promise<CheckError | CheckSuccess>=>{
    try {
        const localToken = localStorage.getItem('token')

        if(!localToken){
            return {ok: false}
        }

        if(localToken && localToken.length < 10) return {ok: false}

        const {data} = await tesloApi.get<AuthResponse>('/auth/check-status')

        return {
            ok: true,
            user: data.user,
            token: data.token
        }

    } catch (error) {
        if(isAxiosError(error) && error.response?.status === 401){
            return {
```

```
                ok: false
            }
        }

        throw new Error("No se ha podido verificar la sesión")
    }
}
```

- Veamos como mandar a llamar esta acción para preservar la sesión del usuario

Subscripción de estado y redirecciones

- Cuando recargamos el navegador se pierde la información del store cuando estoy autenticado (exceptuando el token que está en el LocalStorage)
- Podríamos guardarlo todo en el LocalStorage, pero yo voy a querer verificar ese token, puede ser que ya haya expirado, que el usuario esté bloqueado, etc
- Cada vez que se recargue el navegador, vamos a revissar las credenciales basado en el token
- Para ello creamos el checkAuthAction. ¿Dónde lo llamamos?
- auth.store

```
const checkAuthStatus =async (): Promise<boolean>=>{
        try {
            const statusResponse = await checkStatusAction()

            if(!statusResponse.ok){
                authStatus.value = AuthStatus.Unauthenticated
                return false
            }

            authStatus.value = AuthStatus.Authenticated
            user.value = statusResponse.user
            token.value = statusResponse.token

            return true
        } catch (error) {
            logout() //deslogging
            return false
        }
    }
```

- Coloco esta action checkAuthStatus en el return del store para poder usarla
- Un punto importante dónde llamar a este checkAuthStatus en el App.vue, ya que toda nuestra aplicación pasa por aquí
- Hay una forma de suscribirse a todo el store

- Con $subscribe podemos suscribirnos a los cambios que pueda disparar el store, no solo el state, las mutaciones (acciones) también

```
<template>
<router-view />
<vue-query-devtools />
</template>

<script lang="ts" setup>
  import { VueQueryDevtools} from '@tanstack/vue-query-devtools';
import { useAuthStore } from './modules/auth/stores/auth.store';

  const authStore = useAuthStore()

  authStore.$subscribe((mutation, state)=>{
    console.log({mutation,state})
  })


</script>
```

- Si hago un login puedo ver en consola lo que ha pasado
- De esta manera puedo obtener cualquier cambio en el state (cuando se realiza un cambio en este)
- Pero al recargar el navegador no se dispara
- Para ello tengo el objeto de opciones
- Con el immediate en true, se ejecuta tan pronto el componente se monta
- Cuando no quiero ocupar el argumento uso un guión bajo

```
 authStore.$subscribe((_, state)=>{
    console.log({state})
  }, {
    immediate: true
  })
```

- Puedo usar el state para verificar si estoy autenticado o no
- Si está en checking llamo al checkAuthStatus, si sale bien, me setea los valores
- Si sale mal, llama al logout que tenemos el cambio a Unauthenticated
- Cuando estamos autenticados queremos redireccionar al HomeScreen
- App.vue (script setup)

```
import { VueQueryDevtools} from '@tanstack/vue-query-devtools';
import { useAuthStore } from './modules/auth/stores/auth.store';
import { AuthStatus } from './modules/auth/interfaces/auth-status.enum';
import { useRoute, useRouter } from 'vue-router';
```

```
  const authStore = useAuthStore()
  const router = useRouter()
  const route = useRoute()

  authStore.$subscribe((_, state)=>{
    if(state.authStatus === AuthStatus.Checking){
      authStore.checkAuthStatus()
      return
    }

    if(route.path.includes('/auth') && state.authStatus === AuthStatus.Authenticated){
      //uso replace para no poder navegar a la pantalla anterior si ya está autenticado
      router.replace({name: 'home'})
      return
    }
  }, {
    immediate: true
  })
```

- De esta manera si estoy autenticado me redirige al home
- Pero si estoy autenticado **no debería poder ir al Login**
- Para ello usaremos un **Guard** (o que los botones no aparecieran)
- Si estoy autenticado, el botón de Login debería ser de Logout

Guard - No autenticado

- Si estamos autenticados no deberíamos poder ir a la pantalla de Login
- Creemos un Guard que impida entrar en esta pantalla si estamos autenticados
- En auth/guards pegamos este código que no está acabado
- is-authenitcated.guard.ts

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';

const isAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {
  const userId = localStorage.getItem('userId');
  localStorage.setItem('lastPath', to.path);

  if (!userId) {
    return next({
      name: 'login',
    });
  }

  return next();
};
```

```
export default isAuthenticatedGuard;
```

- Creo una copia que se llama is-not-authenticated.ts
- Hay que exportarlo por defecto para usarlo como lo usaremos

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';

const isNotAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {

  console.log(to) //veamos que tiene to (hacia donde va)


  return next();
};

export default isNotAuthenticatedGuard;
```

- Lo uso en el router
- Tengo el beforeEnter
- routes/index.ts

```
import type { RouteRecordRaw } from "vue-router";
import isNotAuthenticatedGuard from "../guards/is-not-authenticated.guard";

export const authRoutes: RouteRecordRaw = {
    path: '/auth',
    name: 'auth',
    beforeEnter:[isNotAuthenticatedGuard],
    component: ()=> import('@/modules/auth/layouts/AuthLayout.vue'),
    children:[
        {
        path: 'login',
        name: 'login',
        component: ()=> import('@/modules/auth/views/LoginView.vue')
        },
        {
        path: 'register',
        name: 'register',
        component: ()=> import('@/modules/auth/views/RegisterView.vue')
        }
```

```
    ]
}
```

- Si voy al Login puedo ver en consola el to
- Quiere ir al fullPath:'/auth/login'
- Podríamos determinar el estado tomándolo del authStore y usar un ternario

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';
import { AuthStatus } from '../interfaces/auth-status.enum';

const isNotAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {

  const authStore = useAuthStore();

  (authStore.authStatus === AuthStatus.Authenticated) ? next({name:'home'}) : next()

};

export default isNotAuthenticatedGuard;
```

- Pero si yo conozco la ruta y voy en el navegador a auth/login me deja entrar
- Debo verificar en este punto con el checkAuthStatus si estoy autenticado o no, porque si no estoy checking

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';
import { AuthStatus } from '../interfaces/auth-status.enum';

const isNotAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {

  const authStore = useAuthStore();

  await authStore.checkAuthStatus();

  (authStore.authStatus === AuthStatus.Authenticated) ? next({name:'home'}) : next()

};
```

```
export default isNotAuthenticatedGuard;
```

- Ahora si quiero ir a /auth/login no me deja entrar
- Falta el is-authenticated y verificar roles

Cerrar sesión

- Si estoy autenticado debería ver el botón de Logout
- Si mi usuario tiene el rol de admin quiero que aparezca el botón de Admin
- Quiero determinar desde el store si el usuario es administrador con un getter
- **NOTA**: Removemos el token en el logout!
- auth.store.ts

```
import { defineStore } from "pinia";
import { computed, ref } from "vue";
import type { User } from "../interfaces/user.interface";
import { AuthStatus } from "../interfaces/auth-status.enum";
import { loginAction } from "../actions/login.action";
import { useLocalStorage } from "@vueuse/core";
import { registerAction } from "../actions/register.action";
import { checkStatusAction } from "../actions/check-status.action";

export const useAuthStore = defineStore('auth', ()=>{

    const authStatus = ref(AuthStatus.Checking)
    const user= ref<User | undefined>()
    const token = useLocalStorage('token', '')

    const login = async(email: string, password: string)=>{
        try {
            const loginResp = await loginAction(email, password)
            if(!loginResp.ok){
                logout()
                return false

            }

            user.value = loginResp.user
            token.value = loginResp.token
            authStatus.value = AuthStatus.Authenticated

            return true

        } catch (error) {
          logout()
          return false
        }
    }
```

```
    const register =async (fullName: string, email: string, password: string)=>{
        try {
            const registerResp = await registerAction(fullName, email, password)
            if(!registerResp.ok){
                logout()
                return {ok:false, message: registerResp.message}
            }

            user.value = registerResp.user
            token.value = registerResp.token
            authStatus.value = AuthStatus.Authenticated

            return {ok: true, message: ''}

        } catch (error) {
            logout()
            return {ok:false, message: 'Error al registrar el usuario'}
        }


    }

    const checkAuthStatus =async (): Promise<boolean>=>{
        try {
            const statusResponse = await checkStatusAction()

            if(!statusResponse.ok){
                authStatus.value = AuthStatus.Unauthenticated
                return false
            }

            authStatus.value = AuthStatus.Authenticated
            user.value = statusResponse.user
            token.value = statusResponse.token

            return true
        } catch (error) {
            logout() //deslogging
            return false
        }
    }



    const logout = ()=>{
        localStorage.removeItem('token')
          authStatus.value = AuthStatus.Unauthenticated
          user.value = undefined
          token.value = ''
          return false
    }

    return {
        authStatus,
        user,
        token,
```

```
        //getters
        isChecking: computed(()=>authStatus.value === AuthStatus.Checking ),
        isAuthenticated: computed(()=>authStatus.value === AuthStatus.Authenticated),
        isUnauthenticated: computed(()=>authStatus.value ===
AuthStatus.Unauthenticated),
        username : computed(()=>user.value?.fullName),
        //si es admin regresa un true, también puede regresar undefined, por lo que
regresamos un false en ese caso
        isAdmin: computed(()=> user.value?.roles.includes('admin') ?? false), // ??
sirve para "es una cosa u es otra"

        //Actions
        login,
        logout,
        register,
        checkAuthStatus
    }
})
```

- shop/components/TopMenu.vue
- Llamo al authStore desde el script setup. Uso un template para renderizar condicionalmente con el v-if los botones

```
 <template v-if="!authStore.isAuthenticated">
    <button
    type="button"
    class="rounde mr-3 hidden border hover:bg-blue-300 border-blue-700 py-1.5 px-6
text-center text-sm font-medium text-blue-700 focus:outline-none focus:ring-4
focus:ring-blue-300 md:inline-block rounded-lg">
      <RouterLink :to="{ name: 'login' }" class="hover:no-underline">
        Login
      </RouterLink>
    </button>
    <button
    type="button"
    class="rounde mr-3 hidden bg-blue-700 py-1.5 px-6 text-center text-sm font-medium
text-white hover:bg-blue-800 focus:outline-none focus:ring-4 focus:ring-blue-300
md:mr-0 md:inline-block rounded-lg">
      <RouterLink :to="{ name: 'register' }" class="hover:no-underline">
        Register
      </RouterLink>
    </button>
  </template>
  <template v-else="authStore.isAuthenticated">
    <button
    type="button"
    class="rounde mr-3 hidden border hover:bg-blue-300 border-blue-700 py-1.5 px-6
text-center text-sm font-medium text-blue-700 focus:outline-none focus:ring-4
focus:ring-blue-300 md:inline-block rounded-lg">
      <RouterLink v-if="authStore.isAdmin" :to="{ name: 'admin' }" class="hover:no-
underline">
        Admin
```

```
      </RouterLink>
    </button>
    <button
    @click="authStore.logout()"
    type="button"
    class="rounde mr-3 hidden bg-blue-700 py-1.5 px-6 text-center text-sm font-medium
text-white hover:bg-blue-800 focus:outline-none focus:ring-4 focus:ring-blue-300
md:mr-0 md:inline-block rounded-lg">
      Logout
    </button>
</template>
```

• Creo el componente modules/admin/layouts/AdminLayout.vue

```
<template>
    <h1>AdminLayout</h1>
    <RouterView />
</template>
```

• Como en auth, creo en admin/routes/index.ts

```
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue')
}
```

• Lo coloco en el router principal

```
import { adminRoutes } from '@/modules/admin/routes'
import { authRoutes } from '@/modules/auth/routes'
import ShopLayout from '@/modules/shop/layouts/ShopLayout.vue'
import HomeView from '@/modules/shop/views/HomeView.vue'
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'shop',
      component: ShopLayout,
      children: [
        {
          path: '',
          name: 'home',
          component: ()=>import('@/modules/shop/views/HomeView.vue')
        }
```

```
      ]
    },
    authRoutes,
    adminRoutes
  ],
})

export default router
```

- Ahora hay que crear un Guard para que solo pueda acceder a la ruta admin si el usuario es admin

Guard isAdminGuard

- En auth/guards/is-admin.guard.ts

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';
import { AuthStatus } from '../interfaces/auth-status.enum';

const isAdminGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {
  const authStore = useAuthStore();

    await authStore.checkAuthStatus();

    authStore.isAdmin ? next(): next({name: 'home'})
};

export default isAdminGuard;
```

- Y el is-authenticated.guard

```
import type { NavigationGuardNext, RouteLocationNormalized } from 'vue-router';
import { useAuthStore } from '../stores/auth.store';
import { AuthStatus } from '../interfaces/auth-status.enum';

const isAuthenticatedGuard = async (
  to: RouteLocationNormalized,
  from: RouteLocationNormalized,
  next: NavigationGuardNext,
) => {
  const authStore = useAuthStore();

    await authStore.checkAuthStatus();

    (authStore.authStatus === AuthStatus.Unauthenticated) ? next({name:'home'}) :
```

```
next()
};

export default isAuthenticatedGuard;
```

- En el objeto del router del admin uso el before>Enter y le paso los dos guards

```
import isAdminGuard from "@/modules/auth/guards/is-admin.guard";
import isAuthenticatedGuard from "@/modules/auth/guards/is-authenticated.guard";
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    beforeEnter: [isAuthenticatedGuard, isAdminGuard],
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue')
}
```

- Es importante usar el isAuthenticatedGuard también con el isAdminGuard
- En principio ahora el código que hiciomos con $subscribe en App.vue no sería necesario
- Pongamos un loader a nivel global

Pantalla de carga

- Se entrega este loader para usar
- Hay muchos otros spinners

> https://github.com/n3r4zzurr0/svg-spinners

```
<svg width="24" height="24" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/
svg"><style>.spinner_9y7u{animation:spinner_fUkk 2.4s linear infinite;animation-
delay:-2.4s}.spinner_DF2s{animation-delay:-1.6s}.spinner_q27e{animation-delay:-.8s}
@keyframes spinner_fUkk{8.33%{x:13px;y:1px}25%{x:13px;y:1px}33.3%{x:13px;y:13px}50%{x:
13px;y:13px}58.33%{x:1px;y:13px}75%{x:1px;y:13px}83.33%{x:1px;y:1px}}</style><rect
class="spinner_9y7u" x="1" y="1" rx="1" width="10" height="10"/><rect
class="spinner_9y7u spinner_DF2s" x="1" y="1" rx="1" width="10" height="10"/><rect
class="spinner_9y7u spinner_q27e" x="1" y="1" rx="1" width="10" height="10"/></svg>
```

- En modules/common/components/FullScreenLoader.vue copio el svg
- Uso shift+Alt+F para formatear el código
- La etiqueta style no puede ir dentro del template
- Uso scoped para que solo aplique el css en este componente
- Para asegurarme que el loader quede en el centro de la pantalla lo meto dentro de un div
- Uso la propiedad fill para añadirle un color

```
<template>
    <div class="w-screen h-screen flex justify-center items-center ">
        <svg width="24" height="24" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/
svg">
            <rect class="spinner_9y7u" x="1" y="1" rx="1" width="10" height="10"
fill="#007bff" />
            <rect class="spinner_9y7u spinner_DF2s" x="1" y="1" rx="1" width="10"
height="10" />
            <rect class="spinner_9y7u spinner_q27e" x="1" y="1" rx="1" width="10"
height="10" />
        </svg>
    </div>
</template>

 <style scoped>
    .spinner_9y7u {
        animation: spinner_fUkk 2.4s linear infinite;
        animation-delay: -2.4s
    }

    .spinner_DF2s {
        animation-delay: -1.6s
    }

    .spinner_q27e {
        animation-delay: -.8s
    }

    @keyframes spinner_fUkk {
        8.33% {
            x: 13px;
            y: 1px
        }

        25% {
            x: 13px;
            y: 1px
        }

        33.3% {
            x: 13px;
            y: 13px
        }

        50% {
            x: 13px;
            y: 13px
        }

        58.33% {
            x: 1px;
            y: 13px
        }

        75% {
```

```
            x: 1px;
            y: 13px
        }

        83.33% {
            x: 1px;
            y: 1px
        }
    }
</style>
```

