Para administrar archivos en Auco, generalmente sigues estos pasos básicos:

1. **Accede a la Plataforma**: Inicia sesión en la plataforma Auco utilizando tus credenciales.
2. **Subir Archivos**: Utiliza la opción de carga o subida de archivos para añadir documentos a la plataforma. Puedes arrastrar y soltar archivos o utilizar métodos de carga específicos según la interfaz de usuario de Auco.
3. **Organización y Etiquetado**: Una vez que los archivos están en la plataforma, puedes organizarlos utilizando etiquetas, categorías o carpetas según la estructura que Auco te permita establecer. Esto facilita la búsqueda y gestión posterior.
4. **Gestión de Versiones**: Si es necesario, Auco puede permitirte gestionar diferentes versiones de los archivos, asegurando que siempre tengas acceso a la última versión o a versiones anteriores según sea necesario.
5. **Acceso y Colaboración**: Auco podría ofrecer herramientas para gestionar permisos de acceso, permitiendo que usuarios específicos o equipos puedan ver, editar o compartir archivos según los roles y permisos asignados.
6. **Seguridad y Cumplimiento**: Auco probablemente tenga características de seguridad integradas para proteger tus archivos y cumplir con regulaciones de privacidad y cumplimiento.

Para detalles específicos sobre cómo realizar estas tareas en Auco, te recomendaría consultar la documentación de la plataforma o contactar con el soporte técnico de Auco para obtener orientación más detallada y específica según tus necesidades particulares.

**Crear una Plantilla en Auco desde la Interfaz de Usuario (UI)**

**Paso 1: Iniciar Sesión en Auco**

1. **Accede a la Plataforma Auco**: Ve al sitio web de Auco e inicia sesión con tus credenciales.
2. **Accede al Módulo de Plantillas**: Una vez que hayas iniciado sesión, busca y accede al módulo de plantillas o templates. Este nombre puede variar según la interfaz específica de Auco que estés utilizando.

**Paso 2: Crear una Nueva Plantilla**

1. **Selecciona Crear Nueva Plantilla**: Busca la opción o botón para crear una nueva plantilla. Esto generalmente está ubicado en una sección de administración de documentos o configuración avanzada.
2. **Completa los Detalles de la Plantilla**:
   * **Nombre**: Asigna un nombre descriptivo a la plantilla.
   * **Contenido**: Ingresa el contenido de la plantilla. Puedes usar un editor WYSIWYG o HTML para diseñar el formato y el estilo del documento.
3. **Configura Opciones Adicionales**:
   * Dependiendo de la plataforma, puedes tener opciones adicionales como configuraciones de firma, marcadores de posición dinámicos, y opciones de notificación.
4. **Guarda la Plantilla**: Una vez completados los detalles y la configuración, guarda la plantilla. Asegúrate de revisar cualquier configuración adicional antes de confirmar la creación.

**Paso 3: Prueba y Verificación**

1. **Prueba la Plantilla**: Después de crear la plantilla, realiza pruebas para asegurarte de que se renderiza correctamente y cumple con tus requisitos específicos.
2. **Verificación de Integración**: Si planeas integrar esta plantilla con sistemas externos o utilizarla en flujos de trabajo automatizados, verifica la integración para asegurarte de que los datos se envían y reciben correctamente.

**Consideraciones Adicionales**

* **Seguridad**: Mantén seguras tus claves de API y los datos sensibles.
* **Mantenimiento**: Actualiza y modifica las plantillas según sea necesario para mantener la precisión y la relevancia.
* **Soporte**: Si encuentras problemas o tienes preguntas sobre la creación de plantillas, contacta al soporte técnico de Auco para obtener ayuda adicional.

**Nota Importante**

La creación de plantillas a través de la interfaz de usuario puede variar ligeramente según la versión y configuración específica de Auco que estés utilizando. Siempre consulta la documentación y guías de usuario proporcionadas por Auco para obtener instrucciones precisas y actualizadas.

