



SISTEMA DE GESTIÓN DE REGISTROS INTERNOS PARA SDI.

Proyecto de Servicio Social



29 DE SEPTIEMBRE DE 2024

UNIVERSIDAD VERACRUZANA, SECRETARIA DE DESARROLLO INSTITUCIONAL
Xalapa, Veracruz

INDICE

Introducción	3
Objetivos del Sistema	3
Alcance	3
Requisitos del Sistema.....	4
Dependencias.....	4
Dependencias de Desarrollo (Dev Dependencies)	5
Arquitectura del Sistema	6
Diagrama de Arquitectura	6
Tecnologías Utilizadas	7
Especificaciones del Entorno.....	8
Casos de Uso	9
Caso de Uso 1: Iniciar Sesión	9
Caso de Uso 2: Registrar un Archivo.....	9
Caso de Uso 3: Buscar Documentos.....	9
Caso de Uso 4: Actualizar un Registro	10
Caso de Uso 5: Consultar Archivos	10
Diagrama de Secuencia	10
Manual del Usuario	11
Manual de Instalación y Configuración.....	11
Manual de Instalación.....	11
Despliegue en Vercel	12
Seguridad.....	13
Medidas Futuras a Implementar	13
Mantenimiento y Actualizaciones	14
Puntos Clave para un Mantenimiento Eficaz	15
Proceso de Actualización.....	15
Conclusiones.....	16
Estado Actual del Sistema.....	16
Mejoras Futuras.....	16
Próximos Pasos	17
Apéndices	18
Apéndice A: Configuraciones del Sistema	18
Apéndice B: Enlaces Importantes.....	19

Apéndice C: Scripts de Comandos.....	19
Apéndice D: Estructura del Proyecto.....	20
Apéndice E: Versiones de Software	20
Apéndice F: Bibliotecas y Dependencias Clave	20

Introducción

El presente documento detalla el desarrollo y funcionamiento del "Sistema de Gestión de Documentos de Auditoría" implementado para la Secretaría de Desarrollo Institucional (SDI) en 2024. Este sistema tiene como objetivo mejorar la organización, almacenamiento y gestión de los archivos relacionados con auditorías dentro de una plataforma basada en SharePoint.

El sistema permite a los usuarios institucionales registrar documentos asociados a auditorías, subir archivos y organizarlos dentro de un sistema de carpetas estructurado en SharePoint. Esto mejora la accesibilidad, garantiza un almacenamiento eficiente y facilita la búsqueda y consulta de los documentos.

Objetivos del Sistema

Optimizar la carga y organización de archivos digitales relacionados con auditorías, utilizando Microsoft SharePoint como plataforma de almacenamiento.

Facilitar el acceso a los documentos mediante una estructura de carpetas eficiente y un motor de búsqueda para localizar archivos específicos.

Asegurar la autenticación de los usuarios mediante MSAL (Microsoft Authentication Library), garantizando que solo los usuarios institucionales autorizados puedan acceder al sistema.

Alcance

Este sistema cubre las siguientes funcionalidades:

Inicio de sesión mediante cuentas institucionales de Microsoft.

Registro y carga de documentos relacionados con auditorías en SharePoint.

Organización de los documentos dentro de un sistema de carpetas estructurado.

Búsqueda y consulta de documentos registrados en la plataforma.

Actualización de metadatos y archivos previamente almacenados.

El sistema ha sido desarrollado utilizando las siguientes tecnologías:

React.js para el desarrollo del frontend.

Microsoft SharePoint para el almacenamiento de documentos.

Microsoft Graph API para la interacción con los servicios de SharePoint.

MSAL (Microsoft Authentication Library) para la autenticación de usuarios.

Flowbite para los componentes de UI.