- Vamos al App.vue. Uso un víf para renderizar el loader si el estado es isChecking y un v-else para el RouterView

```
<template>
  <full-screen-loader
    v-if="authStore.isChecking"
  />
<router-view v-else/>
<vue-query-devtools />
</template>

<script lang="ts" setup>
  import { VueQueryDevtools} from '@tanstack/vue-query-devtools';
import { useAuthStore } from './modules/auth/stores/auth.store';
import FullScreenLoader from './modules/common/components/FullScreenLoader.vue';
import { AuthStatus } from './modules/auth/interfaces/auth-status.enum';
import { onMounted } from 'vue';


  const authStore = useAuthStore()

  onMounted(async () => {
  await authStore.checkAuthStatus()
})
</script>
```

14 Vue Herrera - Formularios y Mantenimiento de Productos

- Nos vamos a centrar en los formularios
- Haremos que nuestor customInput reciba un v-model
- Usaremos VeeValidate junto a un validador de esquemas llamado Yup
- En este módulo todavía no vamos a mandar la data pero si la vamos a preparar para enviarla a nuestro backend
- Crearemos un dashboard administrativo con un menú lateral
- En el top bar mostraremos el nombre del usuario
- Mostraremos el listado de productos. Si clico encima del nombre voy al formulario
    - Al entrar tengo la información del producto existente en los inputs lista para ser editada
    - Tengo el título, el slug, la descripción, el precio, las tallas, inventario, para subir la imagen, vista previa de las imágenes, el género

Continuación

- En este enlace tengo un panel administrativo

> https://www.creative-tim.com/twcomponents/component/admin

- Lo pego en AdminLayout.vue. La vista la guardo en un componente aparte (DashboardView), es el último div que representa los productos
- Si el componente viniera en un body **hay que quitarlo**
- Le hago algunos retoques
- AdminLayout.vue

```
<template>
<div class="flex w-screen h-screen text-gray-700">
            <div class="flex flex-col items-center w-16 pb-4 overflow-auto border-r
border-gray-300">
                <a class="flex items-center justify-center flex-shrink-0 w-full h-16
bg-gray-300" href="#">
                    <svg class="w-8 h-8"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 7l-8-4-8 4m16 0l-8 4m8-4v10l-8 4m0-10L4 7m8 4v10M4 7v10l8 4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
```

```
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1
1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2 2 0
01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0 01-.707.293h-3.172a1 1 0
01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M16 8v8m-4-5v5m-4-2v2m-2 4h12a2 2 0 002-2V6a2 2 0 00-2-2H6a2 2 0 00-2
2v12a2 2 0 002 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2"
d="M12 6V4m0 2a2 2 0 100 4m0-4a2 2 0 110 4m-6 8a2 2 0 100-4m0 4a2 2 0 110-4m0
4v2m0-6V4m6 6v10m6-2a2 2 0 100-4m0 4a2 2 0 110-4m0 4v2m0-6V4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 mt-auto rounded
hover:bg-gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
```

```
width="2" d="M5.121 17.804A13.937 13.937 0 0112 16c2.5 0 4.847.655 6.879 1.804M15 10a3
3 0 11-6 0 3 3 0 016 0zm6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
                    </svg>
                </a>
            </div>
            <div class="flex flex-col w-56 border-r border-gray-300">
                <button class="relative text-sm focus:outline-none group">
                    <div class="flex items-center justify-between w-full h-16 px-4
border-b border-gray-300 hover:bg-gray-300">
                        <span class="font-medium">
                            Dropdown
                        </span>
                        <svg class="w-4 h-4" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 20 20" fill="currentColor">
                            <path fill-rule="evenodd" d="M5.293 7.293a1 1 0 011.414
0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0
010-1.414z" clip-rule="evenodd" />
                        </svg>
                    </div>
                    <div class="absolute z-10 flex-col items-start hidden w-full pb-1
bg-white shadow-lg group-focus:flex">
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                    </div>
                </button>
                <div class="flex flex-col flex-grow p-4 overflow-auto">
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 1</span>
                    </a>
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 2</span>
                    </a>
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 3</span>
                    </a>
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 4</span>
                    </a>
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 5</span>
                    </a>
                    <a class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Item 6</span>
                    </a>
                    <a
class="flex items-center flex-shrink-0 h-10 px-3 mt-auto text-sm font-medium bg-
```

```
gray-200 rounded hover:bg-gray-300"
                        href="#">
                        <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
                        </svg>
                        <span class="ml-2 leading-none">New Item</span>
                    </a>
                </div>

            </div>
            <div class="flex flex-col flex-grow w-full">
                <div class="flex items-center flex-shrink-0 h-16 px-8 border-b border-
gray-300 justify-between">
                    <h1 class="text-lg font-medium">Page Title</h1>
                    <div class="flex items-center ml-auto">
                        <button class="flex items-center justify-center h-10 px-4 text-
sm font-medium rounded hover:bg-gray-300">
                            Action 1
                        </button>
                        <button class="flex items-center justify-center h-10 px-4 ml-2
text-sm font-medium bg-gray-200 rounded hover:bg-gray-300">
                            Action 2
                        </button>
                        <button class="relative ml-2 text-sm focus:outline-none group">
                            <div class="flex items-center justify-between w-10 h-10
rounded hover:bg-gray-300">
                                <svg class="w-5 h-5 mx-auto" xmlns="http://www.w3.org/
2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                    <path stroke-linecap="round" stroke-
linejoin="round" stroke-width="2"
d="M12 5v.01M12 12v.01M12 19v.01M12 6a1 1 0 110-2 1 1 0 010 2zm0 7a1 1 0 110-2 1 1 0
010 2zm0 7a1 1 0 110-2 1 1 0 010 2z" />
                                </svg>
                            </div>
                            <div class="absolute right-0 flex-col items-start hidden
w-40 pb-1 bg-white border border-gray-300 shadow-lg group-focus:flex">
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                            </div>
                        </button>
                    </div>
                </div>
                <div class="flex-grow p-6 overflow-auto bg-gray-200">
                   <router-view />
                </div>
            </div>


</div>
```

```
<a
class="fixed flex items-center justify-center h-8 pr-2 pl-1 bg-blue-600 rounded-full
bottom-0 right-0 mr-4 mb-4 shadow-lg text-blue-100 hover:bg-blue-600" href="https://
twitter.com/lofiui" target="_top">
    <div class="flex items-center justify-center h-6 w-6 bg-blue-500 rounded-full">
        <svg viewBox="0 0 24 24"
class="w-4 h-4 fill-current r-jwli3a r-4qtqp9 r-yyyyoo r-16y2uox r-1q142lx r-8kz0gk r-
dnmrzs r-bnwqim r-1plcrui r-lrvibr r-1srniue"><g><path d="M23.643 4.937c-.
835.37-1.732.62-2.675.733.962-.576 1.7-1.49 2.048-2.578-.9.534-1.897.922-2.958 1.13-.
85-.904-2.06-1.47-3.4-1.47-2.572 0-4.658 2.086-4.658 4.66 0 .364.042.718.12
1.06-3.873-.195-7.304-2.05-9.602-4.868-.4.69-.63 1.49-.63 2.342 0 1.616.823 3.043 2.072
3.878-.764-.025-1.482-.234-2.11-.583v.06c0 2.257 1.605 4.14 3.737 4.568-.392.106-.
803.162-1.227.162-.3 0-.593-.028-.877-.082.593 1.85 2.313 3.198 4.352 3.234-1.595
1.25-3.604 1.995-5.786 1.995-.376 0-.747-.022-1.112-.065 2.062 1.323 4.51 2.093 7.14
2.093 8.57 0 13.255-7.098 13.255-13.254 0-.2-.005-.402-.014-.602.91-.658 1.7-1.477
2.323-2.41z"></path></g></svg>
    </div>
    <span class="text-sm ml-1 leading-none">@lofiui</span>
</a>
</template>
```

- Creo admin/views/DashboardView.vue con la vista (con el último div del componente, los productos)

```
<template>
     <div class="grid grid-cols-3 gap-6">
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-3 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-3 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-1 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-2 bg-white border border-gray-300"></div>
        <div class="h-24 col-span-3 bg-white border border-gray-300"></div>
```

```
    </div>
</template>
```

- Debemos configurar la ruta en admin/routes/index.ts
- Redirecciono a admin-dashboard

```
import isAdminGuard from "@/modules/auth/guards/is-admin.guard";
import isAuthenticatedGuard from "@/modules/auth/guards/is-authenticated.guard";
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    beforeEnter: [isAuthenticatedGuard, isAdminGuard],
    redirect: {name: 'admin-dashboard'},
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue'),
    children:[
        {
            path:'dashboard',
            name: 'admin-dashboard',
            component: ()=> import('@/modules/admin/views/DashboardView.vue')
        }
    ]
}
```

Pantalla de productos

- Creo admin/views/ProductsView.vue
- Usaremos este componente Striped Table

> https://www.creative-tim.com/twcomponents/component/striped-table

- Le añado un div con un h1 y en el mismo div meto el componente
- Cambio los table headers
- Los table rows los borro todos menos uno
- Creo el script setup
- En el componente shop/views/HomeView.vue tengo el mismo código que voy a ocupar de tanstackQuery para la petición de productos
- Uso los productos en el table row con un v-for. Uso un v-bind en el class para aplicar uun fondo gris a las filas pares
- Uso la primera imagen del arreglo. Habría que colocar algo por si no hay imagen
- Uso un router-link para el titulo, lo mando a una ruta que todavía no existe
- El precio aparece en una especie de pastillita azul
- Uno las tallas que vienen en el array por comas

- Cambio los width de los table headers para que se ajusten bien
- Quito también los w-1/3 de los td, así no hace brincos
- Coloco al final el BottomPagination
- Le paso el page y en hasmore data uso la doble negación para transformar a true products (es decir, si tengo productos) y si el hasMoreData es menor a 10
- ProductsView.vue

```
<template>

<div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Productos</h1>
    <div class="py-8 w-full">
  <div class="shadow overflow-hidden rounded border-b border-gray-200">
    <table class="min-w-full bg-white">
      <thead class="bg-gray-800 text-white">
        <tr>
          <th class="w-10text-left py-3 px-4 uppercase font-semibold text-sm">Imagen</
th>
          <th class="flex-1 text-left py-3 px-4 uppercase font-semibold text-
sm">Título</th>
          <th class="w-28 text-left py-3 px-4 uppercase font-semibold text-sm">Precio</
th>
          <th class="w-60 text-left py-3 px-4 uppercase font-semibold text-sm">Tallas</
th>
        </tr>
      </thead>
    <tbody class="text-gray-700">
      <tr
      v-for="(product, index) in products"
            :key="product.id"
            :class="{
                'bg-gray-100': index % 2 === 0 //coloco fondo gris en los pares
            }">
        <td class="text-left py-3 px-4">
            <img
            class="h-10 w-10 object-cover"
            :src="product.images[0]" :alt="product.title" />
        </td>
        <td class="text-left py-3 px-4">
            <router-link
                :to="`/admin/products/${product.id}`"
                class="hover:text-blue-500 hover:underline"
            >{{ product.title }}
            </router-link>
        </td>
        <td class="text-left py-3 px-4">
            <span class="bg-blue-200 text-blue-600 py-1 px-3 rounded-full text-xs">
                {{ product.price }}
            </span>
        </td>
        <td class="text-left py-3 px-4">
            {{ product.sizes.join(',') }}
```

```
        </td>
      </tr>
    </tbody>
    </table>
  </div>
      <ButtonPagination
        :page="page"
        :has-more-data="!!products && products.length > 10"
    />
</div>
</div>
</template>

<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue'
import { getProductsAction } from '@/modules/products/actions'
import { useQuery, useQueryClient } from '@tanstack/vue-query'
import { ref, watch, watchEffect } from 'vue'
import { useRoute } from 'vue-router'

  const route = useRoute()

const page = ref(Number(route.query.page || 1))
const queryClient = useQueryClient()

const {data:products, isLoading} = useQuery({
  queryKey:['products', {page: page}],
  queryFn: ()=> getProductsAction(page.value),
})

watch(
    ()=>route.query.page,
    (newPage)=>{
        page.value = Number(newPage || 1)

        window.scrollTo({top:0, behavior: 'smooth'})
    }
)

watchEffect(()=>{
    queryClient.prefetchQuery({
        queryKey: ['products', {page: page.value +1}],
        queryFn: ()=> getProductsAction(page.value +1),
    })
})
</script>
```

• El products/action/getProductsAction es este

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
```

```
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)

        return data.map(product=>({
            ...product,
            images: product.images.map(getProductImageAction)}))

    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

• El products/action/getProductImageAction

```
export const getProductImageAction =(imageName: string): string=> {

    return imageName.includes('http')
    ? imageName
    : `${import.meta.env.VITE_TESLO_API_URL}/files/product/${imageName}`
}
```

• Creamos la ruta para visualizar ProductsView.vue en admin/routes/index.ts

```
import isAdminGuard from "@/modules/auth/guards/is-admin.guard";
import isAuthenticatedGuard from "@/modules/auth/guards/is-authenticated.guard";
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    beforeEnter: [isAuthenticatedGuard, isAdminGuard],
    redirect: {name: 'admin-dashboard'},
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue'),
    children:[
        {
            path:'dashboard',
            name: 'admin-dashboard',
            component: ()=> import('@/modules/admin/views/DashboardView.vue')
        },
        {
            path:'products',
            name: 'admin-products',
            component: ()=> import('@/modules/admin/views/ProductsView.vue')
        },
```

```
    ]
}
```

- Si apunto en el url a admin/products, veo la tabla en lugar de los productos (que tampoco veo los productos, no hemos hecho la petición para mostrar los productos)

usePagination - Composable

- Optimización. Creemos un custom composable para la paginación
- Hay quien se crea composable para usar el useQuery, así no hay que ir repitiendo el queryKey
- En realidad, solo necesito obtener la página
- common/composables/usePagination.ts

```
import { ref, watch } from "vue"
import { useRoute } from "vue-router"

export const usePagination =()=>{

    const route = useRoute()
    const page = ref(Number(route.query.page || 1))

    watch(
    ()=>route.query.page,
    (newPage)=>{
        page.value = Number(newPage || 1)

        window.scrollTo({top:0, behavior: 'smooth'})
    }
)

    return {
        page
    }
}
```

- Usémoslo en el script de ProductsView.vue

```
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue'
import { usePagination } from '@/modules/common/composables/use-pagination'
import { getProductsAction } from '@/modules/products/actions'
import { useQuery, useQueryClient } from '@tanstack/vue-query'
import { watchEffect } from 'vue'

const {page} = usePagination()
const queryClient = useQueryClient()

const {data:products, isLoading} = useQuery({
```

```
  queryKey:['products', {page: page}],
  queryFn: ()=> getProductsAction(page.value),
})


watchEffect(()=>{
    queryClient.prefetchQuery({
        queryKey: ['products', {page: page.value +1}],
        queryFn: ()=> getProductsAction(page.value +1),
    })
})
```

Pantalla de Producto

- Creo en admin/routes/index.ts la ruta hija para mostrar el componente de producto

```
import isAdminGuard from "@/modules/auth/guards/is-admin.guard";
import isAuthenticatedGuard from "@/modules/auth/guards/is-authenticated.guard";
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    beforeEnter: [isAuthenticatedGuard, isAdminGuard],
    redirect: {name: 'admin-dashboard'},
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue'),
    children:[
        {
            path:'dashboard',
            name: 'admin-dashboard',
            component: ()=> import('@/modules/admin/views/DashboardView.vue')
        },
        {
            path:'products',
            name: 'admin-products',
            component: ()=> import('@/modules/admin/views/ProductsView.vue')
        },
        {
            path:'products/:productId',
            name: 'admin-product',
            component: ()=> import('@/modules/admin/views/ProductView.vue')
        },
    ]
}
```

- Creo el componente ProductView.vue
- Fernando proporciona el formulario en un Gist