****

**Documentación de la función PostInfoHabeasData**

La función PostInfoHabeasData es una función asíncrona que realiza una petición HTTP POST para guardar información personal utilizando el servicio de Auco. La función maneja la solicitud y la respuesta, y actualiza el estado de la aplicación según sea necesario.

**Parámetros**

* context (Object): Este objeto contiene propiedades como commit y state, que son proporcionadas por Vuex para la gestión del estado.

**Detalles de la Función**

1. **Declaración de Variables**:
   * prk (String): Clave privada para la autorización de la API.
   * post (Object): Objeto que contiene los datos que se enviarán en la solicitud POST.
2. **Construcción del Objeto post**:
   * document (String): Identificador del documento.
   * sign (Boolean): Indicador de si se debe firmar el documento.
   * email (String): Correo electrónico del destinatario.
   * name (String): Nombre del documento.
   * data (Array): Arreglo de objetos con información personal del usuario.
   * camera (Boolean): Opción para habilitar la cámara.
   * otpCode (Boolean): Opción para habilitar el código OTP.
   * options (Object): Opciones adicionales para la solicitud, incluyendo el uso de WhatsApp y el código OTP.
3. **Impresión del Objeto post**:
   * Se imprime el objeto post en la consola para fines de depuración.
4. **Solicitud HTTP POST**:
   * Se realiza una solicitud HTTP POST a la URL https://dev.auco.ai/v1/ext/document/save con el objeto post y el encabezado de autorización.
   * En caso de éxito, se maneja la respuesta actualizando el estado de la aplicación y mostrando una notificación de éxito.
   * En caso de error, se muestra una notificación de error.
5. **Manejo de Excepciones**:
   * Se captura cualquier excepción durante el proceso y se imprime un mensaje de error en la consola.

**Ejemplo de Uso**

javascript

Copiar código

await PostInfoHabeasData({ commit, state });

**Código Completo**

javascript

Copiar código

PostInfoHabeasData: async function ({ commit, state }) {

try {

const prk = "prk\_JSBkX1S1MBpoELYEZEUZB2rpKy2XSRT4";

const post = {

"document": "665f28bf8ebd216cbab45699",

"sign": true,

"email": "lsalazar@gjx.ai",

"name": "Contrato de prueba API1",

"data": [

{ "key": "name\_paciente", "value": state.infoPersonal[0].Primer\_nombre + " " + state.infoPersonal[0].Segundo\_nombre + " " + state.infoPersonal[0].Primer\_apellido + " " + state.infoPersonal[0].Segundo\_apellido },

{ "key": "document\_type\_paciente", "value": state.infoPersonal[0].TipoDocumeto },

{ "key": "cedula\_paciente", "value": state.infoPersonal[0].Identificacion },

{ "key": "cedulae\_paciente", "value": state.infoPersonal[0].Identificacion },

{ "key": "email\_paciente", "value": state.infoPersonal[0].Email },

{ "key": "phone\_celular", "value": state.infoPersonal[0].TelefonoPrefijo + state.infoPersonal[0].PhoneWhatsapp },

{ "key": "nationality\_paciente", "value": state.infoPersonal[0].Pais },

{ "key": "phone\_paciente", "value": state.infoPersonal[0].TelefonoPrefijo + state.infoPersonal[0].PhoneWhatsapp },

{ "key": "date\_fecha", "value": formattedDate }

],

"camera": true,

"otpCode": true,

"options": {

"whatsapp": true,

"otpCode": "phone"

}

}

console.log(post);

axios.post("https://dev.auco.ai/v1/ext/document/save", post, {

headers: { Authorization: prk }

}).then(resp => {

console.log(resp);

console.log(resp.data.document);

commit('AsignacionInformacionPersonal', {

codeDocumentoFirma: resp.data.document

});

store.dispatch('GetInfoPersonal').then(

Swal.fire({

icon: 'success',

title: 'Enviado Correctamente',

text: 'Se registró la información con éxito',

confirmButtonColor: 'green'

})

);

})