Requisitos del Sistema

Categoría	Requisito
Funcionales	1. Autenticación mediante MSAL: El sistema debe permitir la autenticación de usuarios institucionales mediante MSAL.
	2. Registro de archivos: Permitir a los usuarios subir y registrar documentos relacionados con auditorías en SharePoint.
	3. Gestión de metadatos: Al registrar un archivo, se debe permitir asociar metadatos (nombre, fecha, descripción) y almacenarlos en un Excel en SharePoint.
	4. Búsqueda de documentos: Los usuarios deben poder buscar documentos mediante criterios como nombre, fecha o auditoría.
	5. Actualización de registros: Permitir la edición y actualización de metadatos de documentos registrados en el sistema.
	6. Visualización y descarga de archivos: Los usuarios deben poder visualizar y descargar archivos vinculados desde SharePoint.
No Funcionales	1. Rendimiento: El sistema debe procesar documentos eficientemente, permitiendo manejar al menos 20 archivos por día sin retrasos significativos.
	2. Seguridad: Solo los usuarios autenticados y autorizados deben acceder al sistema. Se deben utilizar tokens de acceso seguros de Microsoft y HTTPS.
	3. Disponibilidad: El sistema debe estar disponible el 95% del tiempo laboral y ser accesible desde cualquier dispositivo con conexión a Internet.
	4. Usabilidad: La interfaz debe ser intuitiva para que los usuarios no técnicos puedan usarla con facilidad.
	5. Compatibilidad: Debe ser compatible con navegadores modernos como Google Chrome, Microsoft Edge y Firefox.
	6. Mantenimiento: El sistema debe permitir actualizaciones fáciles de sus componentes sin interrumpir las operaciones.
	7. Escalabilidad: El sistema debe ser escalable, permitiendo manejar mayores volúmenes de documentos sin pérdida de rendimiento.

Dependencias		
Dependencia	Versión	Descripción
@azure/msal-browser	^3.23.0	Biblioteca para manejar la autenticación de Microsoft en el navegador.
@azure/msal-react	^2.0.22	Integración de MSAL con React para manejar la autenticación.
@microsoft/microsoft-graph-client	^3.0.7	Cliente para interactuar con Microsoft Graph API y acceder a los datos de Microsoft 365.
@radix-ui/react-slot	^1.1.0	Componente de Radix UI para gestionar slots en la interfaz de usuario.
@reduxjs/toolkit	^2.2.7	Conjunto de herramientas para la gestión de estado con Redux.

axios	^1.7.7	Cliente HTTP para hacer solicitudes a servicios REST.
class-variance-authority	^0.7.0	Biblioteca para la gestión de clases condicionales en componentes de UI.
clsx	^2.1.1	Biblioteca ligera para combinar clases CSS condicionales.
date-fns	^4.1.0	Herramientas para manipulación y formateo de fechas en JavaScript.
express	^4.21.0	Framework de Node.js para la construcción de aplicaciones web.
flowbite	^2.5.1	Biblioteca de componentes de UI basada en Tailwind CSS.
flowbite-react	^0.10.1	Integración de Flowbite con React.
lucide-react	^0.441.0	Iconos SVG para React.
multer	^1.4.5-lts.1	Middleware de Node.js para la gestión de la subida de archivos.
next	14.2.11	Framework de React para aplicaciones web con funcionalidades del lado del servidor.
pdf-lib	^1.17.1	Biblioteca para manipular archivos PDF en el navegador o servidor.
react	^18	Biblioteca para construir interfaces de usuario.
react-datepicker	^7.3.0	Componente de selección de fechas para React.
react-dom	^18	Complemento de React para la manipulación del DOM.
react-dropzone	^14.2.3	Componente para la gestión de áreas de arrastre de archivos en React.
react-redux	^9.1.2	Enlace de React para Redux para la gestión de estados globales.
react-toastify	^10.0.5	Biblioteca para mostrar notificaciones en React.
sp-request	^2.1.3	Cliente para realizar solicitudes HTTP hacia servicios de SharePoint.
tailwind-merge	^2.5.2	Herramienta para combinar clases de Tailwind CSS.
uuid	^10.0.0	Generador de identificadores únicos universales (UUID).
xlsx	^0.18.5	Biblioteca para leer y escribir archivos Excel en formato .xlsx.

Dependencias de Desarrollo (Dev Dependencies)		
Dependencia	Versión	Descripción
autoprefixer	^10.4.20	Añade prefijos CSS automáticamente para garantizar la compatibilidad entre navegadores.
eslint	^8.57.0	Herramienta para asegurar la calidad del código de JavaScript y detectar errores.
eslint-config-next	14.2.11	Configuración de ESLint para aplicaciones desarrolladas con Next.js.
eslint-config-prettier	^9.1.0	Desactiva reglas de ESLint que interfieren con Prettier.
eslint-plugin-prettier	^5.2.1	Integra Prettier con ESLint para formatear el código automáticamente.
eslint-plugin-react	^7.36.1	Complemento de ESLint para asegurar buenas prácticas en el desarrollo de React.

postcss	^8.4.47	Herramienta para transformar estilos CSS con plugins.
prettier	^3.3.3	Formateador de código para asegurar consistencia en el estilo.
tailwindcss	^3.4.11	Framework CSS de utilidad para construir diseños personalizados rápidamente.