> https://gist.github.com/Klerith/26abee32f3d75c7f3ca52d8ec50f2ffd

```
<template>
  <div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Producto: <small class="text-blue-500">nombre</small></h1>
    <hr class="my-4" />
  </div>

  <form class="grid grid-cols-1 sm:grid-cols-2 bg-white px-5 gap-5">
    <div class="first-col">
      <!-- Primera parte del formulario -->
      <div class="mb-4">
        <label for="title" class="form-label">Título</label>
        <input type="text" id="title" class="form-control" />
      </div>

      <div class="mb-4">
        <label for="slug" class="form-label">Slug</label>
        <input type="text" id="slug" class="form-control" />
      </div>

      <div class="mb-4">
        <label for="description" class="form-label">Descripción</label>
        <textarea
          id="description"
          class="shadow h-32 appearance-none border rounded w-full py-2 px-3 text-
gray-700 leading-tight focus:outline-none focus:shadow-outline"
        ></textarea>
      </div>

      <div class="flex flex-row gap-3">
        <div class="mb-4">
          <label for="price" class="form-label">Precio</label>
          <input type="number" id="price" class="form-control" />
        </div>

        <div class="mb-4">
          <label for="stock" class="form-label">Inventario</label>
          <input type="number" id="stock" class="form-control" />
        </div>
      </div>

      <div class="mb-4">
        <label for="sizes" class="form-label">Tallas</label>
        <button type="button" class="bg-blue-100 p-2 rounded w-14 mr-2">XS</button>
        <button type="button" class="bg-blue-500 text-white p-2 rounded w-14 mr-2">S</
button>
        <button type="button" class="bg-blue-500 text-white p-2 rounded w-14 mr-2">M</
button>
      </div>
    </div>

    <!-- Segunda columna -->
    <div class="first-col">
      <label for="stock" class="form-label">Imágenes</label>
      <!-- Row with scrollable horizontal -->
      <div class="flex p-2 overflow-x-auto space-x-8 w-full h-[265px] bg-gray-200
rounded">
```

```
        <div class="flex-shrink-0">
          <img src="https://via.placeholder.com/250" alt="imagen" class="w-[250px] h-
[250px]" />
        </div>

        <div class="flex-shrink-0">
          <img src="https://via.placeholder.com/250" alt="imagen" class="w-[250px] h-
[250px]" />
        </div>
      </div>
      <!-- Upload image -->
      <div class="col-span-2 my-2">
        <label for="image" class="form-label">Subir imagen</label>

        <input multiple type="file" id="image" class="form-control" />
      </div>

      <div class="mb-4">
        <label for="stock" class="form-label">Género</label>
        <select class="form-control">
          <option value="">Seleccione</option>
          <option value="kid">Niño</option>
          <option value="women">Mujer</option>
          <option value="men">Hombre</option>
        </select>
      </div>

      <!-- Botón para guardar -->
      <div class="my-4 text-right">
        <button
          type="submit"
          class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded"
        >
          Guardar
        </button>
      </div>
    </div>
  </form>
</template>

<style scoped>
 @reference 'tailwindcss'
.form-label {
  @apply block text-gray-700 text-sm font-bold mb-2;
}

.form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
tight focus:outline-none;
}
</style>
```

- Tengo un título Producto: {nombre del producto}

- A la izquierda tengo una caja para el Titulo, debajo el slug, debajo la descripción en un text-area, debajo dos cajas con precio e inventario y debajo las tallas
- A la derecha tengo dos imágenes (que no están todavía), debajo la caja con el botón para subir imagen y debajo el género
- Abajo a la derecha está el botón de guardar
- Creemos las opciones en el dashboard, en el menú lateral para navegar entre dashboard y productos
- Donde va el PageTitle quiero poner el nombre del usuario activo, en el authStore tengo el getter username
- No se aconseja desestructurar directamente del store
- Habría que usar **storeToRefs**(authStore()) para poder desestructurar
- En AdminLayout.vue

```
<template>
<div class="flex w-screen h-screen text-gray-700">
            <div class="flex flex-col items-center w-16 pb-4 overflow-auto border-r
border-gray-300">
                <a class="flex items-center justify-center flex-shrink-0 w-full h-16
bg-gray-300" href="#">
                    <svg class="w-8 h-8"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 7l-8-4-8 4m16 0l-8 4m8-4v10l-8 4m0-10L4 7m8 4v10M4 7v10l8 4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1
1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
```

```
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2 2 0
01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0 01-.707.293h-3.172a1 1 0
01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M16 8v8m-4-5v5m-4-2v2m-2 4h12a2 2 0 002-2V6a2 2 0 00-2-2H6a2 2 0 00-2
2v12a2 2 0 002 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2"
d="M12 6V4m0 2a2 2 0 100 4m0-4a2 2 0 110 4m-6 8a2 2 0 100-4m0 4a2 2 0 110-4m0
4v2m0-6V4m6 6v10m6-2a2 2 0 100-4m0 4a2 2 0 110-4m0 4v2m0-6V4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 mt-auto rounded
hover:bg-gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M5.121 17.804A13.937 13.937 0 0112 16c2.5 0 4.847.655 6.879 1.804M15 10a3
3 0 11-6 0 3 3 0 016 0zm6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
                    </svg>
                </a>
            </div>
            <div class="flex flex-col w-56 border-r border-gray-300">
                <button class="relative text-sm focus:outline-none group">
                    <div class="flex items-center justify-between w-full h-16 px-4
border-b border-gray-300 hover:bg-gray-300">
                        <span class="font-medium">
                            Dropdown
                        </span>
                        <svg class="w-4 h-4" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 20 20" fill="currentColor">
                            <path fill-rule="evenodd" d="M5.293 7.293a1 1 0 011.414
0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0
010-1.414z" clip-rule="evenodd" />
                        </svg>
                    </div>
                    <div class="absolute z-10 flex-col items-start hidden w-full pb-1
bg-white shadow-lg group-focus:flex">
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
```

```
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                    </div>
                </button>

                <!--Menú lateral-->
                <div class="flex flex-col flex-grow p-4 overflow-auto">
                    <RouterLink
                        to="/admin"
                        class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Dashboard</span>
                    </RouterLink>
                    <RouterLink
                        to="/admin/products"
                        class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Productos</span>
                    </RouterLink>

                    <a
class="flex items-center flex-shrink-0 h-10 px-3 mt-auto text-sm font-medium bg-
gray-200 rounded hover:bg-gray-300"
                        href="#">
                        <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
                        </svg>
                        <!--Botón para crear unuevo Producto-->
                        <span class="ml-2 leading-none">Nuevo Producto</span>
                    </a>
                </div>

            </div>
            <div class="flex flex-col flex-grow w-full">
                <div class="flex items-center flex-shrink-0 h-16 px-8 border-b border-
gray-300 justify-between">
                    <h1 class="text-lg font-medium">{{ authStore.username }}</h1>
                    <div class="flex items-center ml-auto">
                        <button class="flex items-center justify-center h-10 px-4 text-
sm font-medium rounded hover:bg-gray-300">
                            Action 1
                        </button>
                        <button class="flex items-center justify-center h-10 px-4 ml-2
text-sm font-medium bg-gray-200 rounded hover:bg-gray-300">
                            Action 2
                        </button>
                        <button class="relative ml-2 text-sm focus:outline-none group">
                            <div class="flex items-center justify-between w-10 h-10
rounded hover:bg-gray-300">
                                <svg class="w-5 h-5 mx-auto" xmlns="http://www.w3.org/
2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                    <path stroke-linecap="round" stroke-
```

```
linejoin="round" stroke-width="2"
d="M12 5v.01M12 12v.01M12 19v.01M12 6a1 1 0 110-2 1 1 0 010 2zm0 7a1 1 0 110-2 1 1 0
010 2zm0 7a1 1 0 110-2 1 1 0 010 2z" />
                                </svg>
                            </div>
                            <div class="absolute right-0 flex-col items-start hidden
w-40 pb-1 bg-white border border-gray-300 shadow-lg group-focus:flex">
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                            </div>
                        </button>
                    </div>
                </div>
                <div class="flex-grow p-6 overflow-auto bg-gray-200">
                   <router-view />
                </div>
            </div>


</div>

<a
class="fixed flex items-center justify-center h-8 pr-2 pl-1 bg-blue-600 rounded-full
bottom-0 right-0 mr-4 mb-4 shadow-lg text-blue-100 hover:bg-blue-600" href="https://
twitter.com/lofiui" target="_top">
    <div class="flex items-center justify-center h-6 w-6 bg-blue-500 rounded-full">
        <svg viewBox="0 0 24 24"
class="w-4 h-4 fill-current r-jwli3a r-4qtqp9 r-yyyyoo r-16y2uox r-1q142lx r-8kz0gk r-
dnmrzs r-bnwqim r-1plcrui r-lrvibr r-1srniue"><g><path d="M23.643 4.937c-.
835.37-1.732.62-2.675.733.962-.576 1.7-1.49 2.048-2.578-.9.534-1.897.922-2.958 1.13-.
85-.904-2.06-1.47-3.4-1.47-2.572 0-4.658 2.086-4.658 4.66 0 .364.042.718.12
1.06-3.873-.195-7.304-2.05-9.602-4.868-.4.69-.63 1.49-.63 2.342 0 1.616.823 3.043 2.072
3.878-.764-.025-1.482-.234-2.11-.583v.06c0 2.257 1.605 4.14 3.737 4.568-.392.106-.
803.162-1.227.162-.3 0-.593-.028-.877-.082.593 1.85 2.313 3.198 4.352 3.234-1.595
1.25-3.604 1.995-5.786 1.995-.376 0-.747-.022-1.112-.065 2.062 1.323 4.51 2.093 7.14
2.093 8.57 0 13.255-7.098 13.255-13.254 0-.2-.005-.402-.014-.602.91-.658 1.7-1.477
2.323-2.41z"></path></g></svg>
    </div>
    <span class="text-sm ml-1 leading-none">@lofiui</span>
</a>
</template>

<script lang="ts" setup>
import { useAuthStore } from '@/modules/auth/stores/auth.store';


    const authStore = useAuthStore()
```

```
</script>
```

- ProductView (el formulario) es un componente que va a manejar mucha lógica: inputs, validaciones, etc
- Creo en admin/views/ProductView.ts

```
import { defineComponent } from "vue"

export default defineComponent({

    setup(){
        console.log('Hola mundo')
    }
})
```

- Debo pasarle el archivo al script de ProductView.vue

```
<script src="./ProductView.ts" lang="ts"></script>
```

- Creo un getter en ProductView.ts

```
import { computed, defineComponent } from "vue"

export default defineComponent({



    setup(){
        console.log('Hola mundo')


        return{
            // Properties


            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL']


            // Actions

        }
    }
})
```

- Ahora puedo usar el allSizes para mostrar las tallas en el ProductView

```
<div class="mb-4">
    <label for="sizes" class="form-label">Tallas</label>
    <div class="flex">
         <button
                v-for="size of allSizes"
                :key="size"
                type="button"
                class="bg-blue-100 p-2 rounded w-14 mr-2 flex-1">
                {{ size }}
         </button>
    </div>
</div>
```

Cargar información del producto

- Quiero cargar la data que contiene el producto al que hago clic en ProductsView del dashboard de admin
- Que cuando se habra la vista de ProductView con rl formulario cargue la data del producto al que he clicado
- Creemos una acción que nos sirva para traer la info de un producto, la crearemos en products//actions/get-product-by-id.action.ts

```
import { tesloApi } from "@/api/tesloApi"
import type { Product } from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"
import { isAxiosError } from "axios"

export const getProductById =async (id: string)=>{

    //TODO: pensar la creación de un nuevo producto

    try {
        const {data} = await tesloApi.get<Product>(`/products/${id}`)
        //console.log(data) para ver el producto en consola

        return{
            ...data,
            images: data.images.map(getProductImageAction)
        }
    } catch (error) {
        console.log(error)

        throw new Error('Error obteniendo el producto')
    }
}
```

- El getProductImageAction.ts

```
export const getProductImageAction =(imageName: string): string=> {

    return imageName.includes('http')
    ? imageName
    : `${import.meta.env.VITE_TESLO_API_URL}/files/product/${imageName}`
}
```

- En admin/views/ProductView.ts (donde está la lógica)
- Cuando se cargue el componente necesito traerme la info del producto
- ¿Cómo le paso el id? El defineProps solo funciona en el script setup
- Creo el objeto de props y se lo paso a la función setup

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { computed, defineComponent } from "vue"

export default defineComponent({

    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        //defineProps() solo funciona en el script setup

        const {data: product} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })


        return{
            // Properties


            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],


            // Actions

        }
```

```
    }
})
```

- Hay una property en el router (props) que hay que ponerle en true para que haga el match con la property productId

```
import isAdminGuard from "@/modules/auth/guards/is-admin.guard";
import isAuthenticatedGuard from "@/modules/auth/guards/is-authenticated.guard";
import type { RouteRecordRaw } from "vue-router";

export const adminRoutes: RouteRecordRaw ={
    path: '/admin',
    name: 'admin',
    beforeEnter: [isAuthenticatedGuard, isAdminGuard],
    redirect: {name: 'admin-dashboard'},
    component : ()=> import('@/modules/admin/layouts/AdminLayout.vue'),
    children:[
        {
            path:'dashboard',
            name: 'admin-dashboard',
            component: ()=> import('@/modules/admin/views/DashboardView.vue')
        },
        {
            path:'products',
            name: 'admin-products',
            component: ()=> import('@/modules/admin/views/ProductsView.vue')
        },
        {
            path:'products/:productId',
            name: 'admin-product',
            props: true, //props en true!!<----------------------
            component: ()=> import('@/modules/admin/views/ProductView.vue')
        },
    ]
}
```

- Puedo poner un console.log de la data en la action para verla en pantalla
- Si no encontrara el producto habría que sacar al usuario de la pantalla producto
- Usaremos un watchEffect. En lugar de router.push uso replace porque no quiero que se pueda volver a esta pantalla si no existe
- ProductView.ts

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { computed, defineComponent, watchEffect } from "vue"
import { useRouter } from "vue-router"

export default defineComponent({

    props:{
```

```
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        watchEffect(()=>{
            if(isError.value && !isLoading.value){
                router.replace('/admin/products')
                return
            }
        })


        return{
            // Properties

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL']


            // Actions
        }
    }
})
```

Ref y reactive - El problema

- Son 8 campos. Va a requerir mucho trabajo
- Que el título tenga un valor, que el slug tenga un valor, que las tallas correspondan alas existentes, etc
- Podríamos trabajar con ref para declarar un objeto myForm reactivo y usar v-model, con un v-if si el myForm tiene data
- Quiero decirle al usuario cuando estoy en Title y le doy Tab para pasar a Slug, quiero decirle al usuario que Tutle es obligatorio
- Validaciones, vaya. Un schema de validación que cumpla ciertas condiciones para disparar las notificaciones

- Usando ref habría que hacer muchas validaciones manuales, los casos cuando sale bien, los focos, los estilos...

VeeValidate

> npm i vee-validate

- No necesita configuración global
- Importamos useForm en ProductView.ts
- Desestructuro values del useForm, lo retorno en la función setup

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { defineComponent, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useForm} from 'vee-validate'

export default defineComponent({

    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {values} = useForm()

        watchEffect(()=>{
            if(isError.value &&!isLoading.value){
                router.replace('/admin/products')
                return
            }
        })


        return{
            // Properties
            values,
```

```
            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions

        }
    }
})
```

- Ahora ya puedo ir a ProductView.vue
- Vamos a crear una manera de visualizar los valores del formulario de forma temporal en pantalla, abajo de la vista del formulario
- Lo coloco dentro del template, al final

```
<div class="grid grid-cols-2">
  <div class="bg-blue-300">
    {{ values }}
  </div>
</div>
```

- Todavía el formulario no tiene ninguna data, por lo que aparecen unas llaves vacías {}
- Necesitamos ir conectando los valores al formulario
- definefield lo tomamos también de useForm
- Lo usamos para llamar a una función que me va a regresae el email y el emailAtrrs
- email sería el valor y emailAttrs las propiedades

```
const {defineField} = useForm()

const [email, emailAttrs] = defineField('email')
```

- En ProductView.ts (dentro de la función setup)

```
const {values, defineField} = useForm()

const [title, titleAttrs] = defineField('title')
const [slug, slugAttrs] = defineField('slug')
const [description, descriptionAttrs] = defineField('description')
const [price, priceAttrs] = defineField('price')
const [stock, stockAttrs] = defineField('stock')
const [gender, genderAttrs] = defineField('gender')
//TODO: defineArray para las tallas y las imagenes
```

- Todo esto necesito retornarlo en el return de la función setup

```
```

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { defineComponent, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useForm} from 'vee-validate'

export default defineComponent({

    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {values, defineField} = useForm()

        const [title, titleAttrs] = defineField('title')
        const [slug, slugAttrs] = defineField('slug')
        const [description, descriptionAttrs] = defineField('description')
        const [price, priceAttrs] = defineField('price')
        const [stock, stockAttrs] = defineField('stock')
        const [gender, genderAttrs] = defineField('gender')
        //TODO: defineArray para las tallas y las imagenes

        watchEffect(()=>{
            if(isError.value &&!isLoading.value){
                router.replace('/admin/products')
                return
            }
        })


        return{
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
            priceAttrs,
```

```
            stock,
            stockAttrs,
            gender,
            genderAttrs,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions

        }
    }
})
```

- Conectemos el título con v-model y v-bind

```
<input
    v-model="title"
    v-bind="titleAttrs"
    type="text"
    id="title"
    class="form-control" />
```

