Taller Vue.Js con TypeScript

Objetivo

El objetivo de este taller es hacer una pequeña introducción al framework de Vue.Js con typeScript, utilizando Single file component así como algunas directivas básicas, dando a conocer la herramienta de manera que los compañeros puedan tener un primer acercamiento, y en futuros proyectos puedan optar por realizar aplicaciones con dicho framework.

Problema a resolver

Se presenta una aplicación sencilla para administrar tareas, así como un apartado con la descripción de la misma. En está aplicación el usuario puede agregar nuevas tareas, editarlas, o eliminarlas, así como marcar una tarea como realizada. Además de ver en un listado aparte todas la tareas, las tareas pendientes y las tareas que marco como completadas.

Por último, pueden tener la opción de marcar todas las opciones como completadas y eliminarlas, en conjunto.

A continuación, se presenta una imagen con el resultado esperado.

Acerca de este proyecto

Por hacer

Requisitos previos

Para poder realizar el taller se debe instalar en el sistema nodeJs y la interfaz de línea de comandos de Vue (CLI).

Importante

En sección de *"Edición de proyecto"*, al inicio de cada instrucción se indica el archivo que debe modificarse.

A partir del paso 13, se indica en que parte de la estructura de cada archivo, debe adjuntarse el código brindado, siendo en el template (HTML), script (puede adjuntarse en data, methods o computed) o style (CSS).

Creación del proyecto

1. Abrir una terminal en la carpeta donde desea crear el proyecto e ingresamos el siguiente comando:

```
vue create taller-vue
```

2. Seleccionar la opción **Manually select features**
3. Marcar únicamente las opciones **Babel**, **TypeScript** y **Router**(las demás opciones deben estar desmarcadas).
4. Indicar que **NO** vamos a usar la sintaxis de clase para el componente.
5. Indicar que **SI** vamos a utilizar Babel junto a TypeScript.
6. Indicar que **NO** vamos a utilizar el modo histórico de router.
7. Seleccionar la opción **In package.json**.

8. Indicar que **NO** vamos a guardar la configuración como preset.

Al final, deberá quedar la configuración de la siguiente manera:

```
Vue CLI v4.1.2
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, TS, Router
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfi
lls, transpiling JSX)? Yes
? Use history mode for router? (Requires proper server setup for index fallback
in production) No
? Where do you prefer placing config for Babel, ESLint, etc.? In package.json
? Save this as a preset for future projects? No
```

Ejecutar el proyecto creado

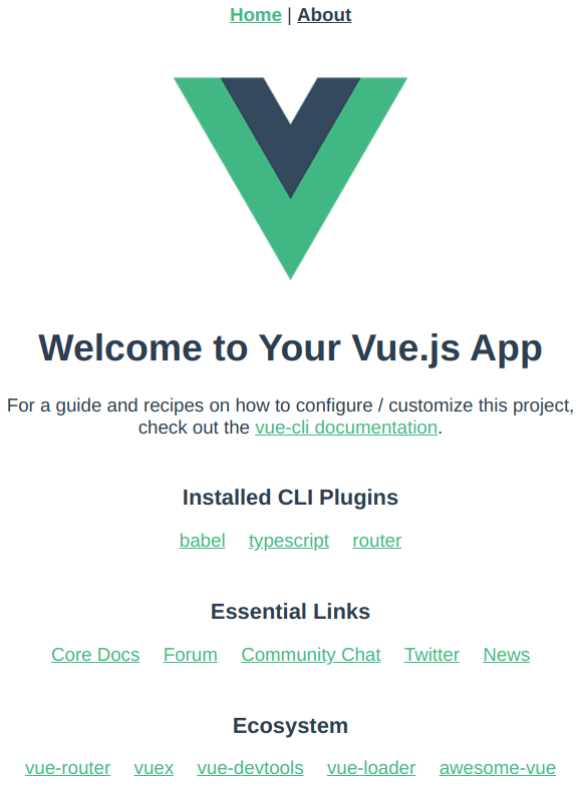
Para poder ejecutar el proyecto que acabamos de crear, nos movemos a la carpeta del proyecto.

```
cd taller-vue
```