Arquitectura del Sistema

Diagrama de Arquitectura		
Componente	Descripción	Interacción
Frontend (Next.js + React)	La interfaz de usuario que permite a los usuarios interactuar con el sistema, manejar formularios y visualizar datos.	Interactúa con el backend y las APIs a través de peticiones HTTP (por ejemplo, para autenticación y gestión de archivos).
Backend (Express)	Middleware ligero que puede manejar la carga de archivos y enrutamiento del lado del servidor.	Procesa peticiones del frontend y actúa como intermediario para las APIs externas (como SharePoint).
APIs (Microsoft Graph API)	Proporciona acceso a los servicios de Microsoft 365 como SharePoint para subir, consultar, y gestionar archivos y carpetas.	El backend interactúa directamente con Microsoft Graph API para gestionar archivos y carpetas en SharePoint.
Autenticación (MSAL)	Protocolo de autenticación de Microsoft utilizado para validar a los usuarios mediante tokens.	El frontend gestiona la autenticación con MSAL y el backend valida el acceso mediante tokens de Microsoft.

I. Frontend (React + Next.js)		
Carpeta	Descripción	Componentes/Archivos Clave
/src/app	Contiene las rutas principales del sistema. Cada carpeta en esta ruta corresponde a una página o sección.	AudiMana, FormAuditorias, setting, page.js, layout.js
/src/componentes	Contiene los componentes reutilizables, organizados por funcionalidad.	AuditoriasForm.jsx, AuditoriasManagment.jsx, FileUploader.jsx
/src/assets/fonts	Almacena las fuentes utilizadas en el sistema.	GeistMonoVF.woff, GeistVF.woff
/src/componentes/auditorias	Componentes para gestionar archivos de auditorías.	AuditoriasForm.jsx, SelectCarpetaFisica.jsx
/src/componentes/heroSeccion	Componentes de la página principal, como el navbar y el footer.	Bienvenida.jsx, Navbar.jsx, Footer.jsx, ButtonAuth.jsx

2. Hooks Personalizados		
Carpeta	Descripción	Hooks Clave
/src/hooks/auditorias	Contiene hooks personalizados para manejar operaciones relacionadas con auditorías.	useExcel.js, useSharePoint.js, useAuth.js

3. Servicios (APIs Externas)		
Carpeta	Descripción	Servicios Clave
/src/services	Gestiona la interacción con servicios externos como MSAL y Microsoft Graph API.	authConfig.js, authService.js, graphService.js

4. Dependencias		
Archivo	Descripción	Dependencias Clave
package.json	Define las dependencias utilizadas en el proyecto.	@azure/msal-browser, @reduxjs/toolkit, axios, next, react, sp-request, tailwindcss

5. Estilos		
Carpeta/Archivo	Descripción	Archivos Clave
/src/app/setting/globals.css	Contiene los estilos globales de la aplicación.	globals.css
tailwind.config.js	Configuración de Tailwind CSS para los estilos personalizados.	tailwind.config.js

6. Backend (Express)		
Dependencia	Descripción	Archivo Clave
express	Middleware utilizado para manejar ciertas rutas o cargar archivos en el servidor, posiblemente con multer.	package.json (definido como dependencia)

7. APIs Externas		
API	Descripción	Archivo Clave
Microsoft Graph API	Permite interactuar con servicios de Microsoft 365, como SharePoint, para la gestión de archivos y carpetas.	graphService.js, useSharePoint.js
MSAL (Microsoft Authentication Library)	Maneja la autenticación de usuarios a través de cuentas de Microsoft.	authService.js, authConfig.js, useAuth.js

Tecnologías Utilizadas		
Tecnología/Herramienta	Descripción	Función en el Sistema

React.js	Biblioteca de JavaScript para construir interfaces de usuario.	Desarrollo del frontend.
Next.js	Framework de React con funcionalidades del lado del servidor.	Enrutamiento de páginas y rendering del lado del servidor.
MSAL (Microsoft Authentication Library)	Biblioteca para autenticación de usuarios mediante cuentas de Microsoft.	Maneja el inicio de sesión y la autenticación segura de los usuarios.
Microsoft Graph API	API que permite interactuar con los servicios de Microsoft 365, como SharePoint.	Gestión de archivos y carpetas en SharePoint.
Express	Framework de servidor de Node.js que maneja rutas y procesamiento de solicitudes.	Backend para gestionar peticiones HTTP.
Flowbite	Biblioteca de componentes de UI basada en Tailwind CSS.	Para la interfaz de usuario moderna y responsive.
SharePoint	Plataforma de almacenamiento y colaboración de Microsoft utilizada para gestionar archivos.	Almacenamiento de archivos y carpetas.
Tailwind CSS	Framework CSS para construir interfaces de usuario con un diseño personalizado.	Estilización de los componentes de React.
Axios	Cliente HTTP para realizar solicitudes a APIs.	Utilizado para realizar peticiones a Microsoft Graph API y el backend.

