PORTAFOLIO "ACADEMIA DE DANZA ARABE"

Resumen de proyecto

Esta es un página web diseñada con el framework vue.js el cual representa la página web de una academia de danza árabe que ofrece inscribirse a los diferentes cursos que se imparten, esta página se divide en las siguientes partes:

- Home: esta es la página principal de inicio en donde tenemos un menú de navegación con las diferentes secciones que se encuentran en la página web, y hay un video con musica que muestra el baile entretenido fusión oriental.

- Quienes somos: esta es la sección de la página web donde se muestra sobre la academia.

- Cursos: en esta sección de la página web, tenemos un carrusel con las imágenes de los diferentes cursos que se imparten y que te puedes inscribir haciendo clic en el botón de inscribirse, en donde me aparece un modal del formulario de inscripción con sus campos validados.La información de los cursos se trae desde un json(una simulación de consumo de api).

- Galería: esta es la sección de la página web donde se muestran imágenes relaciones de la danza árabe, estas imágenes se están trayendo desde una api, y estas imágenes puedo hacer clic en ellas y verlas en un modal con su nombre y descripción que vienen desde la api que se está utilizando.

- Contacto: esta es la sección de la página web donde se muestra un formulario de contacto con sus validaciones.

Características:

- Maquetación con HTML y CSS.
- Uso de componentes Vue.
- Programación con javascript ES6.
- Consumo, manejo de datos y estados.
- Estructura de archivos y carpetas.

Ejecución del proyecto

Abrir el terminal y escribir los siguientes comandos:

- cd portafolio_fran

- npm install

y finalmente compilar el siguiente comando para la visualización del proyecto

- npm run serve

Implementaciones técnicas

1. Maquetación con HTML Y CSS: uso de bootstrap y etiquetas.

Ejemplo de HTML y Bootstrap en src/views/CursosView.vue

```
<template>
  <div>
    <div v-if="cargando" id="carga">
      <pulse-loader :loading="loading" :color="color" :size="size"></pulse-loader>
    </div>

    <div class="container-fluid p-0" v-else>
      <div id="carrusel" class="carousel slide" data-bs-ride="carousel">
        <ol class="carousel-indicators">
          <li data-bs-target="#carrusel" v-for="(curso, index) in cursos" :key="curso.id" :data-bs-slide-to="index" :class="{ active: index === 0 }"></li>
        </ol>
        <div class="carousel-inner">
          <div v-for="(curso, index) in cursos" :key="index" class="carousel-item" :class="{ active: index === 0 }">
            <img :src="curso.imagen" class="d-block w-100" alt="imagen del curso">
            <div class="carousel-caption">
              <h2>{{ curso.nombre }}</h2>
              <p>Días: {{ curso.dias }}</p>
              <p>Horarios: {{ curso.horarios }}</p>
              <p>Cupos: {{ curso.cupos }}</p>
              <p>{{ curso.descripcion }}</p>
              <p class="mb-3"> Valor:  {{new Intl.NumberFormat('es-CL', {currency: 'CLP', style: 'currency' }).format(curso.valor)}}</p>

              <!-- Botón para abrir el modal -->
              <button class="btn" data-bs-toggle="modal" data-bs-target="#formularioModal">Inscribirse</button>

            </div>
          </div>
        </div>
        <a class="carousel-control-prev" href="#carrusel" role="button" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Anterior</span>
        </a>
        <a class="carousel-control-next" href="#carrusel" role="button" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Siguiente</span>
        </a>
      </div>
    </div>
  </div>
```

Se muestra un carrusel de imágenes de los cursos que se imparten, con su nombre, días en los que se realizan, los horarios, cupos y su descripción más un botón de inscribirse que abre un formulario modal para su inscripción al curso.