- Si empiezo a escribir en el input del titulo puedo visualizarlo en el div que coloqué con values al final del template de ProductView.vue
- Si quiero forzar a que el título tenga un valor, ¿cómo lo hago?
- Usaremos un validador de schema llamado **yup**
- El schema se lo puedo pasar al useForm y de esta manejar los errores

> npm i yup

- Puedo colocar el schema fuera del defineComponent en ProductView.ts
- Lo llamo validationSchema, que es igual que el nombre de la propiedad para pasarle el schema
- Si no tendría que poner validationSchema: miSchema, en el useForm
- Desestructuro los errores y los retorno en el return del setup()

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { defineComponent, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useForm} from 'vee-validate'
import * as yup from 'yup'

const validationSchema = yup.object({
    title: yup.string().required().min(2), //minimo 2 caracteres
    slug: yup.string().required(),
    description: yup.string().required(),
```

```
    price: yup.number().required(),
    stock: yup.number().min(1).required(),
    gender: yup.string().required().oneOf(['men', 'women', 'kid']),
})

export default defineComponent({

    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {values, defineField, errors} = useForm({
            validationSchema
        })

        const [title, titleAttrs] = defineField('title')
        const [slug, slugAttrs] = defineField('slug')
        const [description, descriptionAttrs] = defineField('description')
        const [price, priceAttrs] = defineField('price')
        const [stock, stockAttrs] = defineField('stock')
        const [gender, genderAttrs] = defineField('gender')
        //TODO: defineArray para las tallas y las imagenes

        watchEffect(()=>{
            if(isError.value &&!isLoading.value){
                router.replace('/admin/products')
                return
            }
        })


        return{
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
```

```
            priceAttrs,
            stock,
            stockAttrs,
            gender,
            genderAttrs,
            errors,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions
        }
    }
})
```

- En ProductView.vue me creo otro div para visualizar el validationSchema

```
<div class="grid grid-cols-2">
    <div class="bg-blue-300">
        {{ values }}
    </div>
    <div class="bg-red-300">
        {{ errors }}
    </div>
</div>
```

- Si ahora me pongo en el input de Title en el navegador y me salgo sin escribir nada, me salta el error { “title”: “title is a required field” }
- Lo puedo ver en pantalla porque estoy renderizando los errores en el div dentro de ProductView.vue

Mostrar los errores en pantalla

- Centrémonos en el title, ya conectaremos el resto del formulario, puesto que vamos a crear un componente reutilizable para reutilizar la lógica
- Una forma elegante sería marcar el borde en rojo cuando hay un error
- Uso :class para añadir lógica al css con tailwind
- Dentro del arreglo dejo la clase form-control para que siempre la tenga y abro llaves para usar lógica
- ProductView.vue

```
<input
    v-model="title"
    v-bind="titleAttrs"
    type="text"
    id="title"
    :class="['form-control', {
```

```
        'border-red-500': errors.title //si hay un error el borde será rojo
    }]" />
```

- Para mostarr el error podemos usar un span
- ProductView.vue

```
<div class="mb-4">
    <label for="title" class="form-label">Título</label>
    <input
    v-model="title"
    v-bind="titleAttrs"
    type="text"
    id="title"
    :class="['form-control', {
        'border-red-500': errors.title
    }]" />
    <span
    class="text-red-400"
    v-if="errors.title">{{ errors.title }}</span>
</div>
```

- Como todos los componentes van a tener la misma lógica, lo mejor será crear un componente reutilizable CustomInput
- Puedo crear un v-model personalizado. Para que ese v-model funcione hay que definir el update:modelValue y el :value con el modelValue definido en las props

Custom input y v-model

- En common/components/CustomInput.vue
- Conecto las props al input
- A :value le paso el modelValue
- Uso @input porque es un evento, uso el emit con update:modelValue. Esto habilita el v-model
- Ocupamos emitir el valor que tenga el input (y siempre queremos emitirlo) usamos (($event.target as HTMLInputElement)?.value ?? ''). Si no emite nada emite un string vacio
- Uso blur para cuando pierde el foco emitir blur
- Uso defineEmits para ambos
- CustomInput.vue

```
<template>
    <input
      :type="type"
      :value="modelValue"
      @input="$emit('update:modelValue', ($event.target as HTMLInputElement)?.value ??
'')"
```

```
      @blur="$emit('blur')"
      :class="['form-control', {
          'border-red-500': error
      }]" />
    <span
      class="text-red-400"
      v-if="error">{{ error}}</span>
</template>

<script lang="ts" setup>
    interface Props{
        modelValue?: string | number
        error?: string
        type?: 'text' | 'number'
    }

    withDefaults(defineProps<Props>(),{
        type: 'text'
    })

    defineEmits(['update:modelValue', 'blur'])

</script>

<style scoped>
 @reference 'tailwindcss'

 .form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
tight focus:outline-none;
}
</style>
```

- Usémoslo en admin/views/ProductView.ts
- Usando el script setup podíamos importar los componentes directamente
- Cuando usamos defineComponents necesitamos declarar que componentes va a disponer el componente padre explicitamente con el objeto components

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { defineComponent, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useForm} from 'vee-validate'
import * as yup from 'yup'
import CustomInput from "@/modules/common/components/CustomInput.vue"

const validationSchema = yup.object({
    title: yup.string().required().min(2),
    slug: yup.string().required(),
    description: yup.string().required(),
    price: yup.number().required(),
    stock: yup.number().min(1).required(),
```

```
    gender: yup.string().required().oneOf(['men', 'women', 'kid']),
})

export default defineComponent({
    components:{
        CustomInput //<----- AQUI!
    },
    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {values, defineField, errors} = useForm({
            validationSchema
        })

        const [title, titleAttrs] = defineField('title')
        const [slug, slugAttrs] = defineField('slug')
        const [description, descriptionAttrs] = defineField('description')
        const [price, priceAttrs] = defineField('price')
        const [stock, stockAttrs] = defineField('stock')
        const [gender, genderAttrs] = defineField('gender')
        //TODO: defineArray para las tallas y las imagenes

        watchEffect(()=>{
            if(isError.value &&!isLoading.value){
                router.replace('/admin/products')
                return
            }
        })


        return{
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
```

```
            priceAttrs,
            stock,
            stockAttrs,
            gender,
            genderAttrs,
            errors,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions

        }
    }
})
```

- Ahora ya puedo importar el CustomInput en ProductView.vue

```
<CustomInput
 v-model="title"
 v-bind="titleAttrs"
 :error="errors.title"
/>
```

- Me aparece este warning

```
Vue warn: Extraneous non-emits event listeners (change, input) were passed to
component but could not be automatically inherited because component renders fragment
or text root nodes. If the listener is intended to be a component custom event
listener only, declare it using the "emits" option.
  at <CustomInput modelValue=undefined onUpdate:modelValue=fn
onChange=fn<onChange>  ... >
  at <ProductView productId="3552ca52-56d9-4805-9f20-d721c92250c2"
onVnodeUnmounted=fn<onVnodeUnmounted> ref=Ref< Proxy(Object) > >
  at <RouterView >
  at <AdminLayout onVnodeUnmounted=fn<onVnodeUnmounted> ref=Ref< Proxy(Object) > >
  at <RouterView key=1 >
  at <App>
```

- Cuando se trabaja con este v-model y hay dos elementos retornados en el root (template) da problemas
- Simplemente colocándolos dentro de un div se soluciona

```
<template>
    <div> <!--USO UN DIV PARA QUITAR EL WARNING-->
        <input
          :type="type"
          :value="modelValue"
          @input="$emit('update:modelValue', ($event.target as
```

```
HTMLInputElement)?.value ?? '')"
          @blur="$emit('blur')"
          :class="['form-control', {
              'border-red-500': error
          }]" />
        <span
          class="text-red-400"
          v-if="error">{{ error}}</span>
    </div>
</template>

<script lang="ts" setup>
    interface Props{
        modelValue?: string | number
        error?: string
        type?: 'text' | 'number'
    }

    withDefaults(defineProps<Props>(),{
        type: 'text'
    })

    defineEmits(['update:modelValue', 'blur'])

</script>

<style scoped>
 @reference 'tailwindcss'

 .form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
tight focus:outline-none;
}
</style>
```

- Uso el CustomInput con el resto de campos (slug, precio e inventario)
- La descripción es un text area, crearé un CustomtextArea. Falta el del género
- Para los números uso el modificador v-model.number para transformar de string a número
-

```
<template>
  <div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Producto: <small class="text-blue-500">nombre</small></h1>
    <hr class="my-4" />
  </div>

  <form class="grid grid-cols-1 sm:grid-cols-2 bg-white px-5 gap-5">
    <div class="first-col">
      <!-- Primera parte del formulario -->
      <div class="mb-4">
        <label for="title" class="form-label">Título</label>
          <CustomInput
```

```
            v-model="title"
            v-bind="titleAttrs"
            :error="errors.title"
            />
        </div>

        <div class="mb-4">
          <label for="slug" class="form-label">Slug</label>
            <CustomInput
            v-model="slug"
            v-bind="slugAttrs"
            :error="errors.slug"
            />
        </div>

        <div class="mb-4">
          <label for="description" class="form-label">Descripción</label>
          <textarea
            id="description"
            class="shadow h-32 appearance-none border rounded w-full py-2 px-3 text-
gray-700 leading-tight focus:outline-none focus:shadow-outline"
          ></textarea>
        </div>

        <div class="flex flex-row gap-3">
          <div class="mb-4">
            <label for="price" class="form-label">Precio</label>
              <CustomInput
            v-model.number="price"
            v-bind="priceAttrs"
            :error="errors.price"
            />
          </div>

          <div class="mb-4">
            <label for="stock" class="form-label">Inventario</label>
              <CustomInput
            v-model.number="stock"
            v-bind="stockAttrs"
            :error="errors.stock"
            />
          </div>
        </div>

        <div class="mb-4">
          <label for="sizes" class="form-label">Tallas</label>
          <div class="flex">
              <button
                  v-for="size of allSizes"
                  :key="size"
                  type="button"
                  class="bg-blue-100 p-2 rounded w-14 mr-2 flex-1">
                  {{ size }}
              </button>
          </div>
        </div>
```

```
    </div>

    <!-- Segunda columna -->
    <div class="first-col">
      <label for="stock" class="form-label">Imágenes</label>
      <!-- Row with scrollable horizontal -->
      <div class="flex p-2 overflow-x-auto space-x-8 w-full h-[265px] bg-gray-200
rounded">
        <div class="flex-shrink-0">
          <img src="https://via.placeholder.com/250" alt="imagen" class="w-[250px] h-
[250px]" />
        </div>

        <div class="flex-shrink-0">
          <img src="https://via.placeholder.com/250" alt="imagen" class="w-[250px] h-
[250px]" />
        </div>
      </div>
      <!-- Upload image -->
      <div class="col-span-2 my-2">
        <label for="image" class="form-label">Subir imagen</label>

        <input multiple type="file" id="image" class="form-control" />
      </div>

      <div class="mb-4">
        <label for="stock" class="form-label">Género</label>
        <select class="form-control">
          <option value="">Seleccione</option>
          <option value="kid">Niño</option>
          <option value="women">Mujer</option>
          <option value="men">Hombre</option>
        </select>
      </div>

      <!-- Botón para guardar -->
      <div class="my-4 text-right">
        <button
          type="submit"
          class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded"
        >
          Guardar
        </button>
      </div>
    </div>
  </form>

  <div class="grid grid-cols-2">
  <div class="bg-blue-300">
    {{ values }}
  </div>
  <div class="bg-red-300">
    {{ errors }}
  </div>
</div>
</template>
```

```
<script src="./ProductView.ts" lang="ts"></script>

<style scoped>
    @reference 'tailwindcss'
.form-label {
  @apply block text-gray-700 text-sm font-bold mb-2;
}

.form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none;
}
</style>
```

- Falta ver el submit, bloqueo de botón, CustomtextArea, manejo de selectores (género, tallas)

CustomTextArea y posteo del formulario

- En common/components/CustomtextArea.vue (similar al CustomInput.vue)
- Copio las clases que tiene y se las añado al componente dentro del style scoped

```
<template>
    <div>
        <textarea
          :value="modelValue"
          @input="$emit('update:modelValue', ($event.target as
HTMLTextAreaElement)?.value ?? '')"
          @blur="$emit('blur')"
          :class="['form-control',{
              'border-red-500': error
          }]"></textarea>
        <span
          class="text-red-400"
          v-if="error">{{ error}}</span>
    </div>
</template>

<script lang="ts" setup>
    interface Props{
        modelValue?: string | number
        error?: string

    }

    defineProps<Props>()


    defineEmits(['update:modelValue', 'blur'])

</script>
```

```
<style scoped>
 @reference 'tailwindcss'

 :host {
  display: block;
  width: 100%;
}

 .form-control {
  @apply shadow h-32 appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none
 }
  </style>
```

- Recuerda que para poder renderizar el componente, cuando usamos defineComponent en lugar del script setup hay que declarar el componente
- ProductView.ts

```
export default defineComponent({
    components:{
        CustomInput,
        CustomtextArea
    },

{...code}
})
```

- Lo uso en ProductView.vue

```
<div class="mb-4">
    <label for="description" class="form-label">Descripción</label>
    <CustomtextArea
    v-model="description"
    v-bind="descriptionAttrs"
    :error="errors.description"
    />
</div>
```

- **NOTA:** No se recomienda usar archivos de barril para exportar los archivos .vue
- Para manejar el posteo del formulario es bien sencillo
- Desestructuro el handleSubmit del useForm
- ProductView.ts

```
const {values, defineField, errors, handleSubmit} = useForm({
            validationSchema
        })
```

```
const onSubmit = handleSubmit((value)=>{
            console.log({value})
        })

//retorno onSubmit en el return del setup() para poder usarlo en el ProductView.vue
```

- Ahora ya puedo usarlo en el form. No hace falta hacer el prevent default porque la función se va a encargar de hacer esto
- ProductView.vue

```
<form
    @submit="onSubmit"
    class="grid grid-cols-1 sm:grid-cols-2 bg-white px-5 gap-5">
```

- Faltan el gender, las tallas y las imágenes para poder hacer el submit!
- Luego veremos como cambiar el idioma de los mensajes de error

Selectores y arreglos

- En las tallas quiero hacer clic en cada uno de los botones que se muestran y que se active el color, por lo que va a llevar un poco más de lógica
- Si no existe que no la agregue, si ya está seleccionada que la remueva, etc
- Para el selector de gender no hay nada especial, ya tengo gender y genderAttrs
- Lo conecto con el select

```
<div class="mb-4">
<label for="stock" class="form-label">Género</label>
<select
    v-model="gender"
    v-bind="genderAttrs"
    class="form-control">
    <option value="">Seleccione</option>
    <option value="kid">Niño</option>
    <option value="women">Mujer</option>
    <option value="men">Hombre</option>
</select>
<span class="text-red-400" v-show="errors.gender">
    {{ errors.gender }}</span>
</div>
```

- Veamos como hacer cuando tenemos un arreglo de imágenes
- Para usar useFieldArray tiene que haber un useForm previamente creado
- Puedes tener un componente que una varios componentes, pero cada uno tiene que tener un useForm independiente

- Es una restricción
- Para indicar en el useFieldArray el tipo no hace falta poner corchetes para indicar que es un array, solo pongo string
- Desestructuro fields, lo renombro a images
- Retorno images en el return del setup
- ProductView.ts

```
const {values, defineField, errors, handleSubmit} = useForm({
        validationSchema
    })

const {fields: images} = useFieldArray<string>('images')

//retorno images
```

- Hago uso de un v-for en el componente que muestra las imágenes de ProductView.vue

```
<div
    v-for="image of images"
    :key="image.key"
    :alt="title"
    class="flex-shrink-0">
    <img :src="image.value" alt="imagen" class="w-[250px] h-[250px]" />
</div>
```

- No vemos nada de la misma forma que todavía no vemos cargado el titulo del producto, ni el slug ni nada
- ¿Cómo hacemos para mostrar esa información?
- En ProductView.ts tengo el product que extraigo del useQuery
- ProductView.ts

```
const {data: product, isError, isLoading} = useQuery({
    queryKey: ['product', props.productId ],
    queryFn: ()=> getProductById(props.productId),
    retry: false //que no vuelva a intentarlo si falla
})
```

- Es tarde para establecerlo con initialValues porque cuando TansTackQuery resuelve, el fromulario ya está montado

```
const {values, defineField, errors, handleSubmit} = useForm({
    validationSchema,
```

```
    initialValues: product.value //no funciona
})
```

- Esto funciona si previamente tengo los valores
- Nos falta el array de las tallas
- Lo creo
- ProductView.ts

```
const {fields: sizes} = useFieldArray<string>('sizes')

//retorno sizes en el return del setup()
```