Especificaciones del Entorno		
Entorno	Especificación	Descripción
Sistema Operativo (Desarrollo)	Windows 10 / macOS / Linux (dependiendo del entorno de desarrollo)	Entorno principal para el desarrollo del sistema.
Sistema Operativo (Producción)	Plataforma en la nube o servidor Linux para desplegar la aplicación.	Alojamiento en producción.
Node.js	Versión 16.x o superior	Entorno de ejecución para JavaScript y backend con Express.
Next.js	Versión 14.x	Framework utilizado para frontend con React y backend ligero.
React.js	Versión 18.x	Biblioteca de frontend.
Tailwind CSS	Versión 3.x	Framework CSS para estilización del frontend.
Microsoft Graph API	Última versión disponible	Para interactuar con SharePoint y otros servicios de Microsoft 365.
MSAL	Versión 3.x	Biblioteca de autenticación para Microsoft.

Git	Sistema de control de versiones (versión 2.x o superior)	Utilizado para el control de versiones del proyecto.
NPM	Versión 7.x o superior	Para la gestión de dependencias.
Eslint / Prettier	Herramientas para asegurar la calidad y el formateo del código.	Se utilizan durante el desarrollo para asegurar la consistencia del código.

Casos de Uso

Caso de Uso 1: Iniciar Sesión

Caso de Uso	Iniciar sesión
Actor	Usuario institucional
Descripción	El usuario inicia sesión en el sistema utilizando su cuenta institucional de Microsoft mediante el protocolo de autenticación MSAL.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona la opción para iniciar sesión. 2. El sistema redirige al servicio de autenticación de Microsoft. 3. El usuario ingresa sus credenciales de la cuenta institucional. 4. Microsoft verifica las credenciales y redirige de vuelta al sistema con un token de acceso. 5. El sistema valida el token de acceso y permite al usuario entrar.
Flujo Alternativo	Si las credenciales no son válidas o no pertenecen a una cuenta institucional, se muestra un mensaje de error.
Precondiciones	El usuario debe tener una cuenta institucional de Microsoft.
Postcondiciones	El usuario queda autenticado y puede acceder a las funcionalidades del sistema.

Caso de Uso 2: Registrar un Archivo

Caso de Uso	Registrar un archivo
Actor	Usuario institucional
Descripción	El usuario sube un archivo y lo registra en un Excel que está en SharePoint.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona la opción para registrar un archivo. 2. El sistema muestra un formulario donde el usuario completa los datos del archivo. 3. El usuario selecciona un archivo para subirlo. 4. El sistema guarda el archivo en la misma carpeta que contiene el Excel en SharePoint. 5. Los metadatos del archivo (nombre, fecha, etc.) se registran en una hoja del Excel, y se guarda el enlace para acceder al archivo. 6. El sistema confirma que el archivo ha sido registrado exitosamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. Si el archivo ya existe, el sistema muestra un mensaje de advertencia. 2. Si la carga del archivo falla, se muestra un mensaje de error.
Precondiciones	El usuario debe estar autenticado.
Postcondiciones	El archivo queda registrado en SharePoint y el Excel se actualiza con los metadatos y el enlace del archivo.

Caso de Uso 3: Buscar Documentos

Caso de Uso	Buscar documentos
Actor	Usuario institucional

Descripción	El usuario busca documentos previamente registrados en el Excel en SharePoint.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona la opción para buscar documentos. 2. El sistema muestra una interfaz de búsqueda donde el usuario puede ingresar criterios de búsqueda (nombre, fecha, etc.). 3. El sistema consulta el Excel en SharePoint y muestra una lista de resultados que coinciden con los criterios. 4. Si un archivo tiene un enlace para visualización, se muestra junto con los resultados. 5. El usuario puede seleccionar un registro para ver más detalles.
Flujo Alternativo	Si no se encuentran resultados, se muestra un mensaje indicando que no hay coincidencias.
Precondiciones	El usuario debe estar autenticado.
Postcondiciones	Se muestran los documentos que coinciden con los criterios de búsqueda.

Caso de Uso 4: Actualizar un Registro

Caso de Uso	Actualizar un registro
Actor	Usuario institucional
Descripción	El usuario puede actualizar un registro previamente guardado en el Excel en SharePoint.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona un registro de la lista de documentos. 2. El sistema permite al usuario modificar los metadatos del archivo (nombre, fecha, descripción, etc.). 3. El usuario guarda los cambios. 4. El sistema actualiza el Excel con la información modificada. 5. El sistema confirma que los cambios han sido guardados.
Flujo Alternativo	Si el archivo no puede ser actualizado por algún error, se muestra un mensaje de advertencia.
Precondiciones	El usuario debe estar autenticado y debe tener permisos para actualizar el registro.
Postcondiciones	El registro es actualizado exitosamente en el Excel en SharePoint.