Css y Responsividad

```
<style scoped>
#carrusel img {
  width: 100%;
  height: 500px;
  object-fit: cover;
}
1 reference
.carousel-caption {
  background-color: rgba(4, 3, 3, 0.5);
  padding: 10px;
  color: #e7dcdc;
  margin-bottom: 10px;
  font-family: 'Comfortaa', cursive;
}
1 reference
.carousel-indicators li {
  background-color: #fff;
  border: none;
  height: 10px;
  width: 10px;
  margin: 0 5px
}
```

```
3 references
.btn {
  background-color: initial;
  background-image: linear-gradient( #8614f8 0,  #760be0 100%);
  border-radius: 5px;
  border-style: none;
  box-shadow:  rgba(245, 244, 247, .25) 0 1px 1px inset;
  color:  #fff;
  cursor: pointer;
  display: inline-block;
  font-family: 'Comfortaa', cursive;
  font-size: 16px;
  font-weight: 500;
  height: 50px;
  line-height: 50px;
  margin-left: -4px;
  outline: 0;
  text-align: center;
  transition: all .3s cubic-bezier(.05, .03, .35, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  vertical-align: bottom;
  width: 190px;
}
```

```
3 references
.btn:hover {
  opacity: .7;
}
@media screen and (max-width: 1000px) {
  3 references
  .btn {
    font-size: 14px;
    height: 55px;
    line-height: 55px;
    width: 150px;
  }
}
1 reference
.modal {
  font-family: 'Comfortaa', cursive;
  color:  #2a2a2a;
}
</style>
```

2. **Uso de componentes Vue**

Uso de componentes en src/App.vue

```vue
<template>
  <NavBar />
  <router-view />
  <FooterBar />
</template>
<script>
  import NavBar from "@/components/NavBar.vue"
  import FooterBar from "@/components/FooterBar.vue"
  export default (await import('vue')).defineComponent({
  name: 'App',
  components: {
    NavBar,
    FooterBar
    }
})
</script>
```

Componentes:

- **NavBar:** Este componente es la barra de navegación del sitio web.
- **router-view:** Este componente es proporcionado por un enrutador y se utiliza para renderizar las vistas correspondientes según la ruta actual.
- **FooterBar:** Este componente es el pie de página del sitio.

Ejemplo de uso de ciclos de vida de un componente:

```
computed: {
  ...mapState(["cursos"])
},
created: async function() {
  try {
    this.cargando = true;
    this.cursos = await Cursos.getAllCursos(); // Usa await para esperar la carga de los cursos
    this.cargando = false;
  } catch (error) {
    console.error(error);
  }
},
```

Ejemplo de utilización de buenas prácticas en la definición de rutas:

```
const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/quienessomos',
    name: 'quienessomos',
    component: QuienesSomos
  },
  {
    path: '/cursos',
    name: 'cursos',
    component: CursosView
  },
```

```
  {
  path: '/galeria',
  name: 'galeria',
  component: GaleriaView,
  children: [
    {
      path: ':id',
      name: 'detalleimagen',
      component: DetalleImagen,
      props: true
    }
  ]
},
  {
    path: '/contacto',
    name: 'contacto',
    component: ContactoView
  },
  {
    path: '/:pathMatch(.*)*',
    name: 'NotFound',
    component: NotFound
    }
]
```

Rutas del proyecto:

La primera ruta es para la vista de home ("/") y en donde tenemos la vista principal de inicio de nuestra página con un video con música en el fondo y se asocia al componente HomeView.

La segunda ruta es para la vista de quienes somos ("/quienessomos") y en donde visualizamos sobre la academia de danza árabe y se asocia al componente QuienesSomos.

