Requerimientos

1.- Autenticación: El sistema debe permitir que el administrador inicie sesión con usuario y contraseña para acceder a sus funciones.

2.- Carga de archivos: El sistema debe permitir al administrador subir archivos Excel para su procesamiento.

3.- Vista previa de archivos: El sistema debe proporcionar una vista previa del contenido del archivo cargado antes de procesarlo.

4.- Procesamiento de datos: El sistema debe ejecutar cálculos sobre los datos almacenados en los archivos cargados.

5.- Descarga de resultados: El sistema debe permitir al administrador descargar un archivo Excel con los resultados obtenidos.

Especificaciones

1.- -Implementación de un sistema de autenticación con Firebase

-Endpoint /login para validar usuario y contraseña

-Middleware para proteger rutas privadas

2.- - Uso de Multer para la carga de archivos en el backend

- Restricción a formatos .xlsx

- Validación del contenido antes de su almacenamiento

3.- - Lectura del archivo con xlsx-populate

4.- - Backend con funciones de procesamiento de datos (Node.js con xlsx-populate)

5.- - Generación del archivo Excel con xlsx-populate

- Botón en el frontend para iniciar la descarga al terminar de procesar el documento

Requisitos funcionales

Inicio de sesión:

- El sistema debe validar las credenciales del administrador antes de permitir el acceso.
- Debe haber un mecanismo de recuperación de contraseña.

Carga de archivos:

- Debe permitir la carga de archivos en formato .xlsx y .xls.
- El sistema debe rechazar archivos de formato incorrecto o que excedan los 10MB.
- El archivo cargado debe almacenarse temporalmente para su procesamiento.

Vista previa de archivos:

- El sistema debe mostrar las primeras 10 filas y 5 columnas del archivo antes de procesarlo.
- Debe permitir al usuario cancelar la carga si el archivo es incorrecto.

Procesamiento de datos:

- El sistema debe ejecutar cálculos basados en reglas definidas sobre los datos del archivo.
- Los resultados deben almacenarse en una base de datos para su consulta posterior.

Descarga de resultados:

- El sistema debe generar un archivo Excel con los datos procesados.
- Debe proporcionar un botón de descarga en la interfaz.

Requisitos no funcionales

Seguridad:

- Las contraseñas deben almacenarse de forma segura usando hashing con bcrypt.
- La sesión del usuario debe expirar tras 30 minutos de inactividad.

Escalabilidad:

- El sistema debe soportar al menos 100 archivos procesados por semana sin afectar el rendimiento.

Usabilidad:

- La interfaz debe ser intuitiva y permitir la carga de archivos con drag & drop.

Compatibilidad:

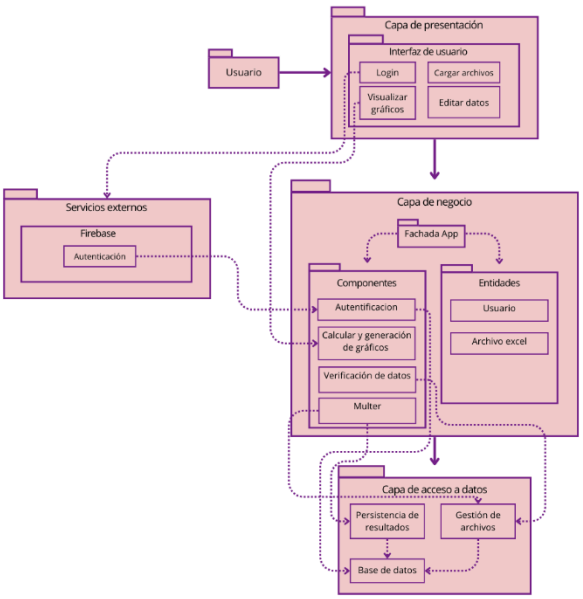
- El sistema debe ser accesible desde Google Chrome, Mozilla Firefox y Microsoft Edge.

Disponibilidad:

- El sistema debe estar disponible al menos el 99% del tiempo.

Modelo Arquitectónico

Diagrama de componentes



Casos de prueba (por lo menos 2)

ID	Descripcion	Supuesto	pasos	Resultados (Esperado)	Resultados(Real)
1	Validar que el usuario pueda iniciar sesión con credenciales correctas.	url: localhost:3009/ Email:ief_infor@test.cl Pass: ief_infor123	1) Ingresar url 2) Ingresar Credenciales 3) Click : Ingresar	redireccion a a la pagina localhost:3009/home	redirecciona a la pagina localhost:3009/home
2	Verificar que el sistema permita la carga de archivos Excel válidos.	Se encuentra en localhost:3009/home	1) Presionar seleccionar archivo 2) seleccionar archivo 2024_BDA_4UTCUTS.xlsx de la carpeta Descargas 3) Presionar Subir	El archivo ha sido subido exitosamente	El archivo ha sido subido exitosamente
3)	Validar que el usuario pueda iniciar	url: localhost:3009/ Email:ief_infor@test	1) Ingresar url 2) Ingresar	credenciales invalidas	credenciales invalidas

