Portafolio "El ilustronco"

Resumen de proyecto

Este en una pagina web diseñada en Vue.js en el cual representa un portal de trabajos y merchadising diseñados por Rodrigo Pequeño bajo el alias de "El ilustronco", esta página se divine en varias partes:

- Home: Carrusel con información que redirije a las otras secciones.
- Merch: Sección con una lista representativa del merchandising del dibujante.
    - Detalle: Vista detallada del producto seleccionado.
- Galería: Representación de dibujos que se ha realizado y subido a través de instagram por medio de una API.
- Comisiones: Simulación de Formulario para poder realizar un pedido de dibujo directo al dibujante con validadores en cada input. Solo accesible al iniciar sesión.
- Login: Vista en donde se puede iniciar sesión a través de Firebase, y además de un botón que abre una ventana para poder registrarse con una cuenta nueva a través del mismo sistema de autenticación.

Pasos previo instalación de proyecto

Instalar extensión de Chrome/Edge "Access-Control-Allow-origin" en este link Extension

- Agregar extension
- Habilitar extension en el navegador
- Clickear el logo de la extension
- Clickear el logo dentro de la ventana, o en su defecto clickear toggle ON
- Clickear "Test CORS"
- Clickear AJAX::GET, no cerrar la extension durante la prueba de la pagina.

Ejecucion de proyecto

Abrir terminal y escribir los siguientes comandos

```
cd troncofolio
npm install
```

Luego abrir el documento index.js de la carpeta store Index Store y remueve los // del comentario en la linea 42.

```
//'X-RapidAPI-Key': 'f7064d60acmshee8183e4f73c262p1e6ffdjsn67ec8b1c3005',
```

Finalmente escriba este comando en la terminal

```
npm run serve
```

La API usada es de uso limitado, por lo que se sugiere arrancar el proyecto de forma moderada o comentar la linea mencionada anteriormente si no desea navegar por la vista de Galeria

Implementaciones tecnicas

Maqueteado de vista con HTML y CSS

- Utilizacion de tags HTML, Responsividad y Framework CSS En la linea 3 de CommView.js, se logra pareciar el uso de tags como titulos, forms, etiquetas y otros, responsividad establecida por Bootstrap (y también por media query en otros archivos)

```
<form class="commform row g-3">
        <h2>Plataforma de Comisiones</h2>
        <div class="row mb-3">
            <label for="inputEmail3" class="col-sm-2 col-form-
label">Correo</label>
            <div class="col-sm-10">
                <input type="email" class="form-control" id="inputEmail3"
:value="correo" aria-label="Disabled input example" disabled>
            </div>
        </div>
```

línea 227 del mismo archivo, diseño del contenedor del formulario de pedido de dibujos.

```
.commform {
    font-size: large;
    margin: auto;
    margin-top: 15px;
    width: 95vw;
    height: 85vh;
    background: linear-gradient(to right, #2bceffa8, #41ff80b7);
    padding: 20px;
    overflow-y: scroll;
    border-radius: 15px;
    box-shadow: 0.3rem 0.4rem 0.4rem rgba(0, 0, 0, 0.4);
    border: 1px solid lightgray;
}
```

Uso de componente Vue

- Código mantenible mediante componentes, el cuál es demostrado en App.vue donde se aprecia el llamado de diferentes componentes.

```
<template>
<div class="page-container">
  <div class="wrapper">
```

```
    <NavBar/>
    <router-view v-slot="{ Component }">
  <transition name="slide-fade">
    <component :is="Component" />
  </transition>
</router-view>
  </div>
  <FooterBar/>
</div>

</template>

<script>
import NavBar from "./components/NavBar.vue"
import { mapActions } from "vuex";
import FooterBar from "./components/FooterBar.vue"

export default{

  components: {
    NavBar,
    FooterBar,
  },
  computed:{
    ...mapActions(['feedInsta']),
  },
  async created(){
    await this.feedInsta
  }
}
</script>
```

- **Comunicación entre componentes**, esta característica se aprecia con la interacción entre el store y el componente ArtView en la línea 22

```
<script>
import { mapState } from 'vuex';


export default {
    computed: {
        ...mapState(['imagenes'])
    }
}
</script>
```