Caso de Uso 5: Consultar Archivos

Caso de Uso	Consultar archivos
Actor	Usuario institucional
Descripción	El usuario consulta los registros del Excel en SharePoint y visualiza archivos vinculados.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario selecciona un registro en el Excel. 2. Si el registro tiene un archivo vinculado, el sistema muestra un enlace para visualizar o descargar el archivo. 3. El usuario puede hacer clic en el enlace para abrir o descargar el archivo desde SharePoint.
Flujo Alternativo	Si no hay un archivo vinculado, se muestra un mensaje indicando que no existe archivo para este registro.
Precondiciones	El usuario debe estar autenticado.
Postcondiciones	El usuario puede visualizar o descargar el archivo vinculado desde SharePoint.

Diagrama de Secuencia

Flujos principales del sistema...

Manual del Usuario

Pasos para usar el sistema: Cómo iniciar sesión, cómo registrar archivos, buscar documentos, actualizar registros, etc.

Capturas de pantalla: Incluir imágenes de las interfaces clave.

Mensajes de error comunes y cómo solucionarlos.

Manual de Instalación y Configuración

Manual de Instalación	
Paso	Descripción
1. Requisitos Previos	<ul style="list-style-type: none">- Instalar Node.js (versión 16.x o superior).- Instalar NPM o Yarn.- Instalar Git para clonar el repositorio.
2. Clonar el Repositorio	<ul style="list-style-type: none">- Clona el repositorio con el comando: <code>git clone https://github.com/tu-repositorio/archivo-sdi.git</code>- Navega al directorio del proyecto: <code>cd archivo-sdi</code>
3. Instalar Dependencias	<ul style="list-style-type: none">- Ejecuta el siguiente comando para instalar las dependencias: <code>npm install</code> Esto instalará todas las dependencias listadas en package.json.
4. Configuración del Sistema	
4.1 Configurar MSAL (Autenticación)	<ul style="list-style-type: none">- En el archivo authConfig.js, configura los siguientes parámetros: clientId, authority, y redirectUri con los valores de tu aplicación en Azure.
4.2 Configurar Microsoft Graph API	<ul style="list-style-type: none">- En el archivo graphService.js, asegúrate de que las configuraciones de siteId y driveId de SharePoint estén correctas.
5. Configuración de Tailwind CSS	<ul style="list-style-type: none">- Configura los estilos globales en el archivo tailwind.config.js si necesitas personalizar la interfaz.- Asegúrate de que las rutas de tus archivos React estén incluidas en la propiedad content.
6. Configuración de Flowbite	<ul style="list-style-type: none">- Para utilizar Flowbite, asegúrate de que Flowbite está instalado correctamente (verifica en package.json).- Importa Flowbite en tu archivo CSS global (globals.css): <code>@import 'flowbite';</code>- Utiliza los componentes de UI de Flowbite en tu código React como se especifica en la documentación oficial de Flowbite. Flowbite React - UI Component Library (flowbite-react.com)

Despliegue en Vercel	
Paso	Descripción
1. Crear cuenta en Vercel	<ul style="list-style-type: none"> - Ve a vercel.com y crea una cuenta si aún no la tienes. - Puedes iniciar sesión con tu cuenta de GitHub para facilitar la integración.
2. Importar tu Repositorio	<ul style="list-style-type: none"> - Desde el dashboard de Vercel, selecciona "Nuevo Proyecto". - Conecta tu repositorio GitHub, GitLab o Bitbucket donde se aloja tu proyecto.
3. Configurar Variables de Entorno	<ul style="list-style-type: none"> - En la sección de configuración de tu proyecto en Vercel, configura las variables de entorno necesarias (por ejemplo, <code>NEXT_PUBLIC_GRAPH_CLIENT_ID</code>, <code>NEXT_PUBLIC_TENANT_ID</code>).
4. Seleccionar la Configuración por Defecto	<ul style="list-style-type: none"> - Vercel detectará automáticamente que tu proyecto es un Next.js y configurará las opciones adecuadas para el build y despliegue.
5. Desplegar el Proyecto	<ul style="list-style-type: none"> - Haz clic en "Deploy". Vercel construirá tu proyecto y lo desplegará automáticamente en una URL pública que puedes compartir.
6. Producción y Pruebas	<ul style="list-style-type: none"> - Una vez que el despliegue se complete, la aplicación estará disponible en un dominio proporcionado por Vercel. - Puedes personalizar el dominio si lo necesitas.
7. Actualizaciones Automáticas	<ul style="list-style-type: none"> - Cada vez que realices un git push a tu repositorio, Vercel desplegará automáticamente la nueva versión de la aplicación.