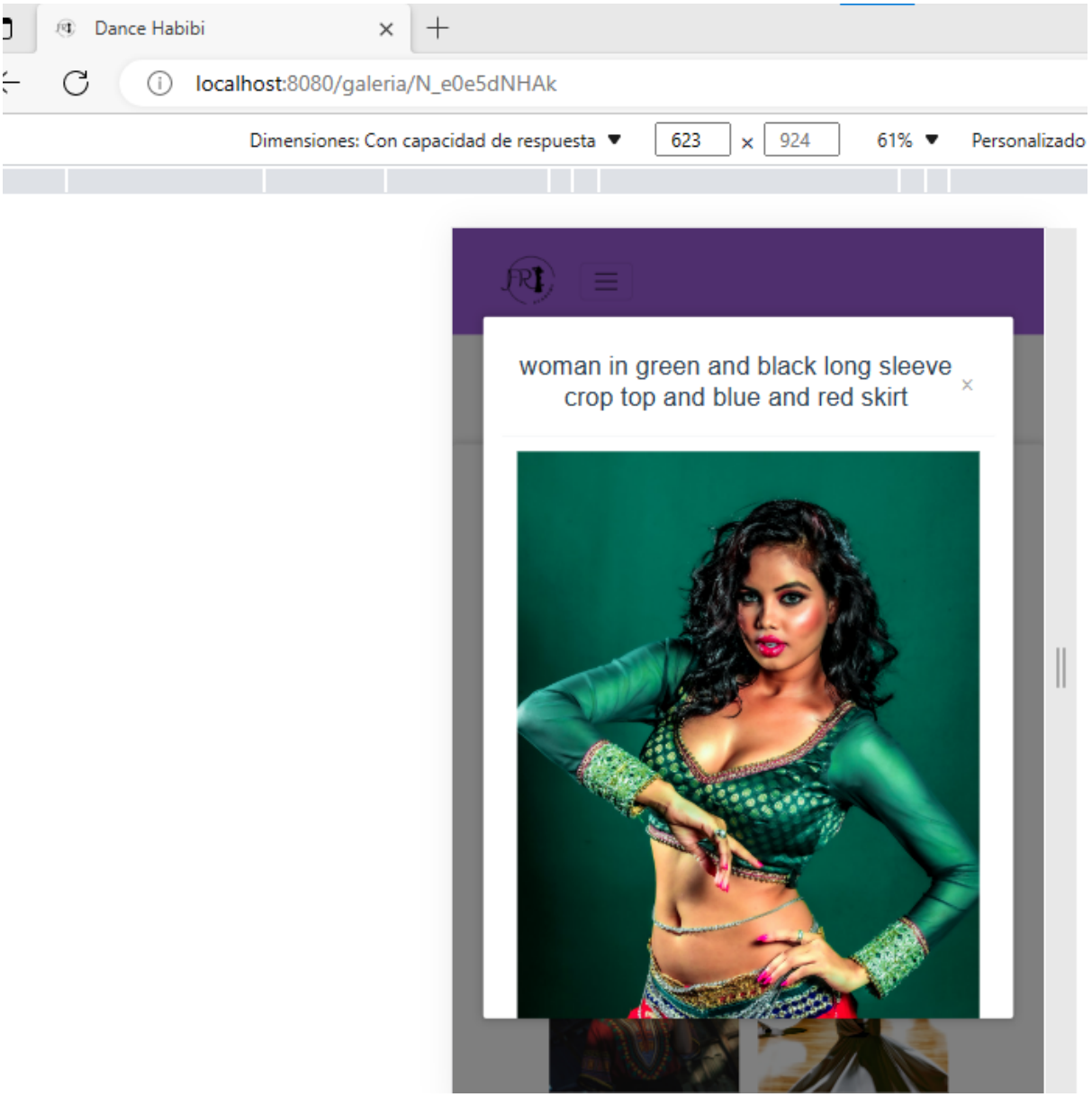
La tercera ruta es para la vista de cursos ("/cursos") y en donde podemos ver los cursos que se ofrecen en la academia y se asocia al componente CursosView.