- **Ciclos de vida de un componente**, un ejemplo es en el componente ListaMerch.vue donde primero se obtiene el merchandising através del ciclo created para luego ser visualizado e interactuado antes de las otras funciones

```
<script>
import { ProductService } from '@/services/ProductService';

export default{
    data: function(){
        return {
            listaP:[],
        }
    },
    created: async function(){
        try{
            let respuesta = await ProductService.getAllMerch();
            this.listaP = respuesta;
        } catch (error){
            this.errormessage = error
        }
    }
}
</script>
```

- **Definición de rutas**
    - *Rutas URI*: Las rutas URI definidas en el proyecto (Véase en index.js Router se han ordenado según al contenido que aguardan y al orden lógico de navegación de la página:

```
  const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/merch',
    name: 'merch',
    component:MerchView,
  },
  {
    path: '/merch/:id',
    name: 'detalle-merch',
    component: DetalleProducto
  },
  {
    path: '/login',
    name: 'login',
    component:LoginView,
  },
  {
    path: '/comm',
    name: 'comm',
    component: CommView,
  },
```

```
  {
    path: '/gallery',
    name: 'gallery',
    component: ArtView,
  },
  // redireccion automatica a 404 cuando se ingresa una ruta no existe en primera
rama
  {
    path: '/:pathMatch(.*)*',
    name: 'not-found',
    component: NotFound
  },]
```

- Parámetros por URL, los cuales se encuentran en el ejemplo anterior

```
{
    path: '/producto',
    redirect: '/merch',
    children: [
      {
        path: '/producto/:id',
        name: 'detalle-merch',
        component: DetalleProducto,
      }],
  },
```

- Rutas por defecto el cual encontrarás en el primer path de la lista routes:

```
const routes = [
  {
    path: '/merch',
    name: 'merch',
    component:MerchView,
  },
  {
    path: '/merch/:id',
    name: 'detalle-merch',
    component: DetalleProducto
  },
```

Programacion con Javascript

Usos básicos y de resolución de problemas a través de ES6/7 lo pueden encontrar en códigos tales como a partir de la línea 14 en DetalleProducto.vue

```
<script>
import router from '@/router';
```

```
import { ProductService } from '@/services/ProductService';


export default {
  data() {
    return {
      merch: null,
      checkearLista: [],
    };
  },
  created() {
    const id = this.$route.params.id;
    this.loadMerch(id);
  },
  watch: {
    '$route.params.id': function(newId) {
      this.loadMerch(newId);
    }
  },
  methods: {
    async loadMerch(id) {
      try {
        this.merch = await ProductService.getMerchById(id);
        if(this.merch === null) {
            router.push({name: "not-found"})
        }
      } catch (error) {
        console.log(error);
      }
    }
  }
  }

</script>
```

Consumo, manejo de datos y estados

- Consumo de servicios de datos, hay varios componentes que consumen datos, uno de ellos es el componete de login/registro, un extracto encontrado en la linea 58

```
<script>
...
export default{
    name: 'LoginFire',
    data: function(){
        return {
            formulario:{
                correo:"",
                password:""
            },
            nuevoTronco:{
```

```
                correoN:"",
                passwordN:""
            }
        };
    },
 ...
</script>
```

- Manejo de estados internos de los componentes web, esto es demostrado en index.js del Store de la aplicacion, el cual la funcion desconectar() modifica el estado conexion y correo simultaneamente

```
////
export default createStore({
  state: {
    conexion: false,
    imagenes: null,
    correo: "",
    /////
desconectar(state) {
      if (state.conexion == true) {
        signOut(auth).then(() => {
          state.conexion = false;
          state.correo = "";
          router.push('/')
        }).catch((error) => {
          alert(error);
        })
      } else {
        alert("Ya estas desconectado")
      }
    },
})
```

- Manejo de estado de la aplicacion con Vuex, en el componente NavBar.vue, llama al estado *conexion* y *correo* por medio de Vuex, ademas se llama a la mutacion *desconectar* para arrancar la funcion de desloggeo del usuario

```
<script>
import {mapState,mapMutations} from "vuex"

export default{
    computed:{
        ...mapState(['conexion']),
        ...mapState(['correo'])
    },
    methods: {
        ...mapMutations(['desconectar'])
    },
}
```

```
</script>
```

Customize configuration

See Configuration Reference.