¿Por qué Vercel?

Integración Perfecta con Next.js: Vercel es el creador de Next.js, por lo que proporciona la mejor integración y rendimiento para proyectos basados en este framework.

Despliegue Automático: Los cambios en tu repositorio se sincronizan y despliegan automáticamente.

Optimización de Imagen y Datos Estáticos: Vercel optimiza de manera automática las imágenes y archivos estáticos de tu aplicación, mejorando la experiencia de usuario y el rendimiento.

Escalabilidad: La plataforma maneja de manera automática la escalabilidad, ajustando la capacidad de tu aplicación según la demanda.

Seguridad

Aspecto de Seguridad	Descripción
1. Autenticación (MSAL)	<ul style="list-style-type: none"> - El sistema utiliza MSAL para autenticar a los usuarios mediante sus cuentas de Microsoft. - Los usuarios deben iniciar sesión con sus credenciales institucionales, y se generan tokens de acceso para autenticar y autorizar el acceso a los recursos.
2. Autorización mediante Tokens	<ul style="list-style-type: none"> - Los tokens de acceso generados por MSAL se utilizan para realizar solicitudes a Microsoft Graph API y a recursos protegidos como SharePoint. Los usuarios no autenticados no pueden acceder a los recursos.
3. HTTPS (Cifrado de Datos en Tránsito)	<ul style="list-style-type: none"> - Todo el tráfico entre el cliente y el servidor se cifra mediante HTTPS, lo que asegura que los datos sensibles no se intercepten. - Vercel ya implementa HTTPS automáticamente para proteger la aplicación en producción.
4. Almacenamiento de Tokens	<ul style="list-style-type: none"> - Los tokens de acceso se almacenan en el navegador usando LocalStorage o SessionStorage. Es importante que estos tokens solo se usen en aplicaciones servidas a través de HTTPS para evitar posibles vulnerabilidades.
5. Gestión de Sesiones	<ul style="list-style-type: none"> - Los tokens tienen una duración limitada y deben renovarse para mantener la sesión activa. Si el token expira, el usuario debe iniciar sesión nuevamente para continuar accediendo a los recursos protegidos.
6. Protección contra XSS (Cross-Site Scripting)	<ul style="list-style-type: none"> - En las áreas donde se muestra información generada por el usuario, debes asegurarte de escapar y sanitizar el contenido antes de renderizarlo en el DOM. Esto ayudará a prevenir ataques XSS. (Puedes considerar agregar esta protección más adelante).

Medidas Futuras a Implementar

Aspecto	Descripción
1. Protección contra CSRF (Cross-Site Request Forgery)	Implementar tokens anti-CSRF para proteger la aplicación contra ataques que intenten realizar solicitudes en nombre del usuario autenticado. Esto puede ser necesario en áreas donde se envían formularios sensibles.
2. Auditoría y Monitoreo	Implementar registros para actividades críticas (inicios de sesión, cambios en archivos, etc.) y utilizar herramientas de monitoreo para detectar comportamientos sospechosos o accesos no autorizados. Esto ayudará a mejorar la seguridad del sistema.
3. Control de Acceso Basado en Roles (RBAC)	Definir roles más específicos dentro de la aplicación para garantizar que solo los usuarios con permisos adecuados puedan realizar ciertas acciones (por ejemplo, administrador versus usuario estándar).

