Documentación Asambleas – Giramaster – Frontend

En la presente documentación se encuentra la explicación de cada funcionalidad del proyecto presentado en la carpeta *ASAMBLEAS-GIRAMASTER-MAIN,* con el objetivo de identificar el funcionamiento de la aplicación y los alineamientos con los que está construida.

El frontend está construido bajo las siguientes características:

1. **Framework Base**
   - **Vue.js:** Es el framework principal utilizado para desarrollar el frontend. Vue.js es un framework progresivo para construir interfaces de usuario reactivas y dinámicas.

2. **Componentes y Estilos**
   - **BootstrapVue:** Una implementación de Bootstrap para Vue, que proporciona componentes reutilizables y diseñados con Bootstrap, como botones, formularios y modales.
   - **SCSS (Sassy CSS):** Utilizado para gestionar los estilos personalizados de la aplicación. Esto se deduce de las líneas de importación de archivos SCSS y la referencia al archivo principal de estilos (assets/scss/style).
   - **Iconos: Font Awesome** que se emplea para incorporar iconos populares en los componentes y, **CoreUI Icons y Simple Line Icons** que son **i**conos adicionales utilizados para enriquecer la interfaz.

3. **Gestión de Estados y Rutas**
   - **Vuex:** Utilizado para la gestión del estado global de la aplicación, lo cual es evidente por la importación del archivo store/index.
   - **Vue Router:** Configura el sistema de navegación entre las diferentes vistas de la aplicación, permitiendo control de accesos según los roles del usuario.

4. **Comunicación con APIs**
   - **Axios:** Utilizado para realizar solicitudes HTTP hacia el backend. Las solicitudes están configuradas con un baseURL (APIENDPOINT), lo cual indica que el frontend interactúa con un backend mediante APIs REST.

5. **Plugins Adicionales**
   - **SweetAlert (VueSwal):** Para mostrar alertas modales con diseño atractivo.
   - **Vue Notification:** Para manejar notificaciones emergentes dentro de la aplicación.
   - **Vue Meta:** Para gestionar dinámicamente las metaetiquetas, como el título y descripción de las páginas.
   - **Vue Head:** Permite modificar las etiquetas <head> de manera programática.

6. **Compatibilidad**
   - **Core JS:** Incluye polyfills para compatibilidad con navegadores antiguos, lo que indica que la aplicación busca ser funcional en una amplia variedad de entornos.

7. **Diseño Responsivo**
   La integración de Bootstrap, SCSS y los estilos personalizados (como la regla html { scroll-behavior: smooth; }) sugiere que la aplicación está optimizada para ser responsiva y adaptarse a diferentes tamaños de pantalla.

8. **Características Especiales**
   - **Zoom SDK (ZoomMtg):** Se utiliza una biblioteca de Zoom (aparentemente para habilitar reuniones o videollamadas) y está disponible globalmente en la instancia Vue (Vue.prototype.ZoomMtg).

Tabla de contenido

Documentación frontend: En la carpeta frontend se encuentra todo lo relacionado con la visualización e interfaces que maneja la aplicación para los usuarios.

1. **Carpeta dist:** En la carpeta dist encontramos toda la compresión del proyecto frontend, con fines de realizar la importación y publicación en algún servidor o servicio.

2. **Carpeta public:** En esta carpeta encontramos archivos tales como imágenes y el index.html.

   2.1 Carpeta img: Carpeta que almacena las imágenes, avatares e iconos.
   - **Index.html:** Este archivo es el punto de entrada del frontend y proporciona el entorno necesario para que la aplicación se ejecute correctamente en el navegador.

3. **Carpeta src:**

   3.1 Assets: Contiene archivos estáticos para el diseño, hojas de estilo.
   - **scss:** Se encarga de organizar y contener todos los **estilos de la aplicación** mediante el uso de Sass, permitiendo una mayor **flexibilidad**, **mantenibilidad** y **escalabilidad** en el diseño y la apariencia del frontend. Desde la definición de variables globales hasta la implementación de correcciones para navegadores antiguos, esta carpeta centraliza y organiza todo lo necesario para aplicar estilos de forma eficiente en el proyecto.

   3.2 Containers: En esta carpeta se gestionan las interacciones más complejas de la aplicación, como el manejo de datos, la comunicación con el backend, y la integración con el estado global. Los containers gestionan los datos y la lógica, y pasan esta información a los componentes de presentación para ser visualizada de manera adecuada.

   3.2.1 Auth: Dentro de esta carpeta se encuentra el componente que se encargará de gestionar la vista del inicio de sesión dentro de la aplicación.
   3.2.2 Authqr: Dentro de esta carpeta se encuentra el componente que se encarga de la vista de autenticación, pero está más enfocado en un flujo de autenticación con QR, mostrando un código QR que el usuario debe escanear para autenticarse.