- Usemos @click y pasémosle una función que reciba la talla
- ProductView.vue

```
<div class="flex">
<button
    v-for="size of allSizes"
    :key="size"
    @click="toggleSize(size)"
    type="button"
    class="bg-blue-100 p-2 rounded w-14 mr-2 flex-1">
    {{ size }}
</button>
</div>
```

- Creo la función en ProductView.ts

```
            //desestructuro los métodos remove y push, los renombro a removeSize y
pushSize
const {fields: sizes, remove: removeSize, push: pushSize} =
useFieldArray<string>('sizes')

const toggleSize = (size: string)=>{
    const currentSizes = sizes.value.map(size=>size.value)
    const hasSize = currentSizes.includes(size)

    if(hasSize){ //si ya existe lo elimino (para desmarcar la selección)
        removeSize(currentSizes.indexOf(size)) //hay que mandarle un índice para
poderlo eliminar
    }else{
        pushSize(size)
    }
}
```

```
//retorno la función en el return del setup()
```

- La funcionalidad está
- Ahora falta añadir la clase de tailwind para que cuando esté seleccionado se vea en azul y cuando clico lo desselecciono

Inicializar formulario con valores del producto

- Este watchEffect se encarga de que si no tengo producto o hay un error me devuelva a la pantalla de products
- Está en el ProductView.ts, dentro del setup()

```
watchEffect(()=>{
        if(isError.value &&!isLoading.value){
            router.replace('/admin/products')
            return
        }
    })
```

- Si tenemos un producto podría colocar la lógica dentro de este watchEffect después del if, pero se recomienda que cada watchEffect se encargue solo de una tarea específica
- Puedo usar un watch, que a diferencia del watchEffect dentro tiene sus propiedades reactivas, analizando el objeto del cambio
- Queremos estar pendientes de product
- Si tengo el product.value, puedo poner solo product y no hace falta que coloque ()=> product
- Una vez tenga el producto, voy a querer llamar a la función resetForm que desestructuro del useForm
- Coloco el deep en true para que esté pendiente de las properties internas del objeto
- Si tengo cambios en el usuario, mediante el deep está pendiente también de esos objetos anidados
- El inmediate en true es para que se ejecute tan pronto el componente es construido
- ProductView.ts

```
const {values, defineField, errors, handleSubmit, resetForm} = useForm({
            validationSchema
        })

watch(product, ()=>{
        if(!product) return

        resetForm({
```

```
            values: product.value
        })
    }, {
        deep: true,
        inmediate: true
    })
```

- Formateo la salida del código que estoy mostrando debajo en la pantalla para el desarrollo de los formularios
- Uso JSON.stringify y una etiqueta pre
- **NOTA:** esto es solo para visualizar mejor el json del producto en pantalla y los errores
-
- ProductView.vue

```
<div class="grid grid-cols-2">
  <pre class="bg-blue-300">
    {{ JSON.stringify(values,null, 2) }}
  </pre>
  <pre class="bg-red-300">
    {{ JSON.stringify(errors, null, 2) }}
  </pre>
</div>
```

- Ahora habría que marcar las tallas existentes y darles un color más oscuro para que se vea que están marcadas y
- Hay que poner los errores en castellano
- Deshabilitar el botón de guardar si no están todos los campos sin errores
- VeeValidate tiene sus propios componentes personalizados

Indicador visual de tallas seleccionadas

- En ProductView, en los inputs Precio e inventario agrego flex-1 para evitar que el diseño se desordene
- ProductView.vue

```
<div class="flex flex-row gap-3">
<div class="mb-4 flex-1">
    <label for="price" class="form-label">Precio</label>
    <CustomInput
    v-model.number="price"
    v-bind="priceAttrs"
    :error="errors.price"
    />
</div>
```

```
<div class="mb-4 flex-1">
    <label for="stock" class="form-label">Inventario</label>
    <CustomInput
    v-model.number="stock"
    v-bind="stockAttrs"
    :error="errors.stock"
    />
</div>
</div>
```

- Lo más sencillo para hacer lo de las tallas es crear una función que reciba la talla y nos regrese un valor booleano si la talla existe o no existe
- Si requiero solo la función en el template y no la voy a usar en el script setup, puedo definirla (la función/acción) en el return del setup()
- ProductView.ts

```
export default defineComponent({

{...code}

return{
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
            priceAttrs,
            stock,
            stockAttrs,
            gender,
            genderAttrs,
            errors,
            images,
            sizes,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions
            onSubmit,
            toggleSize,
            hasSize: (size: string)=>{ //<------ AQUI!
                const currentSizes = sizes.value.map(size=>size.value) //aplano
con .map

                return currentSizes.includes(size) //esto devuelve un booleano
            }
```

```
        }
})
```

- Para meter lógica de css le coloco un v-bind al class y entre llaves, primero las clases que siempre van a ir y en un objeto coloco la lógica
- Coloco también la lógica del bg-blue-100 para que las clases de css no compitan y no dejárselo al navegador
- ProductView.vue

```
<div class="mb-4 flex-1">
<label for="sizes" class="form-label">Tallas</label>
<div class="flex">
    <button
        v-for="size of allSizes"
        :key="size"
        @click="toggleSize(size)"
        type="button"
        :class="['p-2 rounded w-14 mr-2 flex-1',{
            'bg-blue-500 text-white': hasSize(size),
            'bg-blue-100': !hasSize(size)

        }]">
        {{ size }}
    </button>
</div>
</div>
```

- En ProductView.vue, donde tenía nombre coloco el title para mostrar el nombre del producto

```
  <div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Producto: <small class="text-blue-500">{{  title }}</small></
h1>
    <hr class="my-4" />
  </div>
```

Metadata del formulario

- Para deshabilitar el botón de guardar si el formulario no es válido necesito acceder a la metadata del formulario
- En el useForm puedo desestructurar meta
- ProductView.ts

```
const {values, defineField, errors, handleSubmit, resetForm, meta} = useForm({
            validationSchema
```

```
        })
//regreso meta en el return del setup()
```

- Renderizo también meta en la pantalla de ProductView.vue para ver
- Uso col-span-2 para que ocupe dos espacios de mi grid

```
<pre class="bg-green-200 p-2 col-span-2">
    {{ JSON.stringify(meta, null, 2) }}
</pre>
```

- Fuera del objeto JSON de producto aparecen estas propiedades (si no toco nada del formulario y lo dejo tal cual lo renderiza por primera vez con la data de producto)

```
"touched": false, // si toco algo del formulario esto cambia a true
"pending": false, //muestra si hay algñún procedimiento asíncrono procesándose
"valid": true,   // en true pasa todas las validaciones
"dirty": false   // cuando el formulario ha sido manipulado se pone en true
```

Errores en español

- En src/config/yup.ts
- No hace falta colocar back tics para usar ${values} (le paso los valores)

```
import * as yup from 'yup'

yup.setLocale({
    mixed: {
        default: 'No es válido',
        required: 'Este campo es requerido',
        oneOf: 'Debe de ser uno de los siguientes valores ${values}' //seleccionado
algo fuera de la lista de valores
    }
})
```

- En el main.ts importo el archivo yup.ts

```
import './assets/styles.css'
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import Toast from 'vue-toastification'
import 'vue-toastification/dist/index.css'
import App from './App.vue'
import router from './router'
import { VueQueryPlugin } from '@tanstack/vue-query'
import './config/yup'
```

```
const app = createApp(App)

app.use(createPinia())
app.use(VueQueryPlugin)
app.use(router)
app.use(Toast)

app.mount('#app')
```

- Ahora si no coloco un título me aparece 'Este campo es requerido'
- En género, si no selecciono nada del select me devuelve 'Debe de ser uno de los siguientes valores' con men, women, kid
- Este es un ejemplo de la configuración típica de estos errores en un archivo para yup

```
import * as yup from 'yup';

// Establecer el idioma de los mensajes de error en español
yup.setLocale({
  mixed: {
    default: 'No es válido',
    required: 'Este campo es requerido',
    oneOf: 'Debe ser uno de los siguientes valores: ${values}',
    notOneOf: 'No debe ser uno de los siguientes valores: ${values}',
    defined: 'Debe estar definido',
    notNull: 'No puede ser nulo',
    notType: 'Debe ser de tipo ${type}',
  },
  string: {
    length: 'Debe tener exactamente ${length} caracteres',
    min: 'Debe tener al menos ${min} caracteres',
    max: 'Debe tener como máximo ${max} caracteres',
    email: 'Debe ser un correo electrónico válido',
    url: 'Debe ser una URL válida',
    trim: 'No debe contener espacios al inicio o al final',
    lowercase: 'Debe estar en minúsculas',
    uppercase: 'Debe estar en mayúsculas',
    matches: 'Debe coincidir con el siguiente patrón: "${regex}"',
  },
  number: {
    min: 'Debe ser mayor o igual a ${min}',
    max: 'Debe ser menor o igual a ${max}',
    lessThan: 'Debe ser menor a ${less}',
    moreThan: 'Debe ser mayor a ${more}',
    positive: 'Debe ser un número positivo',
    negative: 'Debe ser un número negativo',
    integer: 'Debe ser un número entero',
  },
  date: {
    min: 'Debe ser posterior a ${min}',
    max: 'Debe ser anterior a ${max}',
  },
```

```
  array: {
    min: 'Debe tener al menos ${min} elementos',
    max: 'Debe tener como máximo ${max} elementos',
  },
});
```

- En el objeto de yup que he creado en en el ProductView.ts puedo sobreescribir mensajes de error

```
const validationSchema = yup.object({
    title: yup.string().required('Este campo es necesario').min(2),
    slug: yup.string().required(),
    description: yup.string().required(),
    price: yup.number().required(),
    stock: yup.number().min(1).required(),
    gender: yup.string().required().oneOf(['men', 'women', 'kid']),
})
```

- Paso componentes finales de este módulo
- ProductView.vue

```
<template>
  <div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Producto: <small class="text-blue-500">{{  title }}</small></
h1>
    <hr class="my-4" />
  </div>

  <form
    @submit="onSubmit"
    class="grid grid-cols-1 sm:grid-cols-2 bg-white px-5 gap-5">
    <div class="first-col min-w-0">
      <!-- Primera parte del formulario -->
      <div class="mb-4">
        <label for="title" class="form-label">Título</label>
          <CustomInput
          class="block w-full"
          v-model="title"
          v-bind="titleAttrs"
          :error="errors.title"
          />
      </div>

      <div class="mb-4">
        <label for="slug" class="form-label">Slug</label>
          <CustomInput
          v-model="slug"
          v-bind="slugAttrs"
          :error="errors.slug"
          />
      </div>
```

```
      <div class="mb-4">
        <label for="description" class="form-label">Descripción</label>
       <CustomtextArea
        v-model="description"
        v-bind="descriptionAttrs"
        :error="errors.description"
       />
      </div>

      <div class="flex flex-row gap-3">
        <div class="mb-4 flex-1">
          <label for="price" class="form-label">Precio</label>
            <CustomInput
          v-model.number="price"
          v-bind="priceAttrs"
          :error="errors.price"
          />
        </div>

        <div class="mb-4 flex-1">
          <label for="stock" class="form-label">Inventario</label>
            <CustomInput
          v-model.number="stock"
          v-bind="stockAttrs"
          :error="errors.stock"
          />
        </div>
      </div>

      <div class="mb-4 flex-1">
        <label for="sizes" class="form-label">Tallas</label>
        <div class="flex">
            <button
                v-for="size of allSizes"
                :key="size"
                @click="toggleSize(size)"
                type="button"
                :class="['p-2 rounded w-14 mr-2 flex-1',{
                  'bg-blue-500 text-white': hasSize(size),
                  'bg-blue-100': !hasSize(size)

                }]">
                {{ size }}
            </button>
        </div>
      </div>
    </div>

    <!-- Segunda columna -->
    <div class="first-col">
      <label for="stock" class="form-label">Imágenes</label>
      <!-- Row with scrollable horizontal -->
      <div class="flex p-2 overflow-x-auto space-x-8 w-full h-[265px] bg-gray-200
rounded">
        <div
```

```
            v-for="image of images"
            :key="image.key"
            :alt="title"
          class="flex-shrink-0">
            <img :src="image.value" alt="imagen" class="w-[250px] h-[250px]" />
          </div>


        </div>
        <!-- Upload image -->
        <div class="col-span-2 my-2">
          <label for="image" class="form-label">Subir imagen</label>

          <input multiple type="file" id="image" class="form-control" />
        </div>

        <div class="mb-4">
          <label for="stock" class="form-label">Género</label>
          <select
            v-model="gender"
            v-bind="genderAttrs"
            class="form-control">
            <option value="">Seleccione</option>
            <option value="kid">Niño</option>
            <option value="women">Mujer</option>
            <option value="men">Hombre</option>
          </select>
          <span class="text-red-400" v-show="errors.gender">
            {{ errors.gender }}</span>
        </div>

        <!-- Botón para guardar -->
        <div class="my-4 text-right">
          <button
            type="submit"
            class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded"
          >
            Guardar
          </button>
        </div>
      </div>
    </form>

    <div class="grid grid-cols-2">
    <pre class="bg-blue-300">
      {{ JSON.stringify(values,null, 2) }}
    </pre>
    <pre class="bg-red-300">
      {{ JSON.stringify(errors, null, 2) }}
    </pre>
    <pre class="bg-green-200 p-2 col-span-2">
      {{ JSON.stringify(meta, null, 2) }}
    </pre>
  </div>
</template>
```

```
<script src="./ProductView.ts" lang="ts"></script>

<style scoped>
    @reference 'tailwindcss'
.form-label {
  @apply block text-gray-700 text-sm font-bold mb-2;
}

.form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
tight focus:outline-none;
}
</style>
```

• ProductView.ts

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useQuery } from "@tanstack/vue-query"
import { defineComponent, watch, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useFieldArray, useForm} from 'vee-validate'
import * as yup from 'yup'
import CustomInput from "@/modules/common/components/CustomInput.vue"
import CustomTextArea from "@/modules/common/components/CustomTextArea.vue"

const validationSchema = yup.object({
    title: yup.string().required().min(2),
    slug: yup.string().required(),
    description: yup.string().required(),
    price: yup.number().required(),
    stock: yup.number().min(1).required(),
    gender: yup.string().required().oneOf(['men', 'women', 'kid']),
})

export default defineComponent({
    components:{
        CustomInput,
        CustomTextArea
    },
    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading} = useQuery({
            queryKey: ['product', props.productId ],
```

```
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {values, defineField, errors, handleSubmit, resetForm, meta} = useForm({
            validationSchema
        })

        const {fields: images} = useFieldArray<string>('images')
        const {fields: sizes, remove: removeSize, push: pushSize} =
useFieldArray<string>('sizes')

        const toggleSize = (size: string)=>{
            const currentSizes = sizes.value.map(size=>size.value)
            const hasSize = currentSizes.includes(size)

            if(hasSize){ //si existe lo elimino
                removeSize(currentSizes.indexOf(size)) //hay que mandarle un índice
para poderlo eliminar
            }else{
                pushSize(size)
            }
        }

        const onSubmit = handleSubmit((value)=>{
            console.log({value})
        })

        const [title, titleAttrs] = defineField('title')
        const [slug, slugAttrs] = defineField('slug')
        const [description, descriptionAttrs] = defineField('description')
        const [price, priceAttrs] = defineField('price')
        const [stock, stockAttrs] = defineField('stock')
        const [gender, genderAttrs] = defineField('gender')
        //TODO: defineArray para las tallas y las imagenes

        watchEffect(()=>{
            if(isError.value &&!isLoading.value){
                router.replace('/admin/products')
                return
            }
        })

        watch(product, ()=>{
            if(!product) return

            resetForm({
                values: product.value
            })
        }, {
            deep: true,
            immediate: true
        })


        return{
```

```
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
            priceAttrs,
            stock,
            stockAttrs,
            gender,
            genderAttrs,
            errors,
            images,
            sizes,
            meta,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions
            onSubmit,
            toggleSize,
            hasSize: (size: string)=>{
                const currentSizes = sizes.value.map(size=>size.value)

                return currentSizes.includes(size) //esto devuelve un booleano
            }
        }
    }
})
```

• ProductsView.vue

```
<template>

<div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Productos</h1>
    <div class="py-8 w-full">
  <div class="shadow overflow-hidden rounded border-b border-gray-200">
    <table class="min-w-full bg-white">
      <thead class="bg-gray-800 text-white">
        <tr>
          <th class="w-10text-left py-3 px-4 uppercase font-semibold text-sm">Imagen</
th>
          <th class="flex-1 text-left py-3 px-4 uppercase font-semibold text-
sm">Título</th>
          <th class="w-28 text-left py-3 px-4 uppercase font-semibold text-sm">Precio</
th>
          <th class="w-60 text-left py-3 px-4 uppercase font-semibold text-sm">Tallas</
th>
        </tr>
```