La cuarta ruta es para la vista de galería ("/galeria") en donde en esta vista visualizamos imágenes de danza arabe cargadas desde una api, las cuales se pueden pinchar la imagen y visualizar desde un modal la imagen con su nombre y descripción que se trae la api ocupando ahí una ruta secundaria; se asocia con el componente GaleriaView y con el componente DetalleImagen que es el cual nos muestra el modal.

La quinta ruta es para la vista de contacto ("/contacto") en donde podemos visualizar un formulario de contacto y se asocia al componente ContactoView.

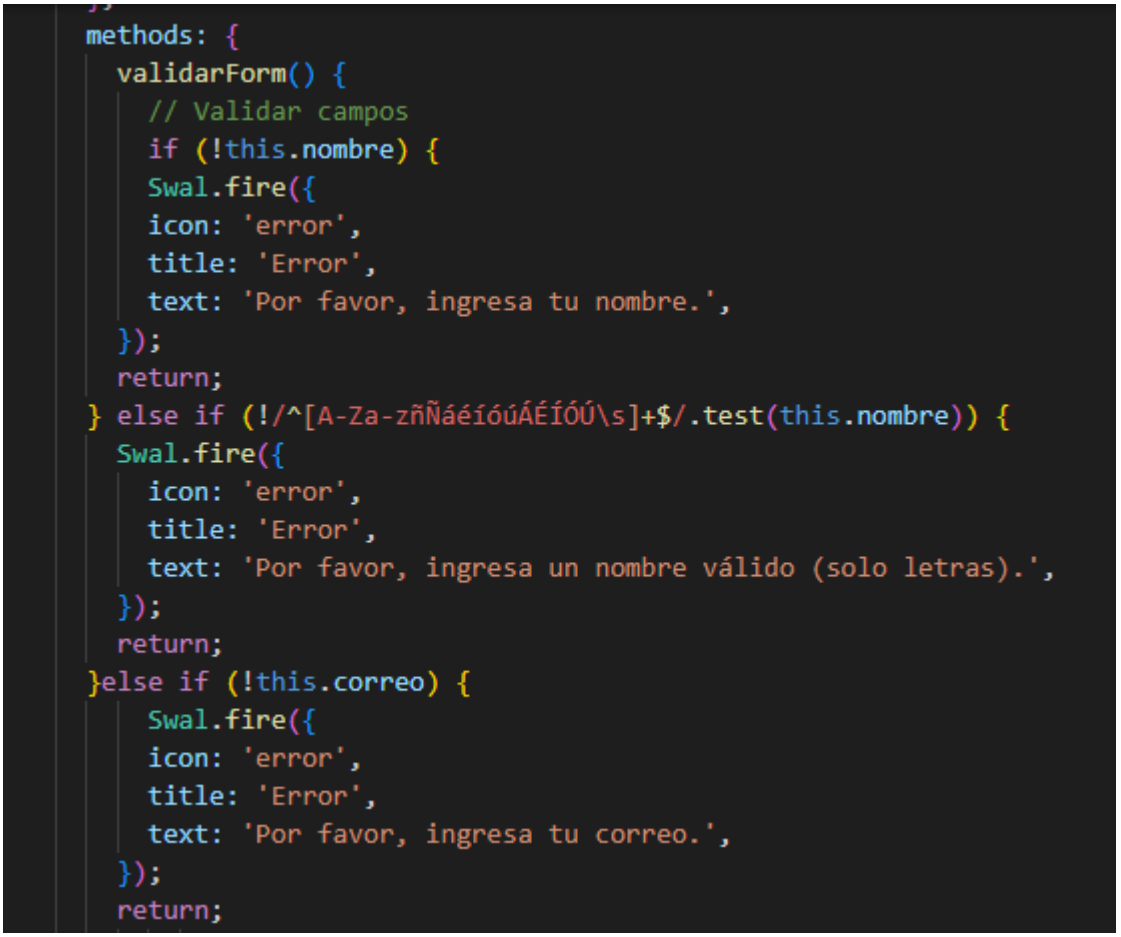
La última ruta es una ruta de comodín (/:pathMatch(.*)* que se utiliza para manejar las rutas no encontradas y controlar el error404 y se asocia al componente NotFound.

