

Documentación técnica proyecto: RelojReady.



UNIVERSIDAD
DE LA SERENA
CHILE

Equipo de trabajo:

Brayan Ávalos Vega - Jefe de proyecto
Benjamín Chávez Flores - Desarrollador
Valeria Milla Pizarro - Ingeniera de calidad

1-. Introducción

El presente documento está destinado a ser utilizado por los desarrolladores, administradores del sistema y otros involucrados, con el objetivo de asegurar una comprensión completa del sistema y facilitar su uso, mantenimiento y extensión.

El proyecto se desarrolló bajo el contexto de la asignatura Ingeniería de Software II de la carrera Ingeniería en Computación en la Universidad de La Serena, con el propósito de atender las necesidades de administración de personal del Hospital de Coquimbo.

1.1 Propósito del proyecto

El área de RRHH del Hospital de Coquimbo, la cual será referida a partir desde ahora como nuestro cliente, solicitó un software que sirviera de ayuda para resolver una problemática en el proceso de carga de un archivo de asistencia (reloj.log)

El software actuará como una herramienta que sirva como punto intermedio entre la descarga del archivo "reloj" y su posterior carga en el sistema de gestión de RRHH, garantizando que los datos sean correctos y consistentes antes de ser procesados.

1.1 Alcance

El alcance de este proyecto incluye:

- Desarrollo de un aplicación web que permita importar el archivo 'reloj' como también los archivos de horarios(horarios asignados y creados).
- Implementación de funcionalidades para solucionar errores comunes, como son la omisión de marcaje, marcaje múltiples y ajustes de entrada o salida.
- Desarrollo de una interfaz gráfica que permita revisar los archivos 'reloj' en forma tabular, además de tener funcionalidades de filtrado y resaltado de registros
- Exportación del archivo 'reloj' corregido y lista de empleados no registrados
- Visualización de historial de cambios

El alcance no incluye:

- No se realizará una conexión directa con el sistema de RRHH ni con otros sistemas
- La carga automática de los archivos corregidos al sistema de RRHH.
- El sistema está diseñado para ejecutarse únicamente en **localhost**
- Solo se garantiza el soporte para los formatos entregados previamente de los archivos 'reloj' y horarios
- El sistema no está diseñado para manejar múltiples usuarios simultáneos de manera óptima.