```
      </thead>
    <tbody class="text-gray-700">
      <tr
      v-for="(product, index) in products"
            :key="product.id"
            :class="{
                'bg-gray-100': index % 2 === 0 //coloco fondo gris en los pares
            }">
        <td class="text-left py-3 px-4">
            <img
            class="h-10 w-10 object-cover"
            :src="product.images[0]" :alt="product.title" />
        </td>
        <td class="text-left py-3 px-4">
            <router-link
                :to="`/admin/products/${product.id}`"
                class="hover:text-blue-500 hover:underline"
            >{{ product.title }}
            </router-link>
        </td>
        <td class="text-left py-3 px-4">
            <span class="bg-blue-200 text-blue-600 py-1 px-3 rounded-full text-xs">
                {{ product.price }}
            </span>
        </td>
        <td class="text-left py-3 px-4">
            {{ product.sizes.join(',') }}
        </td>
      </tr>
    </tbody>
    </table>
  </div>
  <ButtonPagination
        :page="page"
        :has-more-data="!!products && products.length > 10"
    />
</div>
</div>
</template>

<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue'
import { usePagination } from '@/modules/common/composables/use-pagination'
import { getProductsAction } from '@/modules/products/actions'
import { useQuery, useQueryClient } from '@tanstack/vue-query'
import { watchEffect } from 'vue'

const queryClient = useQueryClient()
const {page} = usePagination()

const {data:products, isLoading} = useQuery({
  queryKey:['products', page],
  queryFn: ()=> getProductsAction(page.value),
})
```

```
watchEffect(()=>{
    queryClient.prefetchQuery({
        queryKey: ['products', {page: page.value +1}],
        queryFn: ()=> getProductsAction(page.value +1),
    })
})
</script>
```

• CustomInput.vue

```
<template>
    <div>
        <input
          :type="type"
          :value="modelValue"
          @input="$emit('update:modelValue', ($event.target as
HTMLInputElement)?.value ?? '')"
          @blur="$emit('blur')"
          :class="['form-control', {
              'border-red-500': error
          }]" />
        <span
          class="text-red-400"
          v-if="error">{{ error}}</span>
    </div>
</template>

<script lang="ts" setup>
    interface Props{
        modelValue?: string | number
        error?: string
        type?: 'text' | 'number'
    }

    withDefaults(defineProps<Props>(),{
        type: 'text'
    })

    defineEmits(['update:modelValue', 'blur'])

</script>

<style scoped>
 @reference 'tailwindcss'

 .form-label {
  @apply block text-gray-700 text-sm font-bold mb-2;
}

.form-control {
  @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
tight focus:outline-none;
}
```

```
</style>
```

• CustomtextArea.vue

```
<template>
    <div>
        <textarea
          :value="modelValue"
          @input="$emit('update:modelValue', ($event.target as
HTMLTextAreaElement)?.value ?? '')"
          @blur="$emit('blur')"
          :class="['form-control',{
              'border-red-500': error
          }]"></textarea>
        <span
          class="text-red-400"
          v-if="error">{{ error}}</span>
    </div>
</template>

<script lang="ts" setup>
    interface Props{
        modelValue?: string | number
        error?: string

    }

    defineProps<Props>()


    defineEmits(['update:modelValue', 'blur'])

</script>

<style scoped>
 @reference 'tailwindcss'

 :host {
  display: block;
  width: 100%;
}

 .form-control {
  @apply shadow h-32 appearance-none border rounded w-full py-2 px-3 text-gray-700
leading-tight focus:outline-none
 }
  </style>
```

• ProductCard.vue

```
<template>
    <article class="rounded-xl bg-white p-3 shadow-lg hover:shadow-xl hover:transform
hover:scale-105 duration-300 ">
      <a href="#">
        <div class="relative flex items-end overflow-hidden rounded-xl">
          <img
          class="h-[250px] object-cover"
          :src="product.images[0]" :alt="product.title" />
        </div>

        <div class="mt-1 p-2">
          <h2 class="text-slate-700 capitalize">{{ product.title }}</h2>
          <p class="mt-1 text-sm text-slate-400">{{ product.gender }}</p>

          <div class="mt-3 flex items-end justify-between">
              <p class="text-lg font-bold text-blue-500">{{ product.price }}</p>
            <div class="flex items-center space-x-1.5 rounded-lg bg-blue-500 px-4
py-1.5 text-white duration-100 hover:bg-blue-600">
              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="h-4 w-4">
                <path stroke-linecap="round" stroke-linejoin="round" d="M2.25 3h1.386c.
51 0 .955.343 1.087.835l.383 1.437M7.5 14.25a3 3 0 00-3
3h15.75m-12.75-3h11.218c1.121-2.3 2.1-4.684 2.924-7.138a60.114 60.114 0
00-16.536-1.84M7.5 14.25L5.106 5.272M6 20.25a.75.75 0 11-1.5 0 .75.75 0 011.5 0zm12.75
0a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
              </svg>

              <button class="text-sm">Add to cart</button>
            </div>
          </div>
        </div>
      </a>
    </article>
</template>


<script lang="ts" setup>
import type { Product } from '../interfaces/products-response.interface';

    interface Props{
        product: Product
    }

    defineProps<Props>()

</script>
```

• ProductList.vue

```
<template>
    <section class="py-10 bg-gray-100">
  <div class="mx-auto grid max-w-6xl  grid-cols-1 gap-6 p-6 sm:grid-cols-2 md:grid-
cols-3 lg:grid-cols-4">
        <ProductCard
```

```
        v-for="product in products" :key="product.id"
        :product="product"
        />
    </div>
</section>
</template>


<script setup lang="ts">
import type { Product } from '../interfaces/products-response.interface';
import ProductCard from './ProductCard.vue';

interface Props{
    products: Product[]
}

defineProps<Props>()

</script>
```

- products/actions/
- get-products.action.ts

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)

        return data.map(product=>({
            ...product,
            images: product.images.map(getProductImageAction)}))

    } catch (error) {
        throw new Error('Error getting products')
    }
}
```

- get-product-image.action.ts

```
export const getProductImageAction =(imageName: string): string=> {

    return imageName.includes('http')
    ? imageName
```

```
    : `${import.meta.env.VITE_TESLO_API_URL}/files/product/${imageName}`
}
```

• get-product-by-id.action.ts

```
import { tesloApi } from "@/api/tesloApi"
import type { Product } from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"
import { isAxiosError } from "axios"

export const getProductById =async (id: string)=>{

    //TODO: pensar la creación de un nuevo producto

    try {
        const {data} = await tesloApi.get<Product>(`/products/${id}`)


        return{
            ...data,
            images: data.images.map(getProductImageAction)
        }
    } catch (error) {
        console.log(error)

        throw new Error('Error obteniendo el producto')
    }
}
```

15 Vue Herrera - Carga de archivos y posteo

- Veremos como manejar la data
- Crearemos una única acción que se encargue de crear y actualizar al mismo tiempo
  - Vamos a poder diferenciar si tenemos un id o no
  - Podemos crear una pantalla diferente para la creación y la actualización (no se hará aquí)
- Crearemos una vista previa de las imágenes antes de enviarlas al backend
- Para crear un nuevo producto le daremos al botón de New
- Usaremos useMutation para hacer la mutación
- Las credenciales para ingresar son email: test1@google.com, password: Abc123

Acción - updateProduct

- El endpoint para la actualización es un PATCH a /api/products/:id
- Nos va a pedir que estemos autenticados, esto lo solucionamos con el interceptor de axios
- src/api/tesloApi.ts

```
import axios from 'axios'


const tesloApi = axios.create({
    baseURL: import.meta.env.VITE_TESLO_API_URL
})

tesloApi.interceptors.request.use((config)=>{
    const token = localStorage.getItem('token')

    const isAuthRoute =
        config.url?.includes('/auth/login') ||
        config.url?.includes('/auth/register')

    if(token && !isAuthRoute){
        config.headers.Authorization = `Bearer ${token}` //coloca el token en los
headers de autorización
    }
    return config
})

export {tesloApi}
```

- En modules/products/actions/create-update-product.action.ts

- Si recibo algo de tipo Product me va a obligar a recibir un usuario y un id, cosa que en el momento de la creación no tengo
- Así mismo en la actualización, quizá solo mande el título o la descripción cambiada
- Para ello uso Partial. Significa que ahora todas las propiedades que tengo en Product son opcionales

```
import type { Product } from "../interfaces/products-response.interface";

export const createUpdateProduct =async (product: Partial<Product>, id?: string)=>{
    if(product.id && product.id != ''){
        //Actualizar producto

    }

    throw new Error('no implementado')
}
```

- Para las imágenes teníamos esta función basada en una variable de entorno (ya que en producción no será localhost:3000)
- products/actions/get-product-image.action.ts

```
export const getProductImageAction =(imageName: string): string=> {

    return imageName.includes('http')
    ? imageName
    : `${import.meta.env.VITE_TESLO_API_URL}/files/product/${imageName}`
}
```

- A la hora de mandarlas debo mandar solo el nombre de la imagen, es decir 928379283723.jpg (sin el localhost:3000)
- create-update-product.action.ts

```
import { tesloApi } from "@/api/tesloApi";
import type { Product } from "../interfaces/products-response.interface";

export const createUpdateProductAction =async (product: Partial<Product>)=>{
    if(product.id && product.id != ''){

        return await updateProduct(product)
    }

    throw new Error('no implementado')
}

const updateProduct = async(product: Partial<Product>)=>{

    const images: string[]= product.images?.map(image=>{
```

```
      if(image.startsWith('http')){
        const imageName = image.split('/').pop() //divido por el slash y extraigo la
última posición

        //imageName puede regresar undefined
        return imageName ? image: '' //solo para asegurarnos de tener en images un
arreglo de strings
      }

      return image
    }) ?? []

    //esta parte es por como está organizado el backend
    const productId = product.id
    delete product.id //en la actualización no se espera que le mandemos el product.id
    delete product.user //el usuario va a estar en el token, no lo mando
    product.images = images //le paso el arreglo de strings de imágenes

    //mandamos la data al backend
    try {

        const {data} = await tesloApi.patch(`/products/${productId}`, product)

        return data

    } catch (error) {
        console.log(error)
        throw new Error('Error updating product')
    }
}
```

useMutation - Actualizar producto

- La mutación no es más que realizar un cambio en nuestra data
- En admin/views/ProductView.vue, cuando llamamos al posteo del formulario en @submit=”onSubmit” llamamos a esta función onSubmit
- En ProductView.ts tenemos esta función, que lo único que hace es llamar al handleSubmit del useForm
- ProductView.ts

```
//dentro del setup

const {values, defineField, errors, handleSubmit, resetForm, meta} = useForm({
            validationSchema
        })

const onSubmit = handleSubmit((value)=>{
```

```
            console.log({value})
        })
```

- A menos de que todos los campos requeridos estén, esta función no se dispara
- Esta es la data que le tenemos que mandar a nuestra acción (el value del handleSubmit)
- ¿Porqué voy a querer usar una mutación en lugar de llamar directamente a la acción en el onSubmit?
- Esto funcionaría, pero yo voy a querer saber si hay un error, obtener la data actualizada, tener el isLoading
- El useMutation es muy similar al useQuery, pero usa el mutationFn
- Le paso la función por referencia (si recibiera dos argumentos, el segundo argumento sería el context)
- renombro la data y el isSuccess
- ProductView.ts

```
//dentro del setup

const {mutate, isPending, isSuccess: isUpdateSuccess, data: updatedProduct} =
useMutation({
        mutationFn: createUpdateProductAction
    })
```

- El mutate es la referencia a nuestra función mutationFn
- Cuando está definido el useMutation no quiere decir que se va a realizar la petición como sucede con el useQuery
- Cuando quiero disparar la función porque tengo la data actualizada llamo a mutate
- La llamo en el onSubmit
- ProductView.ts

```
//dentro del setup

const onSubmit = handleSubmit(async (values)=>{
    mutate(values) //regresa void
})
```

- Si hago un cambio en el titulo y le doy a guardar, no veo nada. Pero si recargo el navegador, los cambios persisten. ha hecho el posteo de la data correctamente
- Vamos a querer tener un feedback, si algo salió bien, si algo salió mal
- Lo podemos hacer con un simple watch
- ProductView.ts

```
```

```
//dentro del setup

watch(isUpdateSuccess, ()=>{
            console.log({isUpdateSuccess})
        })
```

- Esto, cuando le doy a guardar y sale bien me devuelve un objeto como este

```
{isUpdateSuccess: ObjectRefImpl}
isUpdateSuccess:
ObjectRefImpl
__v_isRef: true //referencia reactiva, apunta a true cuando ha salido bien
_defaultValue: undefined
_key: "isSuccess"
_object: Proxy(Object) {context: undefined, data: {…}, error: null, failureCount: 0,
failureReason: null, …}
_raw:
{context: undefined, data: {…}, error: null, failureCount: 0, failureReason: null, …}
_shallow: false
_value: true
dep: (...)
value: (...)
[[Prototype]]: Object
[[Prototype]]: Object
```

- Si hago un cambio y clico dos veces el botón de Guardar sin hacer un segundo cambio, el __v__isRef, que es la referencia reactiva apunta a false
- En lugar de usar el isUpdateSuccess.value puedo obtener el value en el callback
- ProductView.ts

```
//dentro del setup

const toast = useToast()


const {mutate, isPending, isSuccess:isUpdateSuccess, data: updatedProduct} =
useMutation({
            mutationFn: createUpdateProductAction
        })

watch(isUpdateSuccess, (value)=>{
    if(!value) return //si el isUpdatedSuccess.value es false no hago nada

    toast.success('Producto actualizado correctamente')

    //TODO: redirección cuando se crea

    resetForm({
        //reestablezco los valores del formulario
        values: updatedProduct.value //updatedProduct es la data del useMutation
```

```
    })

})
```

- Podemos ocultar el botón para evitar un doble posteo
- Uso el isPending en el componente. Para ello lo retorno en la función setup
- ProductView.ts

```
<button
    :disabled="isPending"
    type="submit"

class="disabled:bg-gray-300 bg-blue-500 hover:bg-blue-700 text-white font-bold py-2
px-4 rounded"
>Guardar
</button>
```

Crear nuevo producto

- Copio el updateProduct de modules/products/actions/create-update-product.action.ts y la renombro a createProduct
- No es una petición patch sino post, el path es /products
- create-update-product.action.ts

```
export const createUpdateProductAction =async (product: Partial<Product>)=>{
    if(product.id && product.id != ''){

        return await updateProduct(product)
    }

    return await createProduct(product)
}

//updateProduct

const createProduct = async(product: Partial<Product>)=>{

    const images: string[]= product.images?.map(image=>{
      if(image.startsWith('http')){
        const imageName = image.split('/').pop()


        return imageName ? image: ''
      }

      return image
    }) ?? []
```

```
    const productId = product.id
    delete product.id
    delete product.user
    product.images = images

    try {
        const {data} = await tesloApi.post(`/products`, product)

        return data

    } catch (error) {
        console.log(error)
        throw new Error('Error creating product')
    }
}
```

- En el AdminLayout es donde tengo el botón de Nuevo Producto

```
<a class="flex items-center flex-shrink-0 h-10 px-3 mt-auto text-sm font-medium bg-gray-200 rounded hover:bg-gray-300"
    href="#">
    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
    </svg>
    <span class="ml-2 leading-none">Nuevo Producto</span>
</a>
```

- Cambiamos el anchor tag por un RouterLink
- Apunto a /admin/products/create. Le pongo algo como create porque necesita un id

```
<RouterLink class="flex items-center flex-shrink-0 h-10 px-3 mt-auto text-sm font-medium bg-gray-200 rounded hover:bg-gray-300"
to="/admin/products/create">
<svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
    <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
</svg>
<span class="ml-2 leading-none">Nuevo Producto</span>
</RouterLink>
```

- Si clico en Nuevo Producto, en la pantalla de Products me da un 404
- Cuando estoy en un producto (editando) si me deja entrar, me cambia el url, el producto se queda en pantalla pero no se está volviendo a disparar la petición y demás
- Vayamos a la pantalla de Product, que cuando aprieto en Nuevo Producto no me deja entrar (error 404)

- En get-product-by-id.action.ts hago un console.log del id

```
import { tesloApi } from "@/api/tesloApi"
import type { Product } from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"
import { isAxiosError } from "axios"

export const getProductById =async (id: string)=>{

    //TODO: pensar la creación de un nuevo producto

    console.log({id})

    try {
        const {data} = await tesloApi.get<Product>(`/products/${id}`)


        return{
            ...data,
            images: data.images.map(getProductImageAction)
        }
    } catch (error) {
        console.log(error)

        throw new Error('Error obteniendo el producto')
    }
}
```