Y una vez en el directorio, mediante el siguiente comando se ejecuta el proyecto:

```
npm run serve
```

El proyecto creado tiene la siguiente interfaz:



Edición del proyecto

1. **(public/index.html)** Agregar bootstrap y fontAwesome al proyecto en public/index.html para hacer uso de los componentes de Boostrap y los iconos de FontAwesome.
    - Boostrap :

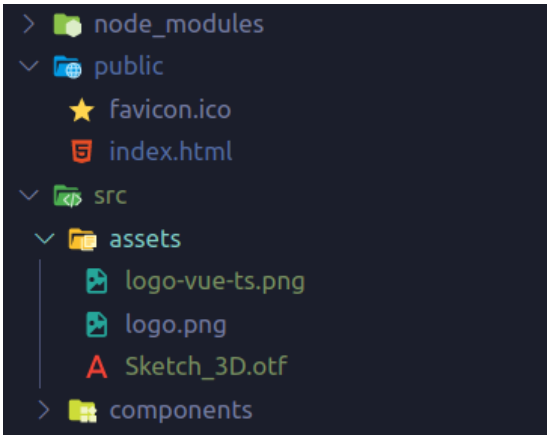
```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.
min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
```

```
 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper
.min.js"
    integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
    crossorigin="anonymous"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.mi
n.js"
    integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
    crossorigin="anonymous"></script>
```

- FontAwesome:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.11.2/css/all.css">
```

2. **(src/assets/)**Añadir en la carpeta Assets los archivos que se adjuntaron con el envío de este instructivo.



3. **(src/App.vue)**Ir al archivo src/App.vue y eliminar el contenido de las etiquetas template y script.

```
<template>

</template>

<style>

</style>
```

4. **(src/App.vue)** Cree un div dentro de la etiqueta template que contendrá todo el código HTML, de este archivo. Esto es requerido ya que se debe tener todo el código HTML en un solo contenedor.

5. **(src/App.vue)** Dentro de la etiqueta div, ingrese el código para crear una barra de navegación, los aspectos importantes de este código son las etiquetas router-link que son utilizadas para especificar la vista que será renderizada en la etiqueta router-view (agregada más adelante).

```
<nav class="navbar navbar-expand-lg navbar-dark bg-info">
      <a class="navbar-brand" href="#">
        <img src="@/assets/logo-vue-ts.png" width="40" height="40" alt />
      </a>
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbarNav"
        aria-controls="navbarNav"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item" :class="{active: pagina == 'acercaDe'}"
@click="pagina='acercaDe'">
            <router-link class="nav-link" to="/">Acerca de este
proyecto</router-link>
          </li>
          <li class="nav-item" :class="{active: pagina == 'porHacer'}"
@click="pagina='porHacer'">
            <router-link class="nav-link" to="/porHacer">Por hacer</router-
link>
          </li>
        </ul>
      </div>
    </nav>
```

6. **(src/App.vue)** Agregar después de la barra de navegación un etiqueta div, que contendrá la etiqueta router-view, encargada de renderizar las vistas especificadas en las rutas (archivo src/router/index.ts modificado más adelante).

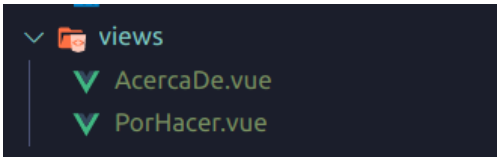
   Este será todo el HTML requerido para este archivo.

```
<div class="container">
     <router-view />
</div>
```

7. **(src/App.vue)** Para finalizar con este archivo agregamos una etiqueta script, especificando que se va a trabajar con typeScript, mediante el atributo lang con un valor de "ts", como se muestra en la imagen. Este contiene una variable página que es utilizada para mostrar el enlace de navegación actual en la barra de navegación.