.catch(error => {

Swal.fire({

icon: 'error',

title: 'Error',

text: 'Error en el registro',

allowOutsideClick: false,

backdrop: false,

confirmButtonColor: '#ff0000a8'

});

});

} catch (error) {

console.log("Error al obtener información de la tabla informacionPersonal", error, {

headers: { Authorization: Key }

});

}

},

**Notas**

* **Manejo de Errores**: Es importante manejar los errores adecuadamente para mejorar la experiencia del usuario y facilitar la depuración.
* **Seguridad**: Asegúrate de no exponer claves privadas o información sensible en el código fuente. En un entorno de producción, utiliza variables de entorno para almacenar estos valores.
* **Mejoras Futuras**: Considera refactorizar el código para separar la lógica de negocio de las llamadas a la API y mejorar la mantenibilidad del código.

La función GetInfoHabeasData es una función asíncrona que realiza una solicitud HTTP GET para obtener el estado de un documento utilizando el servicio de Auco. La función maneja la solicitud y la respuesta, y devuelve un valor booleano basado en el estado de la firma del documento.

**Parámetros**

* context (Object): Este objeto contiene propiedades como commit y state, que son proporcionadas por Vuex para la gestión del estado.

**Detalles de la Función**

1. **Impresión de Mensaje de Depuración**:
   * console.log("auxilioooo");: Imprime un mensaje en la consola para fines de depuración.
2. **Declaración de Variables**:
   * puk (String): Clave pública para la autorización de la API.
3. **Solicitud HTTP GET**:
   * Realiza una solicitud HTTP GET a la URL https://dev.auco.ai/v1/ext/document con los encabezados de autorización y el tipo de contenido.
   * Envía el código del documento (code) como parámetro de la solicitud. Este código se obtiene del estado state.infoPersonal[0].codeDocumentoFirma.
4. **Manejo de la Respuesta**:
   * Imprime el estado de la firma del documento en la consola.
   * Si el estado de la firma del documento (response.data.signatureProfile[0].status) es "FINISH", la función devuelve true.
   * De lo contrario, la función devuelve false.

**Ejemplo de Uso**

javascript

Copiar código

const resultado = await GetInfoHabeasData({ commit, state });

console.log("Resultado de la verificación:", resultado);

**Código Completo**

javascript

Copiar código

GetInfoHabeasData: async function ({ commit, state }) {

console.log("auxilioooo");

const puk = "puk\_d9lABKDf4tWTAHfGAh2SuFVWzDyrAXVv";

try {

var response = await axios.get('https://dev.auco.ai/v1/ext/document', {

headers: {

Authorization: puk,

'Content-Type': 'application/json'

},

params: { code: state.infoPersonal[0].codeDocumentoFirma }

});

console.log(response.data.signatureProfile[0].status);

if (response.data.signatureProfile[0].status == "FINISH") {

return true;

} else {

return false;

}

} catch (error) {

console.error("Error en la solicitud GET:", error);

return false;

}

}

**Notas**

* **Manejo de Errores**: Es importante manejar los errores adecuadamente para mejorar la experiencia del usuario y facilitar la depuración. En este ejemplo, se añadió un bloque try-catch para capturar y manejar cualquier error que ocurra durante la solicitud HTTP.
* **Seguridad**: Asegúrate de no exponer claves privadas o información sensible en el código fuente. En un entorno de producción, utiliza variables de entorno para almacenar estos valores.
* **Depuración**: Los mensajes console.log pueden ayudar en la depuración, pero considera removerlos o ajustarlos en un entorno de producción para evitar el exceso de registros en la consola.

**Mejoras Futuras**

* **Refactorización**: Considera refactorizar el código para separar la lógica de negocio de las llamadas a la API y mejorar la mantenibilidad del código.
* **Reintentos**: Implementar una lógica de reintentos en caso de fallos temporales en la solicitud a la API.
* **Notificaciones al Usuario**: Mejorar la retroalimentación al usuario en caso de errores o estados inesperados durante la verificación del documento.