	sesión con credenciales incorrectas.	.cl Pass: contraseña	Credenciales 3) Click : Ingresar		
4)	Verificar que la interfaz permite la carga de archivos mediante drag & drop	Se encuentra en localhost:3009/home	1) Estar en la página localhost:3009/home. 2) Seleccionar un archivo válido (.xlsx o .xls) desde el explorador de archivos del sistema operativo. 3) Arrastrar el archivo hacia el área designada para la carga en la interfaz web. 4) Soltar el archivo en el área de carga. 5) Presionar subir	El archivo ha sido subido exitosamente	El archivo ha sido subido exitosamente
5)	Verificar que el sistema es accesible desde Microsoft Edge	El usuario tiene instalado Microsoft Edge	1) Abrir el navegador Microsoft Edge 2) Ingresar la URL del sistema (localhost:3009/)	Se abre la pagina localhost:3009/login	Se abre la pagina localhost:3009/login

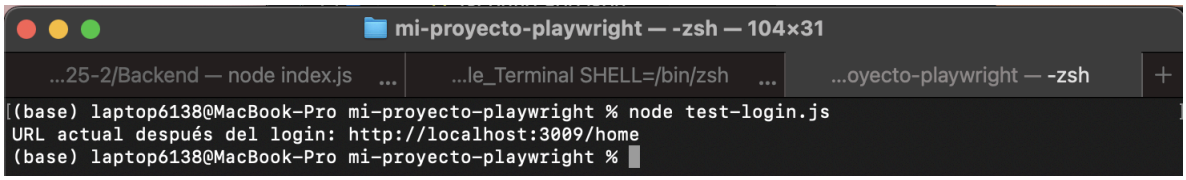
Automatice

En nuestro caso utilizamos playwright

1) Para el Login

```
//test-login.js

const { chromium } = require('playwright');  // o 'firefox' o 'webkit'



(async () => {
```

```
 const browser = await chromium.launch({ headless: false });  // `headless: false` para ver el navegador

 const page = await browser.newPage();

  // Navegar a la página de login

 await page.goto('http://localhost:3009/login');

  // Llenar el formulario de login

 await page.fill('#email', 'ief_infor@test.cl');  // Completar campo de email usando el selector #email

 await page.fill('input[type="password"]', 'ief_infor123'); // Completar el campo de la contraseña usando el tipo 'password'

  // Hacer clic en el botón de login (ajusta el selector si es necesario)

 await page.click('button[type="submit"]');  // Asegúrate de que el selector corresponda al botón de login

  // Esperar a que la página se redirija o que el login sea exitoso

 await page.waitForNavigation();  // Esto espera una redirección (si es que ocurre)

  // Verificar si la redirección fue exitosa o si aparece un mensaje

 const currentURL = page.url();

 console.log('URL actual después del login:', currentURL);



 // Opcional: Verificar si un elemento de la página indica que el login fue exitoso

 // await page.waitForSelector('.dashboard'); // Ejemplo: Espera que aparezca el dashboard después del login

  // Cerrar el navegador

 await browser.close();

})();
```



2) Para la subida de archivos

```
const { chromium } = require('playwright');  // o 'firefox' o 'webkit'



(async () => {

 const browser = await chromium.launch({ headless: false });

 const page = await browser.newPage();

  // Navegar a la página de login

 await page.goto('http://localhost:3009/login');

  // Llenar el formulario de login

 await page.fill('#email', 'ief_infor@test.cl');

 await page.fill('input[type="password"]', 'ief_infor123');

  // Hacer clic en el botón de login

 await page.click('button[type="submit"]');

  // Esperar a que la página se redirija al home

 await page.waitForNavigation();

  // Navegar al home directamente si es necesario

 await page.goto('http://localhost:3009/home');

  // Esperar que el componente de carga esté visible (ajusta el selector
según tu HTML)

 await page.waitForSelector('input[type="file"]');

  // Seleccionar el archivo desde tu sistema local (ajusta la ruta si es
necesario)

 const filePath = '/Users/laptop6138/Downloads/2024_BDA_4UTCUTS.xlsx';  //
Cambia la ruta según corresponda

  // Establecer el archivo en el campo de carga
```

```
 await page.setInputFiles('input[type="file"]', filePath);

  // Verificar que el archivo haya sido seleccionado correctamente (puedes verificar que el nombre del archivo aparezca en la página)

 const fileName = await page.innerText('.bg-navegator');

 console.log(`Archivo seleccionado: ${fileName}`);

  // Hacer clic en el botón de subir utilizando el ID del botón

 await page.click('#upload-button');  // Usando el ID del botón para hacer clic

  // Esperar alguna señal de que el archivo se ha subido (puede ser un mensaje de éxito o la desaparición del archivo seleccionado)

 await page.waitForSelector('.upload-success');  // Ajusta el selector según tu caso

  console.log('Archivo subido exitosamente.');

  // Cerrar el navegador

 await browser.close();

})();
```