```
<script lang="ts">
export default {
  data() {
    return {
      pagina: "acercaDe"
    };
  }
};
</script>
```

8. **(src/views)** Crear en la carpeta src/views los archivos que serán los componentes que se mostrarán en la etiqueta router-view del archivo App.vue. Eliminar los archivos Home.vue y About.vue. Y agregar los mostrados en la imagen. Estos serán importados en el archivo src/router/index.ts en el siguiente paso.



9. **(src/router/index.ts)** Eliminar el "import Home from '../views/Home.vue'" y colocar los imports para los archivos del paso 8.

```
import PorHacer from '../views/PorHacer.vue'
import AcercaDe from '../views/AcercaDe.vue'
```

10. **(src/router/index.ts)** Reemplazar las rutas especificadas en la variable routes con las que utilizaremos para esta aplicación.

```
const routes = [
  {
    path: '/',
    name: 'acerca-de',
    component: AcercaDe
  },
  {
    path: '/porHacer',
    name: 'por-hacer',
    component: PorHacer
  }
]
```

Este momento en el navegador se debería poder observar simplemente la barra de navegación ya que los archivos PorHacer.vue y AcercaDe.vue están vacíos.

11. **(src/views/AcercaDe.vue)** Esta vista tendrá una sección informativa acerca de lo que realizado en este taller. Para ver la estructura de estos archivos ir a la sección de anexos.

    Crear una etiqueta template para agregar el html para esta vista, además en una etiqueta style agregar los estilos.

```
<template>
  <div class="contenedor">
    <h1>Taller de Vue.Js con TypeScript</h1>
    <p>Este taller fue realizado para proponer un ejercicio para ser
desarrollado en clase, de manera que se muestren las funcionalidades que
propone el framework VueJs utilizando TypeScript como lenguaje de
programación. La aplicación que se desarrolla en este taller consiste en un
sistema para administrar tareas o actividades, donde se incluyen todas las
operaciones de leer, crear, modificar y eliminar para los ítemes de tareas.
</p>
  </div>
</template>

<style scoped>
@font-face {
  font-family: sketch;
  src: url(../assets/Sketch_3D.otf);
}

.contenedor {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 60vh;
  padding: 0 100px;
}

h1 {
  font-family: sketch;
  margin-bottom: 50px;
  font-size: 70px;
  text-align: center;
}
</style>
```

Revisar que en el navegador en la sección "acerca de este proyecto" de la aplicación se debería mostrar lo siguiente:

12. **(src/views/PorHacer.vue)** Este archivo tiene la definición del contenido principal de la aplicación. Colocar la siguente estructura base para el archivo PorHacer.vue, se agregaron también las variables utilizadas para la lógica de la aplicación. El código de estilos se irá agregando con forme se añada código HTML.

```
<template>

</template>

<script lang="ts">
    import Vue from "vue";
    export default Vue.extend({
        name: "todo-list",
        data() {
            return{
                newTodo: "",
                idForTodo: 3,
                beforeEditCache: "",
                filter: "all",
                chk: false,
                todos: [
                    {
                        id: 1,
                        title: "Terminar el laboratorio 2",
                        completed: false,
                        editing: false
                    },
                    {
                        id: 2,
                        title: "Ponerle con el proyecto final! :O",
                        completed: false,
                        editing: false
                    }
                ] as any
            };
        },
        directives: {
            focus: {
              inserted: function(el) {
                el.focus();
              }
```

```
            }
        },
        methods:{

        },
        computed:{

        }
  });
</script>

<style scoped>

</style>
```

13. **(src/views/PorHacer.vue)** En template agregar un div con la clase app que será el contenedor padre de todo el HTML de este componente. Además se agrega la estructura base para el diseño de la aplicación como se muestra a continuación:

    HTML:

```
<div class="app">
    <h1 class="titulo">Tareas por hacer</h1>
    <div class="contenedor-principal">

    </div>
</div>
```

    CSS

```
@import
url("https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.2/animate.min.cs
s");
@font-face {
  font-family: sketch;
  src: url(../assets/Sketch_3D.otf);
}
.app {
  margin-top: 60px;
}

.titulo {
  text-align: center;
  margin-bottom: 50px;
  font-family: sketch;
  font-size: 70px;
}

.contenedor-principal {
  width: 80%;
  margin: 0 auto;
}
```

14. **(src/views/PorHacer.vue)** Agregar el código para el input que se utilizará para agregar tareas nuevas

HTML: agregar dentro del div que tiene la clase contenedor-principal

```
<input
       v-model="newTodo"
       type="text"
       class="todo-input"
       placeholder="Nueva tarea..."
       @keyup.enter="addTodo"/>
```

CSS

```
.todo-input {
  width: 100%;
  padding: 10px 18px;
  font-size: 18px;
  margin-bottom: 16px;
  border: 1px solid #aaa;
  border-radius: 8px;
}

.todo-input:focus {
  outline: 0;
  border: 1px solid #17a2b8;
}
```

MÉTODOS: dentro de la propiedad methods del objeto en la etiqueta script

```
addTodo() {
      if (this.newTodo.trim().length == 0) {
        return;
      }
      this.todos.push({
        id: this.idForTodo,
        title: this.newTodo,
        completed: false
      });
      this.newTodo = "";
      this.idForTodo++;
    },
```

15. Agregar la sección para mostrar la lista de tareas, esto debajo del input anterior

HTML

```
      <transition-group
        enter-active-class="animated fadeInUp"
        leave-active-class="animated fadeOutDown"
      >
        <div v-for="(todo, index) in todosFiltered" :key="todo.id"
class="todo-item">
          <div class="todo-item-left">
            <input type="checkbox" v-model="todo.completed" />
            <div
              v-if="!todo.editing"
              @dblclick="editTodo(todo)"
              class="todo-item-label"
```

```
              :class="{completed : todo.completed}"
            >{{todo.title}}</div>
            <input
              v-else
              class="todo-item-edit"
              type="text"
              v-model="todo.title"
              @blur="doneEdit(todo)"
              @keyup.enter="doneEdit(todo)"
              v-focus
              @keyup.esc="cancelEdit(todo)"
            />
          </div>
          <div class="remove-item" @click="removeTodo(index)">
            <i class="fas fa-trash-alt"></i>
          </div>
        </div>
      </transition-group>
```

CSS

```
.todo-item {
  margin-bottom: 12px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  animation-duration: 0.3s;
  padding-right: 20px;
  padding-left: 10px;
  border-radius: 8px;
  background-color: #e9e9e9;
}

.todo-item:hover {
  -webkit-box-shadow: 6px 6px 18px -8px rgba(0, 0, 0, 0.75);
  -moz-box-shadow: 6px 6px 18px -8px rgba(0, 0, 0, 0.75);
  box-shadow: 6px 6px 18px -8px rgba(0, 0, 0, 0.75);
}

.remove-item {
  cursor: pointer;
  margin-left: 14px;
  color: #dc3545;
}

.remove-item:hover {
  color: #b52130;
}

.todo-item-left {
  display: flex;
  align-items: center;
  width: 100%;
}
```

```
.todo-item-label {
  padding: 10px;
  margin-left: 12px;
}

.todo-item-edit {
  font-size: 24px;
  color: #2c3e50;
  margin-left: 12px;
  width: 100%;
  padding: 10px;
  border: 1px solid #e9e9e9;
  background-color: #e9e9e9;
  font-family: "Avenir", Helvetica, Arial, sans-serif;
}