3.2.3 Back-end: En esta carpeta encontramos varios componentes anexados a diferentes carpetas, estas carpetas contienen lo siguiente:

En la carpeta **admin** encontramos el componente que se encarga de contenedor para el área administrativa.

En la carpeta **anfitrion** se encuentra el componente que es parte de un sistema más grande para gestionar información de reuniones o asambleas, mostrando detalles importantes como el quórum, los asistentes y otros datos en tiempo real, el tablero en tiempo real de la entrevista.

En la carpeta **components** se encuentran dos componentes el primero (**DefaultAside.vue**) se encarga de generar una interfaz de usuario dónde se puede ver información cómo: Una lista de eventos para hoy y mañana, con información sobre los participantes y lugares, mensajes de diferentes usuarios, con información adicional como hora y estado, y opciones de configuración del sistema con interruptores para activar o desactivar ciertas características. El segundo (**DefaultHeaderDropdownAccnt.vue**) se encarga de gestionar diversas actividades: Visualización de datos de usuario (Mostrar nombre, apellido y rol del usuario identificado), ofrece opciones como cambiar la contraseña y cerrar sesión, incluye un formulario dentro del modal que permite a los usuarios actualizar su contraseña, gestionar eventos como el cierre de sesión manual o por inactividad y ajustar dinámicamente ciertos elementos del menú al montar el componente.

En la carpeta **copropietario** se encuentra el componente encargado que cumple la función de gestionar y visualizar la información de reuniones de copropietarios en una aplicación web, realizando funcionalidades cómo: Carga y gestión de datos, interfaz dinámica, controles de roles y estados, y estilización y usabilidad.

En la carpeta **invitado** se encuentra el componente encargado de tener una interfaz para medir poderes otorgados a usuario y coeficientes de participación en el que se realiza un seguimiento del estado de cada usuario, su ingreso a la reunión.

En la carpeta **super-anfitrion** se encuentra básicamente la interfaz de usuario de una aplicación Vue.js que maneja navegación y presentación de contenido dinámico, todo ello con un enfoque en la usabilidad y la organización visual.

En la carpeta **Zoom** se encuentra el componente que genera la estructura base para un contenedor que permite integrar Zoom en la aplicación. Su funcionalidad está orientada a cargar componentes basados en las rutas, gestionar el título de la página y eventualmente incluir dependencias externas de Zoom (aunque estas están comentadas por ahora).

- **DefaultContainer.vue:** Este componente define la estructura base de una aplicación utilizando CoreUI para Vue.js. Proporciona un diseño de panel con: Encabezado, barra lateral izquierda, contenido principal, panel lateral derecho, y un pie de página. Navegación dinámica con menús, migas de pan, y vistas renderizadas según la ruta activa. Integra componentes reutilizables como formularios, togglers, y dropdowns para mejorar la experiencia del usuario.

3.3 Funciones: Contiene el archivo llamado funciones.js, este archivo se encarga de definir una función llamada messagevalidate que personaliza los mensajes de error utilizados por una biblioteca de validación (validate.js). La función ajusta los mensajes predeterminados para diferentes tipos de validación, como: **Presencia**: Valida que un campo no esté vacío. **Email**: Verifica que el formato del correo electrónico sea válido. **Numericalidad**: Comprueba si un valor es numérico, entero o si cumple una condición específica (por ejemplo, mayor que un número dado). **Longitud**: Valida si un texto tiene una longitud menor o igual a un límite especificado. **Igualdad**: Verifica si dos valores son iguales. **Regex**: Permite validar patrones personalizados mediante expresiones regulares.

3.4 Router: Contiene la redirección dentro del aplicativo.

- **Index.js:** Este archivo actúa como el núcleo de la navegación en la aplicación, asegurando que cada usuario acceda a las vistas correspondientes a su rol.
- **Indexordinal.js:** El objetivo de este archivo es gestionar las rutas de la aplicación, es decir, definir qué componentes se cargan según la URL que visite el usuario.

3.5 Services: Esta carpeta cuenta con una clase que centraliza las llamadas a una API en tu aplicación frontend. Esta clase actúa como una **abstracción** para realizar solicitudes HTTP usando la librería axios, proporcionando un método consistente y estructurado para interactuar con tu backend.