2-. Requisitos Específicos

2.1 Requerimientos funcionales sintetizados

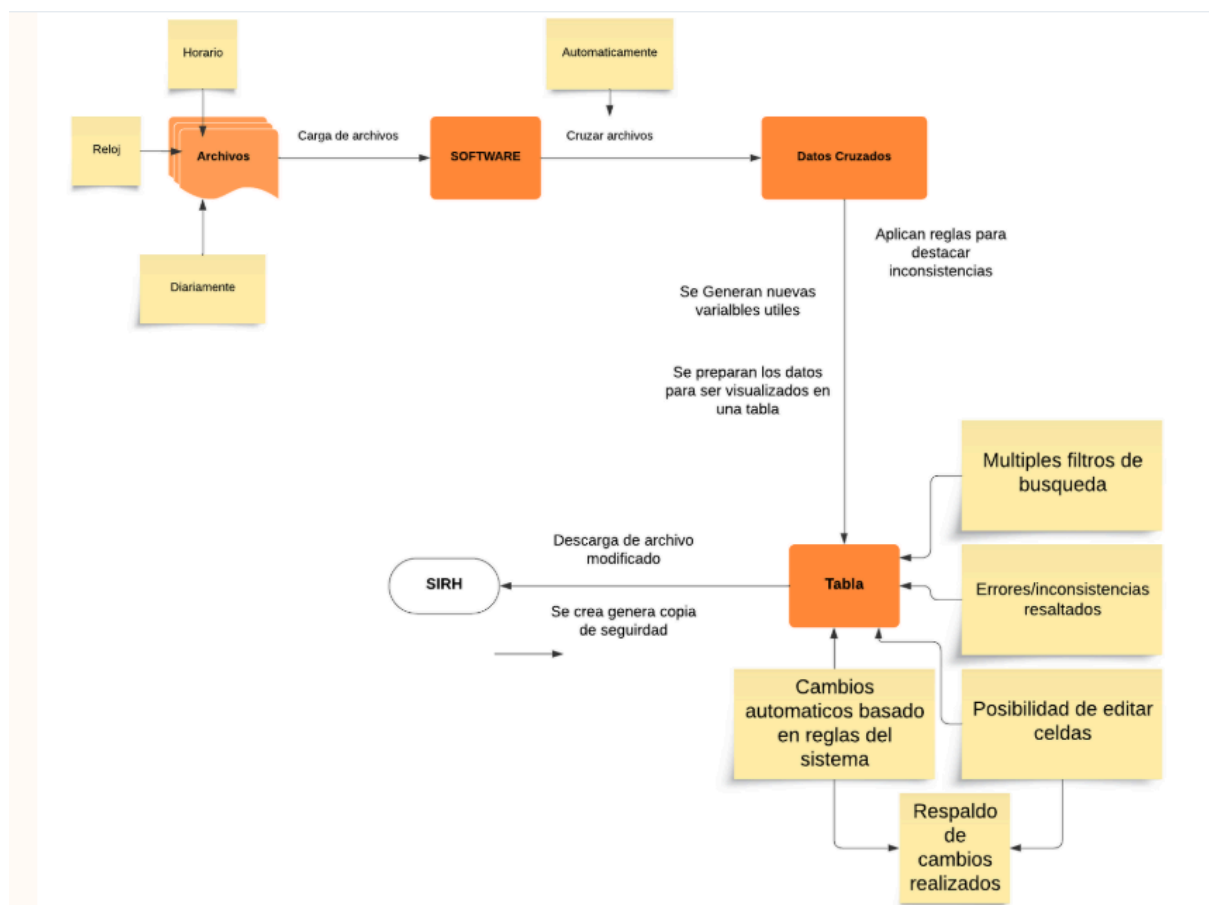
Código	Nombre	Descripción
RF01	Gestión de usuarios	Gestiona los usuarios y otorga accesos diferenciados. Existirán roles de "Usuario" y "Administrador" con diferentes funciones.
RF02	Login y seguridad	El acceso al sistema se realizará mediante un login solicitando nombre de usuario y contraseña.
RF03	Carga de archivo reloj	El sistema debe permitir cargar los archivos que contengan los registros de asistencia y horarios.
RF04	Mensaje para archivo inválido	Mostrar mensajes específicos si el archivo no cumple con el formato, indicando líneas o campos problemáticos.
RF05	Sincronización de datos	Sincronizar automáticamente los datos de los archivos reloj y horarios utilizando el RUT como identificador único.
RF06	Actualización frecuente de horarios	Permitir la carga frecuente del archivo de horarios para mantener los datos actualizados.
RF07	Visualización tabular de datos	Mostrar los datos cargados en formato tabular, incluyendo los campos relevantes.
RF08	Orden y filtrado de datos	Permitir ordenar y filtrar los datos según criterios como fecha, RUT. Los filtros deben ser combinables.
RF09	Resaltado de datos	Resaltar visualmente registros irregulares con colores específicos y un ícono de advertencia.
RF10	Confirmación de cambios	Mostrar un mensaje de confirmación al realizar cambios e indicar la modificación efectuada.
RF11	Ajustar errores de marcaje	Identificar datos duplicados basándose en RUT y fechas de asistencia.
RF12	Registro de cambios	Registrar un historial de modificaciones en un archivo de texto en formato "log", incluyendo detalles.
RF13	Exportación de archivo corregido	Permitir descargar un archivo .log corregido que pueda ser cargado manualmente al ERP (SIRH).
RF14	Verificación previa a exportación	Verificar que los datos exportados cumplan con el formato requerido por el ERP. En caso de error, mostrar mensajes de inconsistencia.

3-. Análisis (Modelado de datos)

Se realizó el diseño de esta aplicación utilizando el siguiente diagrama, el cual muestra el flujo que debe realizarse para cumplir el propósito requerido por el cliente.