Mantenimiento y Actualizaciones

Aspecto	Descripción
1. Actualización de Dependencias	<ul style="list-style-type: none">- Utiliza el comando <code>npm update</code> regularmente para asegurarte de que todas las dependencias están actualizadas.- Revisa las notas de la versión de bibliotecas críticas como Next.js, React, MSAL, y Microsoft Graph API para estar al tanto de posibles cambios de compatibilidad.
2. Manejo de Versiones	<ul style="list-style-type: none">- Implementa un sistema de control de versiones usando Git.- Asegúrate de utilizar ramas separadas para nuevas funcionalidades o correcciones antes de fusionarlas a la rama principal (main o master).- Considera el uso de tags para marcar las versiones estables.
3. Pruebas antes del Despliegue	<ul style="list-style-type: none">- Antes de cualquier actualización en producción, prueba los cambios en un entorno de desarrollo o en una rama de staging.- Utiliza herramientas de pruebas unitarias o de integración para asegurarte de que las funcionalidades claves siguen funcionando.
4. Mantenimiento Preventivo	<ul style="list-style-type: none">- Realiza auditorías periódicas del código para detectar dependencias obsoletas o vulnerabilidades de seguridad potenciales.- Usa herramientas de análisis de vulnerabilidades como npm audit para detectar problemas de seguridad.
5. Monitoreo de la Aplicación	<ul style="list-style-type: none">- Utiliza herramientas de monitoreo como Vercel Analytics o Google Analytics para rastrear el comportamiento de la aplicación en producción.- Registra eventos críticos y errores en producción para poder corregir problemas de manera proactiva.
6. Solución de Problemas Comunes	<ul style="list-style-type: none">- Si la aplicación no se despliega correctamente en Vercel, revisa los logs de despliegue disponibles en el dashboard de Vercel.- Para problemas con autenticación (MSAL), asegúrate de que los tokens están siendo gestionados y almacenados correctamente en el navegador.
7. Limpieza de Dependencias	<ul style="list-style-type: none">- Ejecuta el comando <code>npm prune</code> para eliminar las dependencias no utilizadas del proyecto, ayudando a mantener limpio el entorno de desarrollo.- Esto es importante para evitar acumulación innecesaria de librerías que ya no son necesarias.
8. Despliegue Automatizado	<ul style="list-style-type: none">- En Vercel, las actualizaciones se despliegan automáticamente cuando realizas un git push al repositorio. Sin embargo, asegúrate de probar todos los cambios localmente antes de hacer el push.
9. Backup y Recuperación	<ul style="list-style-type: none">- Mantén respaldos regulares del código fuente en un sistema de control de versiones como GitHub o GitLab.

	- Asegúrate de tener una copia de seguridad de las configuraciones importantes, como claves de API y credenciales de autenticación en un lugar seguro.
10. Documentación	<ul style="list-style-type: none"> - Mantén la documentación actualizada a medida que se introducen nuevas funcionalidades o se modifican componentes del sistema. - Incluir descripciones claras de cómo realizar tareas comunes como despliegues, actualización de dependencias, etc.

Puntos Clave para un Mantenimiento Eficaz

Mantener dependencias actualizadas para asegurar la compatibilidad y seguridad del sistema.

Revisar logs y errores para corregir cualquier problema antes de que afecte a los usuarios.

Pruebas exhaustivas antes de despliegues para minimizar problemas en producción.

Documentar cualquier cambio importante para que otros desarrolladores o miembros del equipo puedan mantenerse al tanto.

Proceso de Actualización

Actualizar dependencias:

Ejecutar **npm update** para actualizar las bibliotecas.

Revisar posibles advertencias y problemas de compatibilidad.

Probar cambios:

Realizar pruebas en el entorno de desarrollo (local) y staging.

Verificar la autenticación, subida de archivos, y otras funcionalidades clave.

Deploy en Vercel:

Hacer git push al repositorio remoto para desplegar los cambios automáticamente en Vercel.

Monitoreo en Producción:

Verificar el correcto funcionamiento de la aplicación usando herramientas de monitoreo.

Conclusiones

El sistema de gestión de auditorías desarrollado para la Secretaría de Desarrollo Institucional (SDI) se encuentra actualmente en fase de pruebas. Hasta ahora, se ha implementado uno de los cuatro tipos de registros planificados, lo que ha permitido comenzar a evaluar el rendimiento y la funcionalidad del sistema en un entorno controlado.

Logros Principales

Primera Fase de Implementación Exitosa:

La implementación del primer tipo de registro ha permitido validar la integración entre el sistema y Microsoft SharePoint, utilizando Microsoft Graph API para gestionar el almacenamiento de documentos de auditoría de manera eficiente.

Autenticación Segura:

El sistema utiliza MSAL para garantizar que solo los usuarios institucionales autorizados puedan acceder y manejar los archivos de auditoría, proporcionando un control seguro de acceso.

Interfaz de Usuario Intuitiva:

Gracias a React.js y Next.js, el sistema ofrece una interfaz amigable y eficiente para los usuarios, permitiéndoles registrar y gestionar documentos de manera sencilla.

Estado Actual del Sistema

El sistema ha logrado implementar el primer tipo de registro de auditorías, lo cual ha sido fundamental para comenzar a validar la funcionalidad base del sistema. El enfoque inicial ha sido garantizar que el registro y la gestión de documentos en SharePoint sean fiables y que la autenticación de usuarios mediante MSAL funcione correctamente.