- En consola me muestra create
- De esta manera podemos determinar que cuando recibo create como id quiero crear un producto y si necesito entrar

```
import { tesloApi } from "@/api/tesloApi"
import type { Product } from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

                                    //productId
export const getProductById =async (id: string) : Promise<Product>=>{

   if(id === 'create'){
    return {
        id: '',
        title: '',
        slug: '',
        description: '',
        price: 0,
        stock: 0,
        images: [],
        tags: [],
        sizes: [],
        gender: '' as any,
        user: {} as any
```

```
    }}

    try {
        const {data} = await tesloApi.get<Product>(`/products/${id}`)


        return{
            ...data,
            images: data.images.map(getProductImageAction)
        }
    } catch (error) {
        console.log(error)

        throw new Error('Error obteniendo el producto')
    }
}
```

- Si estoy en Products y clico Nuevo Producto aparece el formulario vacío
- Pero si clico en Nuevo Producto desde la pantalla de edición de un producto no pasa nada, porque no estoy reaccionando cuando ese argumento por el url cambia
- En admin/views/ProductView.ts (de ProductView.vue, que es la pantalla desde donde editamos el producto) tenemos que estar pendientes de ese argumento por el url
- Cuando este cambie tenemos que volver a disparar la petición del query (useQuery) basado en las props
- Uso un watch. Si accedes a una propiedad de un objeto (props, reactive) → usa función
- En el watch con isUpdateSuccess ya es un ref o computed, por lo que no hace falta usar ()=>
- El refectch del useQuery es para volver a ejecutar la función
- Si todo sale bien, debo redireccionar a la url de ese producto (desde el watch con isUpdateSucces)
- ProductView.ts

```
//dentro del setup

const {data: product, isError, isLoading, refetch} = useQuery({
    queryKey: ['product', props.productId ],
    queryFn: ()=> getProductById(props.productId),
    retry: false //que no vuelva a intentarlo si falla
})

//Si accedes a una propiedad de un objeto (props, reactive) → usa función
watch(()=> props.productId, ()=>{
    refetch()
})

// isUpdateSuccess ya es un ref o computed, por lo que no hace falta usar ()=>
 watch(isUpdateSuccess, (value)=>{
    if(!value) return //si el isUpdatedSuccess.value es false no hago nada

    toast.success('Producto actualizado correctamente')
```

```
    //redirección, updatedProduct es la data del useMutation
    router.replace(`/admin/products/${updatedProduct.value.id}`)

    resetForm({
        values: updatedProduct.value //updatedProduct es la data del useMutation
    })

})
```

- Ahora si, si clico en Nuevo Producto desde la pantalla de edición de Producto se vacía todo el formulario
- Si agrego los campos al formulario y hago el posteo con Guardar, hace el posteo, en la url aparece el ID del producto y los campos del formulario se quedan con la información que he puesto (que ya está en la DB)

Evitar duplicidad de código

- En la acción de create-update-product.action.ts tenemos código duplicado. Mejorémoslo

```
import { tesloApi } from "@/api/tesloApi";
import type { Product } from "../interfaces/products-response.interface";

export const createUpdateProductAction =async (product: Partial<Product>)=>{
     const productId = product.id //aqui para tenerlo accesible
     product = clearProductForCreateObject(product) //lamo a la función

    if(productId && productId != ''){

        return await updateProduct(productId, product)
    }

    return await createProduct(product)
}

const clearProductForCreateObject = (product: Partial<Product>)=>{
    const images: string[]= product.images?.map(image=>{
      if(image.startsWith('http')){
        const imageName = image.split('/').pop()


        return imageName ? image: ''
      }

      return image
    }) ?? []


    const productId = product.id
    delete product.id
    delete product.user
```

```
    product.images = images

    return product
}
                            //le paso el productId
const updateProduct = async(productId: string, product: Partial<Product>)=>{

    try {

        const {data} = await tesloApi.patch(`/products/${productId}`, product)

        return data

    } catch (error) {
        console.log(error)
        throw new Error('Error updating product')
    }
}

const createProduct = async(product: Partial<Product>)=>{

    try {
        const {data} = await tesloApi.post(`/products`, product)

        return data

    } catch (error) {
        console.log(error)
        throw new Error('Error creating product')
    }
}
```

File Selector - Mostrar imágenes antes de cargarlas

- Nos situamos en la página de producto, donde se edita, en un producto que ya tenga fotografías
- Si le doy a Subir Imagen me permite seleccionar cualquier cosa (pdfs, etc) de mi ordenador
- Me gustaría tener una pvista previa de los archivos en el navegador (donde se muestran las fotografías de los productos al editar)
- Hay otra cosa que pasa con los File Selectors que cuando lo vuelvo a tocar y cancelo, limpia los archivos que ya había cargado. No quiero que esto suceda
- El primer paso que necesito es poder alamecenar lo que sea que este File Selector selecciona
- En ProductView.ts creo una propiedad reactiva imageFiles
- No hace falta importar el tipo File, viene en Typescript
- ProuctView.ts

```
//dentro del setup
```

```
const imageFiles = ref<File[]>([])

//lo retorno en el return del setup para poder usarla en ProductView.vue
```

- En ProductView.vue restringo que el input solo reciba imágenes

```
<div class="col-span-2 my-2">
<label for="image" class="form-label">Subir imagen</label>

<input
    @change="onFileChanged"
    accept="image/*"
    multiple
    type="file"
    id="image"
    class="form-control" />
</div>
```

- Defino esta función en ProductView.ts dentro de la función setup y la retorno para pdoer usarla en ProductView.vue
- Recibe un evento de tipo Event (no hace falta importarlo)

```
//dentro de setup

const imageFiles = ref<File[]>([])

const onFileChanged =(event: Event)=>{
    console.log({event})
}

//la retrono para usarla en el componente
```

- El evento trae los archivos

```
const onFileChanged =(event: Event)=>{
    const fileInput = event.target as HTMLInputElement
    const fileList = fileInput.files
    console.log(fileList)
}
```

- Cuando he cargado variso archivos y vuelvo a darle al botón de subir archivo y cancelo me purga los archivos, vuelve a disparar el evento y pone el fileList con length:0
- Para mantener las imágenes previamente cargadas

```
const onFileChanged =(event: Event)=>{
    const fileInput = event.target as HTMLInputElement
```

```
    const fileList = fileInput.files //los archivos seleccionados para cargar

    // si no he seleccionado nada y cancelo, return
    if( !fileList) return
    if(fileList.length === 0) return

    for(const imageFile of fileList){
        //lo que sea que seleccionemos lo irá metiendo en el arreglo
        imageFiles.value.push(imageFile)
    }
}
```

- Para mostrar las imágenes voy a ProductView.vue, al div donde se muestran las imágenes, y barro el imageFiles con un v-for

```
<div class="first-col">
    <label for="stock" class="form-label">Imágenes</label>
    <!-- Row with scrollable horizontal -->
    <div class="flex p-2 overflow-x-auto space-x-8 w-full h-[265px] bg-gray-200
rounded">
    <div
        v-for="image of images"
        :key="image.key"
        :alt="title"
    class="flex-shrink-0">
        <img :src="image.value" alt="imagen" class="w-[250px] h-[250px]" />
    </div>
    <div
        v-for="imageFile of imageFiles "
        :key="imageFile.name"
        :alt="title"
    class="flex-shrink-0">
        <img :src="" alt="imagen" class="w-[250px] h-[250px]" />
    </div>
```

- Para el src de la imagen tenemos que crear una url. Será una función temporalImageUrl, la creo en ProductView.ts
- La puedo definir en el return del setup
- Uso URL de JavaScript para crear la URL que necesito

```
temporalImageUrl: (imageFile: File)=>{
    return URL.createObjectURL(imageFile)
}
```

- Puedo llamarla en el componente

```
<div
    v-for="imageFile of imageFiles "
```

```
    :key="imageFile.name"
    :alt="title"
    class="flex-shrink-0">
    <img :src="temporalImageUrl(imageFile)" class="w-[250px] h-[250px]" />
</div>
```

- Ahora si cargo dos fotos ya aparecen haciendo scroll horizontal en la vista previa
- Si vuelvo a clicar y cancelo, las fotos siguen ahi en la vista previa
- Si recargo el navegador si se eliminan
- Ahora debo pasarle las imágenes al mutation. Puedo elegir las imágenes a cargar, aquellas que no tengan http, porque significará que no están en el backend

POSTMAN - Carga de imagen

- Abrimos POSTMAN
- Hay varias formas de cargar una imagen, y todo depende de cómo esté el backend configurado
- El backend sube las imágenes una por una
- Las coloca en el filesystem, que no sería el lugar correcto. Lo correcto sería la nube

> /api/files/product

- En Body/form-data selecciono una imagen (el backend solo me permite una)
- En la Key le pongo file. Es la llave que necesito
- El backend nos regresa la secureUrl con “http://localhost:3000/api/files/product/9287392833.jpeg”
- Si copi la url y la pego en la barra de navegación de POSTMAN puedo visualizar la imagen que acabo de subir
- En creat-update-product.action.ts creo la función que luego mejoraré

```
const uploadImage = async (images: string[] | File[])=>{
    const imageFile = images[0] as File

    try {
        const formData = new FormData() //creo el objeto formData
        formData.append('file', imageFile) //le paso el key y la imagen

        //obtengo la data del poste
        const {data}= await tesloApi.post<{secureUrl: string}>('/files/product',
formData)

        return data.secureUrl //si va bien devuelvo la url de la imagen en el backend
    } catch (error) {
        console.log(error)
        throw new Error('Error uploading image')
```

```
    }
}
```

Cargar imágenes al guardar producto

- tenemos que ver como ocnectamos todo esto. Como vamos a llamar la función, como vamos a recibir las imágenes, y a la función le falta una implementación todavía
- Vayamos al onSubmit, donde llamamos a llamar el mutate con los valores del formulario
- A mutate solo le podemos mandar un argumento (el segundo es el context)
- Ese único argumento es el que le pasamos a createUpdateProductAction
- En product.images tenemos las imágenes que ya tenemos y también hay que colocar ahí los archivos que quiero subir
- ProductView.ts

```
//dentro del setup

const onSubmit = handleSubmit(async (values)=>{

    const formValues = {
        ...values,
        images: [...values.images, ...imageFiles.value]
    }

    mutate(formValues)
})
```

- En create-update-product.action.ts necesito preparar mis imágenes

```
export const createUpdateProductAction =async (product: Partial<Product>)=>{
     const productId = product.id

     const newImages = await uploadImages(product.images ?? [])

     product.images = newImages


     product = clearProductForCreateObject(product)

    if(productId && productId != ''){

        return await updateProduct(productId, product)
    }

    return await createProduct(product)
}
```

```
const uploadImages = async (images: string[] | File[])=>{
    const filesToUpload = images.filter(image=>image instanceof File) as File[]
    const currentImages = images.filter(image=>typeof image === 'string') as string[]

    const uploadPromises = filesToUpload.map(async(file)=>{
        try {
            const formData = new FormData()
            formData.append('file', file)
            const {data}= await tesloApi.post<{secureUrl: string}>('/files/product',
formData)

            return data.secureUrl
        } catch (error) {
            console.log(error)
            throw new Error('Error uploading image')
        }
    })

    //necesito esperar que todas estas promesas se resuelvan
    const uploadedImages = await Promise.all(uploadPromises)

    return [...currentImages, ...uploadedImages]

}
```

- Paso algunos archivos completos importantes
- ProductView.vue

```
<template>
  <div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Producto: <small class="text-blue-500">{{  title }}</small></
h1>
    <hr class="my-4" />
  </div>

  <form
    @submit="onSubmit"
    class="grid grid-cols-1 sm:grid-cols-2 bg-white px-5 gap-5">
    <div class="first-col min-w-0">
      <!-- Primera parte del formulario -->
      <div class="mb-4">
        <label for="title" class="form-label">Título</label>
          <CustomInput
          class="block w-full"
          v-model="title"
          v-bind="titleAttrs"
          :error="errors.title"
          />
      </div>

      <div class="mb-4">
        <label for="slug" class="form-label">Slug</label>
          <CustomInput
```

```
          v-model="slug"
          v-bind="slugAttrs"
          :error="errors.slug"
          />
      </div>

      <div class="mb-4">
        <label for="description" class="form-label">Descripción</label>
       <CustomtextArea
        v-model="description"
        v-bind="descriptionAttrs"
        :error="errors.description"
       />
      </div>

      <div class="flex flex-row gap-3">
        <div class="mb-4 flex-1">
          <label for="price" class="form-label">Precio</label>
            <CustomInput
          v-model.number="price"
          v-bind="priceAttrs"
          :error="errors.price"
          />
        </div>

        <div class="mb-4 flex-1">
          <label for="stock" class="form-label">Inventario</label>
            <CustomInput
          v-model.number="stock"
          v-bind="stockAttrs"
          :error="errors.stock"
          />
        </div>
      </div>

      <div class="mb-4 flex-1">
        <label for="sizes" class="form-label">Tallas</label>
        <div class="flex">
            <button
                v-for="size of allSizes"
                :key="size"
                @click="toggleSize(size)"
                type="button"
                :class="['p-2 rounded w-14 mr-2 flex-1',{
                  'bg-blue-500 text-white': hasSize(size),
                  'bg-blue-100': !hasSize(size)

                }]">
                {{ size }}
            </button>
        </div>
      </div>
    </div>

    <!-- Segunda columna -->
    <div class="first-col">
```

```
      <label for="stock" class="form-label">Imágenes</label>
      <!-- Row with scrollable horizontal -->
      <div class="flex p-2 overflow-x-auto space-x-8 w-full h-[265px] bg-gray-200
rounded">
        <div
          v-for="image of images"
          :key="image.key"
          :alt="title"
        class="flex-shrink-0">
          <img :src="image.value" alt="imagen" class="w-[250px] h-[250px]" />
        </div>
        <div
          v-for="imageFile of imageFiles "
          :key="imageFile.name"
          :alt="title"
        class="flex-shrink-0">
          <img :src="temporalImageUrl(imageFile)" class="w-[250px] h-[250px]" />
        </div>


      </div>
      <!-- Upload image -->
      <div class="col-span-2 my-2">
        <label for="image" class="form-label">Subir imagen</label>

        <input
          @change="onFileChanged"
          accept="image/*"
          multiple
          type="file"
          id="image"
          class="form-control" />
      </div>

      <div class="mb-4">
        <label for="stock" class="form-label">Género</label>
        <select
          v-model="gender"
          v-bind="genderAttrs"
          class="form-control">
          <option value="">Seleccione</option>
          <option value="kid">Niño</option>
          <option value="women">Mujer</option>
          <option value="men">Hombre</option>
        </select>
        <span class="text-red-400" v-show="errors.gender">
          {{ errors.gender }}</span>
      </div>

      <!-- Botón para guardar -->
      <div class="my-4 text-right">
        <button
          :disabled="isPending"
          type="submit"
          class="disabled:bg-gray-300 bg-blue-500 hover:bg-blue-700 text-white font-
bold py-2 px-4 rounded"
```

```
          >
            Guardar
          </button>
        </div>
      </div>
    </form>

    <div class="grid grid-cols-2">
    <pre class="bg-blue-300">
      {{ JSON.stringify(values,null, 2) }}
    </pre>
    <pre class="bg-red-300">
      {{ JSON.stringify(errors, null, 2) }}
    </pre>
    <pre class="bg-green-200 p-2 col-span-2">
      {{ JSON.stringify(meta, null, 2) }}
    </pre>
  </div>
  </template>

  <script src="./ProductView.ts" lang="ts"></script>

  <style scoped>
      @reference 'tailwindcss'
  .form-label {
    @apply block text-gray-700 text-sm font-bold mb-2;
  }

  .form-control {
    @apply shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-
  tight focus:outline-none;
  }
  </style>
```