3.6 Shared: Esta carpeta tiene archivos de acceso compartido entre las diferentes vistas.

- **Classes.js:** Este archivo contiene configuraciones y utilidades relacionadas con las clases CSS utilizadas para el control de diseño y visualización de elementos responsivos en la aplicación. En particular, define clases para la barra lateral (sidebar), el menú lateral (aside-menu) y los puntos de interrupción (breakpoints). También proporciona una función para validar si un punto de interrupción es válido.
- **Toggle-classes.js:** Este archivo contiene una función que facilita el manejo dinámico de clases CSS en el elemento body del documento. La función toggleClasses permite alternar una clase específica en el body, mientras elimina las clases anteriores en una lista hasta llegar a la clase que se desea alternar. Además, permite forzar o no la adición de la clase. Útil cuándo se debe alternan las visualizaciones en diferentes dispositivos.
- **Utils.js**: El archivo contiene dos funciones útiles para realizar operaciones comunes en JavaScript: una para generar un número aleatorio dentro de un rango y otra para mezclar el orden de los elementos de un arreglo. La función de mezcla utiliza el algoritmo de Durstenfeld, una versión eficiente del algoritmo de Fisher-Yates.

3.7 Store: La carpeta contiene una carpeta llamada back-end, en esta carpeta hay otras dos carpetas, una llamada navigation y la otra llamada zoom, en estas carpetas se encuentran los archivos .js encargados de ... y el archivo index.js.

3.7.1 Back-end: En esta carpeta encontramoslalas acciones de las navegaciones, indexaciones, estados, mutaciones e indexaciones presentes en la navegación del sitio web o en el Zoom.

- **Index.js:** El archivo index.js configura la instancia de Vuex para la gestión de estado en una aplicación Vue.js. Vuex es un patrón de gestión de estado para aplicaciones Vue.js, y este archivo define los módulos Vuex que gestionan el estado relacionado con la navegación (infoNavigation) y el zoom (infoZoom). Además, se utiliza el plugin Vuex Persist para persistir el estado de la tienda en el almacenamiento local del navegador (usando localForage), asegurando que los datos no se pierdan entre recargas de la página.

3.8 Views: En esta carpeta se encuentran las visualizaciones en la aplicación, tales cómo las tablas de las interfaces, pies de página, modales y demás.

3.8.1 Auth: En esta carpeta encontramos los componentes y las tablas presentes en la vista de autenticación o login, también la estructura o vistas para el reinicio de la contraseña e inicio de sesión.

3.8.2 Back-end: En esta carpeta se encuentras la información que se muestra en los diferentes roles que se cuentan, tales cómo administrativo, anfitrión, copropietario e invitado. Dentro de las carpetas con el nombre de los roles están más carpetas que generan las diferentes acciones dentro de la vista perteneciente a determiando rol, también están presentes las tablas que aparecen en las vistas pertenecientes a cada rol.

3.8.3 Components: En esta carpeta se encuentra el componente encargado de mostrar un **ícono de carga** (un "spinner") mientras se realiza alguna operación, como la carga de datos o la espera de una respuesta del servidor.

3.8.4 Pages: En esta carpeta se encuentra los componentes o archivos encargados de la visualización de las páginas que se muestran cuándo se presenta un error 404 o 500.

3.8.5 Zoom: La carpeta zoom en un proyecto probablemente contiene todo lo relacionado con la integración de la plataforma Zoom: desde la autenticación y la obtención de firmas hasta la interacción con el SDK y la gestión de reuniones dentro de la aplicación.

_**navadmin.js:** Este archivo es una configuración para un sistema de navegación, específicamente para el panel de administración de una aplicación web. Este archivo exporta un objeto con una estructura de menús y elementos de navegación para el área administrativa.

_**navanfitrion.js:** Este archivo configura una lista de elementos para una barra de navegación (menú) en una aplicación web, específicamente para un usuario con rol de "anfitrión". Cada uno de los elementos de la lista tiene un nombre, una URL de destino y un ícono asociado.

_**navcopro.js:** Este archivo es una configuración de un menú de navegación (navbar) en una aplicación web, destinado para el rol de copropietario. Al igual que los códigos anteriores, está compuesto por un arreglo de elementos que definen cada ítem del menú.

_**navinvitado.js:** Este archivo define un menú muy sencillo para un **invitado**, permitiéndole acceder solo a la página de inicio del copropietario.

_**navsuperanfitrion.**js: Este archivo está diseñado para configurar el menú de navegación de un **super anfitrión.**

App.config.js: Este archivo se utiliza para definir las URL base para las API y las imágenes, dependiendo del entorno en el que se esté trabajando (desarrollo o producción).

App.vue: Este archivo es el componente raíz de la aplicación y se encarga de gestionar la vista principal, incluyendo la carga de rutas dinámicas y el estilo de la aplicación.

Boostarp.js: Este archivo se utiliza para configurar y proporcionar acceso global a varias herramientas importantes como Lodash, Popper.js, Axios, y posiblemente jQuery y Bootstrap si no están deshabilitados.

Main.js: Este código configura y arranca la aplicación Vue, asegurando que el usuario esté autenticado para acceder a las rutas correspondientes y gestionando la interfaz de usuario con diversas herramientas.