A medida que se completan las pruebas, se espera ajustar y optimizar el sistema para garantizar su estabilidad antes de proceder con la implementación de los otros tres tipos de registros pendientes.

Mejoras Futuras

Con las pruebas en curso, las siguientes áreas de mejora ya están identificadas:

Finalización de los 4 Tipos de Registros:

El objetivo inmediato es completar la implementación de los otros tres tipos de registros planificados.

Pruebas y Optimización del Rendimiento:

Realizar pruebas de carga y optimización del rendimiento para asegurar que el sistema pueda manejar el volumen esperado de documentos y usuarios.

Mejoras en la Seguridad:

Una vez completada la implementación de los registros, se pueden agregar medidas de

seguridad adicionales, como protección contra ataques CSRF y mejoras en la gestión de sesiones.

Próximos Pasos

Completar la implementación de los registros restantes y ajustar el sistema según los resultados de las pruebas actuales.

Monitorear el comportamiento del sistema en producción, una vez que se implemente completamente, para asegurarse de que las funcionalidades básicas y de seguridad estén operando según lo previsto.

Documentar todas las fases del desarrollo e implementación para asegurar una transición fluida hacia las fases finales del proyecto.

En conclusión, el sistema ha avanzado con éxito en la primera fase de implementación, enfocándose en validar el flujo de trabajo para el primer tipo de registro de auditoría. Con los otros tres tipos de registros aún pendientes, el sistema seguirá evolucionando a medida que se completen las pruebas y se implementen las funcionalidades restantes. Este proceso garantizará una solución robusta y eficiente para la SDI.

Apéndices

Apéndice A: Configuraciones del Sistema

Configuración de MSAL (Microsoft Authentication Library)

Archivo: authConfig.js

Parámetros principales:

```
const msalConfig = {  
  auth: {  
    clientId: "tu-clientId",  
    authority: "https://login.microsoftonline.com/tu-tenant-id",  
    redirectUri: "http://localhost:3000",  
  },  
  cache: {  
    cacheLocation: "localStorage",  
    storeAuthStateInCookie: false,  
  }  
};
```

Configuración de Microsoft Graph API

Archivo: graphService.js

Parámetros principales:

```
const siteId = 'tu_site_id_aqui';  
const driveId = 'tu_drive_id_aqui';
```

Configuración de Tailwind CSS

Archivo: tailwind.config.js

Configuración para incluir todos los archivos de la aplicación:

```
module.exports = {
  content: [
    './src/**/*..{js,jsx,ts,tsx}',
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

Apéndice B: Enlaces Importantes

- Documentación de Next.js
<https://nextjs.org/docs>
- Documentación de React.js
<https://reactjs.org/docs>
- Documentación de Microsoft Graph API
<https://docs.microsoft.com/en-us/graph/overview>
- Documentación de MSAL (Microsoft Authentication Library)
<https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview>
- Documentación de Flowbite
<https://flowbite.com/docs/getting-started/react/>

Apéndice C: Scripts de Comandos

Scripts de Desarrollo y Despliegue


- Iniciar el servidor de desarrollo:

```
npm run dev
```

- Compilar el proyecto para producción:

```
npm run build
```

- Iniciar el servidor en producción:



```
npm start
```

- **Despliegue Automático en Vercel:**
Realiza un git push al repositorio conectado a Vercel para un despliegue automático.

Apéndice D: Estructura del Proyecto

Estructura de Carpetas Clave

- **/src/app:** Contiene las páginas y rutas de la aplicación.
- **/src/componentes:** Contiene los componentes de React.
- **/src/services:** Servicios para la autenticación y la interacción con Microsoft Graph API.
- **/src/hooks:** Hooks personalizados para la gestión de auditorías y la autenticación.
- **/src/assets:** Archivos estáticos como fuentes.
-

Apéndice E: Versiones de Software

- **Node.js:** Versión 16.x o superior
- **Next.js:** Versión 14.x
- **React.js:** Versión 18.x
- **Tailwind CSS:** Versión 3.x
- **MSAL (Microsoft Authentication Library):** Versión 3.x
- **Microsoft Graph API:** Última versión disponible

Apéndice F: Bibliotecas y Dependencias Clave

- **React.js:** Para el desarrollo del frontend.
- **Next.js:** Framework de React para aplicaciones del lado del servidor.
- **MSAL:** Para la autenticación segura de usuarios.
- **Microsoft Graph API:** Para la interacción con SharePoint y otros servicios de Microsoft 365.
- **Flowbite:** Para los componentes de la interfaz de usuario.
- **Tailwind CSS:** Para la estilización rápida y personalizada de la interfaz.