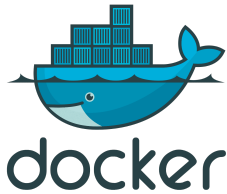
A continuación se describe el proceso:

1. **Archivos:** Los archivos se cargan en el software a partir de un horario establecido y de manera diaria.
2. **Software:** El software cruza los archivos cargados.
3. **Datos Cruzados:** Se generan nuevas variables útiles y se preparan los datos para ser visualizados en una tabla. Se aplican reglas para destacar inconsistencias.
4. **Tabla:** Los datos se presentan en una tabla con múltiples filtros de búsqueda, resaltando errores e inconsistencias. Se realizan cambios automáticos basados en reglas del sistema.
5. **Respaldo de Cambios Realizados:** Se respalda los cambios realizados.
6. **SIRH:** El archivo modificado se carga en el sistema externo SIRH.



4-. Stack tecnológico utilizado.

React (Frontend) / Flask (Backend) / MongoDB (Base de Datos) / Docker (Despliegue) / PostgreSQL (Base de Datos)



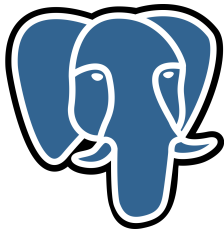
Docker es una plataforma para desarrollar, enviar y ejecutar aplicaciones dentro de **contenedores**. Este proyecto utiliza Docker para garantizar que el **entorno sea consistente entre el desarrollo y la producción**, simplificando el proceso de **despliegue y asegurando la portabilidad**.



React es una biblioteca de JavaScript para construir **interfaces de usuario interactivas y dinámicas**. En este proyecto, se utiliza para desarrollar el frontend, permitiendo una experiencia de usuario fluida



Flask es un framework ligero de Python para **crear aplicaciones web**. En este proyecto, se utiliza para implementar el backend como un **microservicio**, facilitando la lógica del servidor, las validaciones de datos y la comunicación con la base de datos.



PostgreSQL es un **motor de base de datos relacionales (RDBMS)** el cual será encargado de **almacenar los usuarios registrados en el sistema**, para que estos al momento de realizar el login deban autenticarse con las credenciales almacenadas en la base de datos