• ProductView.ts

```
import { getProductById } from "@/modules/products/actions/get-product-by-id.action"
import { useMutation, useQuery } from "@tanstack/vue-query"
import { defineComponent, ref, watch, watchEffect } from "vue"
import { useRouter } from "vue-router"
import {useFieldArray, useForm} from 'vee-validate'
import * as yup from 'yup'
import CustomInput from "@/modules/common/components/CustomInput.vue"
import CustomTextArea from "@/modules/common/components/CustomTextArea.vue"
import { createUpdateProductAction } from "@/modules/products/actions/create-update-
product.action"
import { useToast } from "vue-toastification"

const validationSchema = yup.object({
    title: yup.string().required().min(2),
    slug: yup.string().required(),
    description: yup.string().required(),
    price: yup.number().required(),
    stock: yup.number().min(1).required(),
```

```
    gender: yup.string().required().oneOf(['men', 'women', 'kid']),
})

export default defineComponent({
    components:{
        CustomInput,
        CustomTextArea
    },
    props:{
        productId:{
            type: String,
            required: true
        }
    },

    setup(props){

        const router = useRouter()
        const toast = useToast()

        //defineProps() solo funciona en el scriopt setup

        const {data: product, isError, isLoading, refetch} = useQuery({
            queryKey: ['product', props.productId ],
            queryFn: ()=> getProductById(props.productId),
            retry: false //que no vuelva a intentarlo si falla
        })

        const {mutate, isPending, isSuccess:isUpdateSuccess, data: updatedProduct} =
useMutation({
            mutationFn: createUpdateProductAction
        })

        const {values, defineField, errors, handleSubmit, resetForm, meta} = useForm({
            validationSchema
        })

        const {fields: images} = useFieldArray<string>('images')
        const {fields: sizes, remove: removeSize, push: pushSize} =
useFieldArray<string>('sizes')

        const toggleSize = (size: string)=>{
            const currentSizes = sizes.value.map(size=>size.value)
            const hasSize = currentSizes.includes(size)

            if(hasSize){ //si existe lo elimino
                removeSize(currentSizes.indexOf(size)) //hay que mandarle un índice
para poderlo eliminar
            }else{
                pushSize(size)
            }
        }

        const onSubmit = handleSubmit(async (values)=>{

            const formValues = {
```

```
            ...values,
            images: [...values.images, ...imageFiles.value]
        }

        mutate(formValues)
    })

    const [title, titleAttrs] = defineField('title')
    const [slug, slugAttrs] = defineField('slug')
    const [description, descriptionAttrs] = defineField('description')
    const [price, priceAttrs] = defineField('price')
    const [stock, stockAttrs] = defineField('stock')
    const [gender, genderAttrs] = defineField('gender')

    const imageFiles = ref<File[]>([])

    const onFileChanged =(event: Event)=>{
        const fileInput = event.target as HTMLInputElement
        const fileList = fileInput.files //los archivos seleccionados para cargar

        // si no he seleccionado nada y cancelo, return
        if( !fileList) return
        if(fileList.length === 0) return

        for(const imageFile of fileList){
            //lo que sea que seleccionemos lo irá metiendo en el arreglo
            imageFiles.value.push(imageFile)
        }
    }


    watchEffect(()=>{
        if(isError.value &&!isLoading.value){
            router.replace('/admin/products')
            return
        }
    })

    watch(product, ()=>{
        if(!product) return

        resetForm({
            values: product.value
        })
    }, {
        deep: true,
        immediate: true
    })

    watch(isUpdateSuccess, (value)=>{
        if(!value) return //si el isUpdatedSuccess.value es false no hago nada

        toast.success('Producto actualizado correctamente')

        //redirección, updatedProduct es la data del useMutation
        router.replace(`/admin/products/${updatedProduct.value.id}`)
```

```
            resetForm({
                values: updatedProduct.value //updatedProduct es la data del
useMutation
            })

        })

        //Si accedes a una propiedad de un objeto (props, reactive) → usa función
        watch(()=> props.productId, ()=>{
            refetch()
        })


        return{
            // Properties
            values,
            title,
            titleAttrs,
            slug,
            slugAttrs,
            description,
            descriptionAttrs,
            price,
            priceAttrs,
            stock,
            stockAttrs,
            gender,
            genderAttrs,
            errors,
            images,
            sizes,
            meta,
            isPending,
            imageFiles,

            // Getters
            allSizes: ['XS', 'S', 'M', 'L', 'XL', 'XXL'],

            // Actions
            onSubmit,
            toggleSize,
            hasSize: (size: string)=>{
                const currentSizes = sizes.value.map(size=>size.value)

                return currentSizes.includes(size) //esto devuelve un booleano
            },
            onFileChanged,
            temporalImageUrl: (imageFile: File)=>{
                return URL.createObjectURL(imageFile)
            }
        }
```

```
    }
})
```

• create-update-product.action.ts

```
import { tesloApi } from "@/api/tesloApi";
import type { Product } from "../interfaces/products-response.interface";

export const createUpdateProductAction =async (product: Partial<Product>)=>{
     const productId = product.id

     const newImages = await uploadImages(product.images ?? [])

     product.images = newImages


     product = clearProductForCreateObject(product)

    if(productId && productId != ''){

        return await updateProduct(productId, product)
    }

    return await createProduct(product)
}

const clearProductForCreateObject = (product: Partial<Product>)=>{
    const images: string[]= product.images?.map(image=>{
      if(image.startsWith('http')){
        const imageName = image.split('/').pop()


        return imageName ? image: ''
      }

      return image
    }) ?? []


    const productId = product.id
    delete product.id
    delete product.user
    product.images = images

    return product
}
                            //le paso el productId
const updateProduct = async(productId: string, product: Partial<Product>)=>{

    try {

        const {data} = await tesloApi.patch(`/products/${productId}`, product)

        return data
```

```
    } catch (error) {
        console.log(error)
        throw new Error('Error updating product')
    }
}

const createProduct = async(product: Partial<Product>)=>{

    try {
        const {data} = await tesloApi.post(`/products`, product)

        return data

    } catch (error) {
        console.log(error)
        throw new Error('Error creating product')
    }
}


const uploadImages = async (images: string[] | File[])=>{
    const filesToUpload = images.filter(image=>image instanceof File) as File[]
    const currentImages = images.filter(image=>typeof image === 'string') as string[]

    const uploadPromises = filesToUpload.map(async(file)=>{
        try {
            const formData = new FormData()
            formData.append('file', file)
            const {data}= await tesloApi.post<{secureUrl: string}>('/files/product',
formData)

            return data.secureUrl
        } catch (error) {
            console.log(error)
            throw new Error('Error uploading image')
        }
    })

    //necesito esperar que todas estas promesas se resuelvan
    const uploadedImages = await Promise.all(uploadPromises)

    return [...currentImages, ...uploadedImages]

}
```

• get-product-by-id.action.ts

```
import { tesloApi } from "@/api/tesloApi"
import type { Product } from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"


export const getProductById =async (id: string) : Promise<Product>=>{
```

```
   if(id === 'create'){
    return {
        id: '',
        title: '',
        slug: '',
        description: '',
        price: 0,
        stock: 0,
        images: [],
        tags: [],
        sizes: [],
        gender: '' as any,
        user: {} as any
    }}

    try {
        const {data} = await tesloApi.get<Product>(`/products/${id}`)


        return{
            ...data,
            images: data.images.map(getProductImageAction)
        }
    } catch (error) {
        console.log(error)

        throw new Error('Error obteniendo el producto')
    }
}
```

• get-products-action.ts

```
import { tesloApi } from "@/api/tesloApi"
import type { Product} from "../interfaces/products-response.interface"
import { getProductImageAction } from "./get-product-image.action"

export const getProductsAction =async (page: number=1, limit: number= 10)=>{
    try {

        const {data} = await tesloApi.get<Product[]>(`/products?limit=${limit}&offset=$
{page * limit}`)

        return data.map(product=>({
            ...product,
            images: product.images.map(getProductImageAction)}))

    } catch (error) {
        throw new Error('Error getting products')
```

```
    }
}
```

• ProductsView.vue

```
<template>

<div class="bg-white px-5 py-2 rounded">
    <h1 class="text-3xl">Productos</h1>
    <div class="py-8 w-full">
  <div class="shadow overflow-hidden rounded border-b border-gray-200">
    <table class="min-w-full bg-white">
      <thead class="bg-gray-800 text-white">
        <tr>
          <th class="w-10text-left py-3 px-4 uppercase font-semibold text-sm">Imagen</
th>
          <th class="flex-1 text-left py-3 px-4 uppercase font-semibold text-
sm">Título</th>
          <th class="w-28 text-left py-3 px-4 uppercase font-semibold text-sm">Precio</
th>
          <th class="w-60 text-left py-3 px-4 uppercase font-semibold text-sm">Tallas</
th>
        </tr>
      </thead>
    <tbody class="text-gray-700">
      <tr
      v-for="(product, index) in products"
            :key="product.id"
            :class="{
                'bg-gray-100': index % 2 === 0 //coloco fondo gris en los pares
            }">
        <td class="text-left py-3 px-4">
            <img
            class="h-10 w-10 object-cover"
            :src="product.images[0]" :alt="product.title" />
        </td>
        <td class="text-left py-3 px-4">
            <router-link
                :to="`/admin/products/${product.id}`"
                class="hover:text-blue-500 hover:underline"
            >{{ product.title }}
            </router-link>
        </td>
        <td class="text-left py-3 px-4">
            <span class="bg-blue-200 text-blue-600 py-1 px-3 rounded-full text-xs">
                {{ product.price }}
            </span>
        </td>
        <td class="text-left py-3 px-4">
            {{ product.sizes.join(',') }}
        </td>
      </tr>
    </tbody>
    </table>
```

```
  </div>
  <ButtonPagination
        :page="page"
        :has-more-data="!!products && products.length > 10"
    />
</div>
</div>
</template>

<script lang="ts" setup>
import ButtonPagination from '@/modules/common/components/ButtonPagination.vue'
import { usePagination } from '@/modules/common/composables/use-pagination'
import { getProductsAction } from '@/modules/products/actions'
import { useQuery, useQueryClient } from '@tanstack/vue-query'
import { watchEffect } from 'vue'

const queryClient = useQueryClient()
const {page} = usePagination()

const {data:products, isLoading} = useQuery({
  queryKey:['products', page],
  queryFn: ()=> getProductsAction(page.value),
})


watchEffect(()=>{
    queryClient.prefetchQuery({
        queryKey: ['products', {page: page.value +1}],
        queryFn: ()=> getProductsAction(page.value +1),
    })
})
</script>
```

• AdminLayout.vue

```
<template>
<div class="flex w-screen h-screen text-gray-700">
            <div class="flex flex-col items-center w-16 pb-4 overflow-auto border-r
border-gray-300">
                <a class="flex items-center justify-center flex-shrink-0 w-full h-16
bg-gray-300" href="#">
                    <svg class="w-8 h-8"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 7l-8-4-8 4m16 0l-8 4m8-4v10l-8 4m0-10L4 7m8 4v10M4 7v10l8 4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1
```

```
1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2 2 0
01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0 01-.707.293h-3.172a1 1 0
01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M16 8v8m-4-5v5m-4-2v2m-2 4h12a2 2 0 002-2V6a2 2 0 00-2-2H6a2 2 0 00-2
2v12a2 2 0 002 2z" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 rounded hover:bg-
gray-300" href="#">
                    <svg class="w-5 h-5"  xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2"
d="M12 6V4m0 2a2 2 0 100 4m0-4a2 2 0 110 4m-6 8a2 2 0 100-4m0 4a2 2 0 110-4m0
4v2m0-6V4m6 6v10m6-2a2 2 0 100-4m0 4a2 2 0 110-4m0 4v2m0-6V4" />
                    </svg>
                </a>
                <a
class="flex items-center justify-center flex-shrink-0 w-10 h-10 mt-4 mt-auto rounded
hover:bg-gray-300" href="#">
                    <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2" d="M5.121 17.804A13.937 13.937 0 0112 16c2.5 0 4.847.655 6.879 1.804M15 10a3
3 0 11-6 0 3 3 0 016 0zm6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
                    </svg>
                </a>
            </div>
```

```
            <div class="flex flex-col w-56 border-r border-gray-300">
                <button class="relative text-sm focus:outline-none group">
                    <div class="flex items-center justify-between w-full h-16 px-4
border-b border-gray-300 hover:bg-gray-300">
                        <span class="font-medium">
                            Dropdown
                        </span>
                        <svg class="w-4 h-4" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 20 20" fill="currentColor">
                            <path fill-rule="evenodd" d="M5.293 7.293a1 1 0 011.414
0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0
010-1.414z" clip-rule="evenodd" />
                        </svg>
                    </div>
                    <div class="absolute z-10 flex-col items-start hidden w-full pb-1
bg-white shadow-lg group-focus:flex">
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                        <a class="w-full px-4 py-2 text-left hover:bg-gray-300"
href="#">Menu Item 1</a>
                    </div>
                </button>

                <div class="flex flex-col flex-grow p-4 overflow-auto">
                    <RouterLink
                        to="/admin"
                        class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Dashboard</span>
                    </RouterLink>
                    <RouterLink
                        to="/admin/products"
                        class="flex items-center flex-shrink-0 h-10 px-2 text-sm font-
medium rounded hover:bg-gray-300" href="#">
                        <span class="leading-none">Productos</span>
                    </RouterLink>

                    <RouterLink class="flex items-center flex-shrink-0 h-10 px-3 mt-
auto text-sm font-medium bg-gray-200 rounded hover:bg-gray-300"
                        to="/admin/products/create">
                        <svg class="w-5 h-5" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
                        </svg>
                        <span class="ml-2 leading-none">Nuevo Producto</span>
                    </RouterLink>
                </div>

            </div>
            <div class="flex flex-col flex-grow w-full">
                <div class="flex items-center flex-shrink-0 h-16 px-8 border-b border-
gray-300 justify-between">
                    <h1 class="text-lg font-medium">{{ authStore.username }}</h1>
```

```
                    <div class="flex items-center ml-auto">
                        <button class="flex items-center justify-center h-10 px-4 text-
sm font-medium rounded hover:bg-gray-300">
                            Action 1
                        </button>
                        <button class="flex items-center justify-center h-10 px-4 ml-2
text-sm font-medium bg-gray-200 rounded hover:bg-gray-300">
                            Action 2
                        </button>
                        <button class="relative ml-2 text-sm focus:outline-none group">
                            <div class="flex items-center justify-between w-10 h-10
rounded hover:bg-gray-300">
                                <svg class="w-5 h-5 mx-auto" xmlns="http://www.w3.org/
2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                    <path stroke-linecap="round" stroke-
linejoin="round" stroke-width="2"
d="M12 5v.01M12 12v.01M12 19v.01M12 6a1 1 0 110-2 1 1 0 010 2zm0 7a1 1 0 110-2 1 1 0
010 2zm0 7a1 1 0 110-2 1 1 0 010 2z" />
                                </svg>
                            </div>
                            <div class="absolute right-0 flex-col items-start hidden
w-40 pb-1 bg-white border border-gray-300 shadow-lg group-focus:flex">
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                                <a class="w-full px-4 py-2 text-left hover:bg-
gray-300" href="#">Menu Item 1</a>
                            </div>
                        </button>
                    </div>
                </div>
                <div class="flex-grow p-6 overflow-auto bg-gray-200">
                   <router-view />
                </div>
            </div>


</div>

<a
class="fixed flex items-center justify-center h-8 pr-2 pl-1 bg-blue-600 rounded-full
bottom-0 right-0 mr-4 mb-4 shadow-lg text-blue-100 hover:bg-blue-600" href="https://
twitter.com/lofiui" target="_top">
    <div class="flex items-center justify-center h-6 w-6 bg-blue-500 rounded-full">
        <svg viewBox="0 0 24 24"
class="w-4 h-4 fill-current r-jwli3a r-4qtqp9 r-yyyyoo r-16y2uox r-1q142lx r-8kz0gk r-
dnmrzs r-bnwqim r-1plcrui r-lrvibr r-1srniue"><g><path d="M23.643 4.937c-.
835.37-1.732.62-2.675.733.962-.576 1.7-1.49 2.048-2.578-.9.534-1.897.922-2.958 1.13-.
85-.904-2.06-1.47-3.4-1.47-2.572 0-4.658 2.086-4.658 4.66 0 .364.042.718.12
1.06-3.873-.195-7.304-2.05-9.602-4.868-.4.69-.63 1.49-.63 2.342 0 1.616.823 3.043 2.072
3.878-.764-.025-1.482-.234-2.11-.583v.06c0 2.257 1.605 4.14 3.737 4.568-.392.106-.
803.162-1.227.162-.3 0-.593-.028-.877-.082.593 1.85 2.313 3.198 4.352 3.234-1.595
1.25-3.604 1.995-5.786 1.995-.376 0-.747-.022-1.112-.065 2.062 1.323 4.51 2.093 7.14
2.093 8.57 0 13.255-7.098 13.255-13.254 0-.2-.005-.402-.014-.602.91-.658 1.7-1.477
2.323-2.41z"></path></g></svg>
```

```
    </div>
    <span class="text-sm ml-1 leading-none">@lofiui</span>
</a>
</template>

<script lang="ts" setup>
import { useAuthStore } from '@/modules/auth/stores/auth.store';


    const authStore = useAuthStore()

</script>
```