Ejemplo de URL por parámetro para cada imagen que se haga clic en la galería:



3. Programación con JavaScript ES6

Aca muestro un ejemplo de uso de javascript es6 en validación de los campos del formulario de contacto de la vista ContactoView



```
methods: {
  validarForm() {
    // Validar campos
    if (!this.nombre) {
    Swal.fire({
    icon: 'error',
    title: 'Error',
    text: 'Por favor, ingresa tu nombre.',
  });
  return;
} else if (!/^[A-Za-zñÑáéíóúÁÉÍÓÚ\s]+$/.test(this.nombre)) {
  Swal.fire({
    icon: 'error',
    title: 'Error',
    text: 'Por favor, ingresa un nombre válido (solo letras).',
  });
  return;
}else if (!this.correo) {
    Swal.fire({
    icon: 'error',
    title: 'Error',
    text: 'Por favor, ingresa tu correo.',
  });
  return;
```

```
        // Validar formato de correo
      } else if ((!/^[\w-.]+@([\w-]+\.)+[\w-]{2,4}$/.test(this.correo))) {
        // Swal("Error", "El correo electrónico no es válido.", "error");
      Swal.fire({
      icon: 'error',
      title: 'Error',
      text: 'Por favor, ingrese un correo electrónico válido.',
    });
        return;

      }else if (!this.mensaje) {
        // Swal("Error", "Por favor, completa todos los campos.", "error");
      Swal.fire({
      icon: 'error',
      title: 'Error',
      text: 'Por favor, ingresa tu comentario.',
    });
        return;
    }
    else {
      Swal.fire({
      icon: 'success',
      title: 'Éxito',
      text: '¡Tus datos fueron enviados correctamente!',
        });
      }
    },
  }
```

4. Consumo de datos

Consumo de 2 APIs para obtener imágenes de danza arabe en la galería , se agrega a un arreglo para después ser mostrado en el sitio, y se utiliza una query para que solo me busque imágenes de danza arabe en la api.

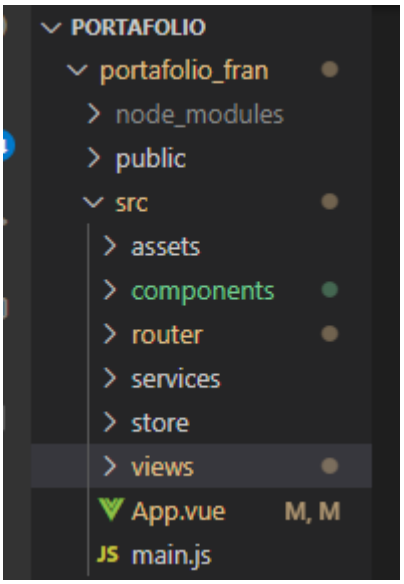
```
methods: {
  async buscarImagenesDanzaArabe() {
    try {
      const ACCESS_KEY = 'n4g1M0ad4Z-z6vg_WNQNDmDkKbQmhzNFjHk-Xeuf2wQ';

      const response = await axios.get('https://api.unsplash.com/search/photos', {
        headers: {
          'Authorization': `Client-ID ${ACCESS_KEY}`
        },
        params: {
          query: 'arabic dance',
        }
      });

      this.imagenes = response.data.results;
    } catch (error) {
      console.error('Error al obtener las imágenes:', error);
    }
  },
```

Se muestra el uso de async/await para realizar solicitudes asincrónicas y esperar las respuestas antes de continuar con la ejecución del código. Esto garantiza que los datos estén disponibles antes de utilizarlos en la lógica posterior.

5. Estructura de archivos y carpetas



Link del proyecto https://github.com/Frany02/portafolio_fran