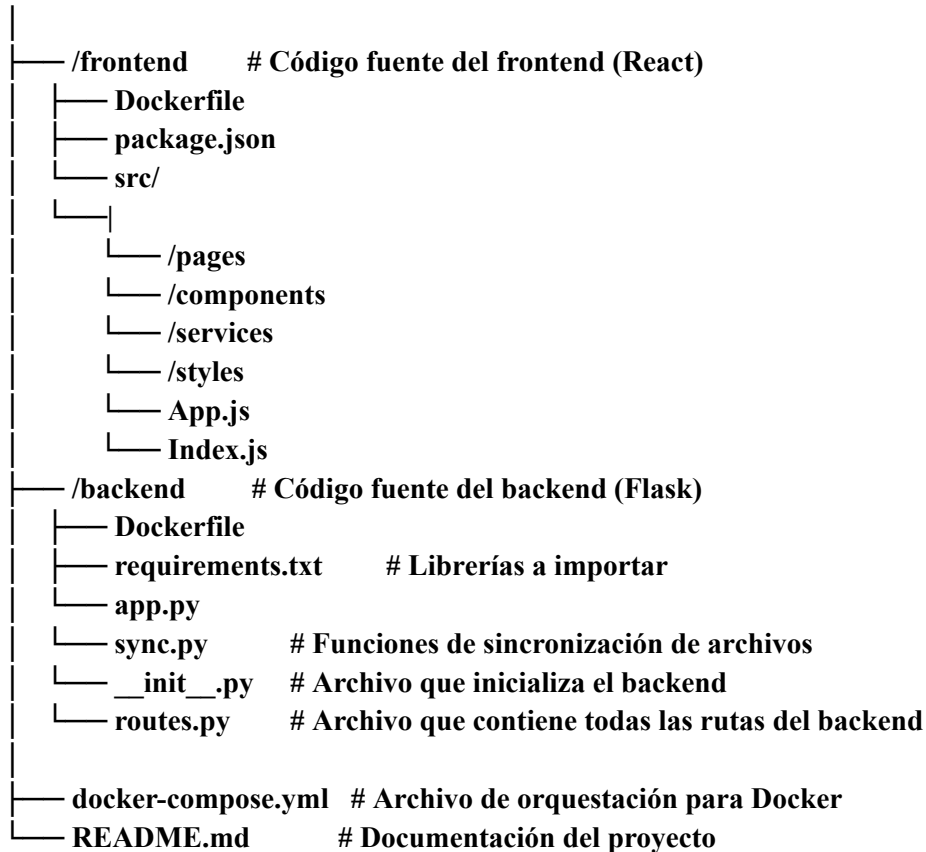


MongoDB es una base de datos que almacena datos en formato de **documentos JSON**, ofreciendo **flexibilidad y escalabilidad**. En este proyecto, se utiliza como la base de datos principal para manejar y almacenar los **registros del reloj** y nuestras nuevas adiciones.

4.1 Diseño de la arquitectura

Preliminarmente diseñamos la arquitectura del proyecto que seguimos usando durante el desarrollo del proyecto, estaba basado en una estructura de 2 microservicios (frontend y backend) que posteriormente se despliegan simultáneamente con docker, dentro del proyecto se encontrará disponible la documentación.

/Gestion-hcoquimbo



4.1 Módulos del sistema

Gestion de usuarios

Este sistema se encarga de las funciones relacionadas con la autenticación de los usuarios, para ello tener un control de quien puede realizar cambios en el sistema y dejar registro de ello.

Para este sistema se utilizó **postgresql** para almacenar los datos, se utilizó **Flask Login** para la gestión de registros de usuarios y **Werkzeug Security** para almacenar contraseñas de forma segura y verificarlas al iniciar sesión.

En el sistema se diseñaron 2 tipos de usuarios, el **administrador** y el **usuario**.

Administrador

Se encarga de **crear y gestionar a los usuarios** las cuentas de los usuarios del sistema además de tener **acceso completo a todas las funciones del sistema**.

Inicio Carga reloj Carga horario Visualizar datos Historial de cambios

Panel de gestión de usuarios del sistema.

RUT	Nombre	Apellido paterno	Apellido materno	Rol	Acciones	
12345678-9	Admin	AdminP	AdminM	administrador	Editar	Eliminar
98765432-1	Valeria	Milla	Pizarro	usuario	Editar	Eliminar
111111111-1	Benjamin	Chávez	Flores	usuario	Editar	Eliminar
12345	Brayan	Vega	Avalos	administrador	Editar	Eliminar

Crear Nuevo Usuario

Usuario

Tiene acceso a subir los archivos de reloj y horarios, puede modificar la tabla de forma manual o con las funciones automatizadas, puede exportar un registro de reloj y ver los logs de los cambios realizados.

Inicio Carga reloj Carga horario Visualizar datos

Carga de archivos

Este sistema se encarga de crear las funciones y los apartados para que se puedan subir los archivos necesarios para el funcionamiento del sistema (Archivo Reloj.log, Horarios Creados.csv y Horarios asignados.csv).

Para ese sistema se utilizaron **carpetas para guardar de forma local y temporal los archivos** subidos, en este caso se encuentran 4 tipos de carpeta, 2 que guardan de forma individual los archivos relacionados para el horario, una que guarda el último registro del reloj, y otra que guarda todos los diferentes archivos del reloj, esto se utiliza como **mecanismo para almacenar los archivos previo a la sincronización** y posterior guardado en **MongoDB**.

Para la subida de archivos se verifica en el caso del reloj que se contengan la misma cantidad de filas que el formato original y el tipo de archivo “.log” y en el caso de los horarios al tener un formato poco definido solamente se verifica que esté subido un archivo “.csv”.

Carga Reloj



Arrastra y suelta el archivo aquí, o haz clic para seleccionar uno

Confirmar subida

Regresar al Home

Carga Horarios



Arrastra y suelta el archivo horarios asignados aquí, o haz clic para seleccionar uno



Arrastra y suelta el archivo horarios creados aquí, o haz clic para seleccionar uno

Confirmar subida

Regresar al Home

Sincronización de archivos

Este sistema es únicamente interno, se utiliza para que al momento de confirmar que se suba un archivo además de guardarse localmente en las carpetas estos se crucen con los archivos del horario, con el fin de obtener los horarios que tiene la gente asignada.