.todo-item-edit:focus {
  outline: none;
}

.label-text {
  margin-left: 20px;
}

.label-text:hover {
  color: #168530;
  font-weight: bold;
}

label {
  padding-left: 10px;
}

.completed {
  text-decoration: line-through;
  color: grey;
}
```

MÉTODOS

```
removeTodo(index: number) {
      this.todos.splice(index, 1);
    },
    editTodo(todo: any) {
      this.beforeEditCache = todo.title;
      todo.editing = true;
    },
    doneEdit(todo: any) {
      todo.editing = false;
      if (todo.title.trim().length == 0) {
        todo.title = this.beforeEditCache;
      }
    },
    cancelEdit(todo: any) {
      todo.title = this.beforeEditCache;
```

```
      todo.editing = false;
    },
```

COMPUTED

```
todosFiltered() {
      var to: any = this.todos;
      if (this.filter == "all") {
        return to;
      } else if (this.filter == "active") {
        return to.filter((todo: any) => !todo.completed);
      } else if (this.filter == "completed") {
        return to.filter((todo: any) => todo.completed);
      }
      return to;
    },
```

16. Agregar la opción de marcar todos como terminados, hacerlo debajo de la etiqueta de cierre de transition-group.

HTML

```
<div class="extra-container">
        <div>
          <label>
            <input v-model="chk" type="checkbox" :checked="!anyRemaining"
@change="checkAllTodos" />
            <span class="label-text">Marcar todos como terminados</span>
          </label>
        </div>
        <div>{{ remaining }} pendientes</div>
      </div>
```

CSS

```
.extra-container {
  display: flex;
  align-items: center;
  justify-content: space-between;
  font-size: 16px;
  border-top: 1px solid lightgrey;
  padding-top: 14px;
  margin-bottom: 14px;
}
```

MÉTODOS

```
checkAllTodos() {
      var to: any = this.todos;
      var chk_value = this.chk;
      to.forEach((todo: any) => (todo.completed = chk_value));
    },
```

COMPUTED

```
remaining() {
      var to: any = this.todos;
      return to.filter((todo: any) => !todo.completed).length;
    },
    anyRemaining() {
      var rem: any = this.remaining;
      if (rem == 0) {
        this.chk = true;
      } else {
        this.chk = false;
      }
      return rem != 0;
    },
```

17. Agregar los botones para el filtrado de tareas, hacerlo debajo del div agregado en el paso 16

HTML

```
<div class="extra-container">
        <div>
          <button :class="{active: filter == 'all'}" @click="filter =
'all'">Todos</button>
          <button :class="{active: filter == 'active'}" @click="filter =
'active'">Pendientes</button>
          <button :class="{active: filter == 'completed'}" @click="filter =
'completed'">Terminados</button>
        </div>
        <div>
          <transition name="fade">
            <button v-if="showClearCompletedButton"
@click="clearCompleted">Clear completed</button>
          </transition>
        </div>
      </div>
```

CSS

```
button {
  font-size: 14px;
  background-color: white;
  border: 1px solid #aaa;
  padding: 0.5em;
  margin-right: 5px;
  border-radius: 8px;
}

button:hover {
  background: #17a2b8;
  color: #f2f2f2;
  border-color: #17a2b8;
}

button:focus {
```

```
  outline: none;
}

.active {
  border-color: #17a2b8;
  border-width: 2px;
}

.fade-enter-active,
.fade-leave-active {
  transition: opacity 0.2s;
}

.fade-enter,
.fade-leave-to {
  opacity: 0;
}
```

METODOS

```
clearCompleted() {
      this.todos = this.todos.filter((todo: any) => !todo.completed);
    },
```

COMPUTED

```
showClearCompletedButton() {
      var to: any = this.todos;
      return to.filter((todo: any) => todo.completed).length > 0;
    },
```

Anexos

Archivos .vue (Single file component)

Estructuración de un proyecto Vue, utilizando Single file component.

Single File Components – Vue.js