Para ello se utilizó pandas como principal tecnológica para crear dataframes que se puedan cruzarse de forma más sencilla y segura, además de ello en **def obtener_df():**, que es la función que engloba todas las funciones necesarias para cruzar estos archivos, se agregó la primera interacción con **MongoDB**, al momento de cruzar estos datos se guardan en la base de datos como la colección “registros”.

El proceso de cruce de archivos consiste en:

1. Los archivos de horarios se filtran y organizan, creando un DataFrame con días de la semana, horas de entrada y salida, y códigos de horario.
2. Los horarios procesados se cruzan con el archivo horario asignado utilizando el código de horario, generando un DataFrame combinado.

3. El DataFrame combinado se cruza con los registros de reloj, transformando y mapeando las fechas a días de la semana, y alineando los datos mediante el RUT de la persona.

De esta forma obtenemos del cruce un dataframe con los datos relevantes para los siguientes modelos, asegurando que los datos de los diferentes archivos se integren eficientemente.

Tabla

Este sistema es núcleo del proyecto, donde se pueden visualizar el archivo del reloj con nueva información relevante, destacando los errores o posibles errores registrados en el sistema.

Como este apartado es netamente frontend la principal tecnología a utilizar fue **React**, la librería **ag-grid-react** y **mongoDB** como la fuente de los datos.

AG Grid React es una biblioteca que proporciona tablas avanzadas, altamente personalizables y con gran rendimiento para proyectos en React.

En esencia en la tabla se muestran todos los datos recolectados en **mongoDB** y se destacan con color las columnas que tengan problemas detectados, según el tipo de error registrados se marcan de uno u otro color.

row-red': params => params.data['Error encontrado'] === 'Omisión de Marcaje'

JUEVES	20-m	1	07/11/24	06:11	08:00	13:00	
JUEVES	20-m	1	07/11/24	06:11	08:00	13:00	Marcaje múltiple

Además la tabla tiene filtros para poder buscar datos de forma más específica.

En esta sección también se encuentran los botones para realizar **cambios automáticos** según el **error registrado**, y los botones que corresponden a **exportar el archivo del reloj** y también un archivo con las **personas que no tengan un horario asignado**.

Omisión de Marcaje

Este botón se encarga de revisar en la base de datos las personas que tengan un la “**hora_reloj**” = “**00:00**” que corresponde a la **hora automática registrada por sistema** del reloj biótico (Queda según la hora que marca el reloj y el día que se registra, si es 00:00, corresponde al día siguiente? o al correspondiente al del día del marcaje de la entrada, este es un aspecto a mejorar),

Permite que en aquellas filas donde la entrada no es marcada se le asigne el horario programado.

Omisión de Marcaje

Marcaje múltiple

Este botón se encarga de revisar la diferencia de tiempo que puede tener un registro de entrada o salida de otro similar, en el caso de que una persona **haya marcado más de una vez el mismo registro**, en caso de que la diferencia de tiempo sea igual o menor a un parámetro establecido (3 minutos de forma default) se eliminarán de los registros de la tabla, no así de la base de datos.

Encuentra filas en donde se ha marcado más de 1 vez, en un rango de tiempo de 5 minutos.

Marcaje múltiple

Error de entrada/salida

Este botón se encarga de corregir que no se hayan marcado consecutivamente más de una entrada o más de una salida de una misma persona, es decir, que en el registro se debe **verificar que cada entrada de una persona esté asociada con una salida**, por lo que en la modificación debería hacer que en los registros se muestran los datos como corresponde en la realidad.

Revisa que no existan dos entradas o dos salidas consecutivas para la misma persona.

Ajustar entrada/salida

Además los 3 botones anteriores contienen un modal que al presionarlos se **visualizan los datos que se van a cambiar y/o eliminar** de los registros.

Confirmar Eliminación

Las siguientes filas serán eliminadas:

- Día: JUEVES, RUT: 20-m, hora_reloj: 06:11
- Día: JUEVES, RUT: 20-m, hora_reloj: 06:24
- Día: JUEVES, RUT: 20-m, hora_reloj: 06:25
- Día: JUEVES, RUT: 20-m, hora_reloj: 06:31
- Día: JUEVES, RUT: 20-m, hora_reloj: 06:35

Confirmar Cancelar

Cambios de reglas del sistema

Se pueden **modificar los parámetros predefinidos** de los botones relacionados con errores del sistema desde un un botón, al presionar el botón se despliega un modal con diferentes inputs correspondientes a cada tipo de modificación, y se pueden cambiar los datos al ingresar, **este dato queda guardado de forma local** por lo que se mantiene permanentemente el dato.

Configurar Valores

Minutos en Marcaje múltiple:

33

Horas en Ajustar entrada/salida Incorrecta:

1

Guardar

Configuración

Historial de cambios


En este apartado se muestran los **registros de los cambios realizados**, donde se especifica la **fecha**, el **usuario** que realizó dicha modificación, y se muestran los **datos importantes de la fila original** que se modificó.


Usuario	Descripción del Cambio	Fecha del Cambio
admin	Se realizó la modificación de 1 en "RUT: 20-m, Entrada/Salida: 1, Hora Entrada: 08:00, Hora Salida: 13:00, Hora Reloj: 06:12, Fecha Reloj: 06/11/24"	2024-12-11 14:12:14
admin	Se realizó la modificación de 1 en "RUT: 20-m, Entrada/Salida: 1, Hora Entrada: 08:00, Hora Salida: 13:00, Hora Reloj: 00:10, Fecha Reloj: 07/11/24"	2024-12-11 14:12:14
admin	Se realizó la modificación de 1 en "RUT: 20-m, Entrada/Salida: 1, Hora Entrada: 08:00, Hora Salida: 13:00, Hora Reloj: 06:13, Fecha Reloj: 07/11/24"	2024-12-11 14:12:14

Exportación de archivo reloj

Con el botón de exportar se abrirá un modal en el cual se pueden seleccionar o escribir 2 **rangos de fechas**, el de inicio y fin, al seleccionarlo y confirmar se envía una solicitud que recorre todos los los datos de la base de datos que no han sido “eliminados” entre ese rango de fecha y **descarga un archivo.log** con el formato correspondiente a los registros del reloj, para poder ser subido posteriormente a su RRHH.

Seleccionar Rango de Fechas

Fecha de Inicio: 

Fecha de Fin: 

Aplicar

Cerrar

Exportar

Exportación de archivo trabajadores sin horario asignado

Esta función permitirá que se puedan extraer los RUT de los usuarios que no posean un horario asignado en los archivos de horarios asignados y horarios creados.

No registrados

5-. Pruebas

Proceso	Detalles
Verificar selección archivo más reciente	Entradas: Archivos .csv y/o .log.
	Resultados esperados: El archivo más reciente es seleccionado, si el archivo no cargó muestra error.
Asegurar que los horarios se procesen correctamente	Entradas: Archivo csv horarios.
	Resultados esperados: Se genera un DataFrame con los días de la semana y las horas en formato 'HH:MM'.
Cruce correcto de los horarios	Entradas: DataFrame de horarios y archivo .csv de horarios asignados.
	Resultados esperados: DataFrame resultante con las columnas combinadas correctamente.
Integración de datos del reloj con horarios y asignaciones	Entradas: DataFrame cruzado de horarios y horarios asignados, y archivo .log de registros de reloj.
	Resultados esperados: DataFrame final contiene las columnas adecuadas y en caso de no existir coincidencia se completa con 'No Registro'.
Verificar la conexión e inserción correcta de datos en MongoDB	Entradas: Archivo JSON proveniente del Dataframe final.
	Resultados esperados: Los documentos se insertan correctamente en la base de datos, se agregan los campos "Modificación" y "Estado".

Proceso	Detalles
Obtener registros con Omisión de marcaje	Entradas: Colección MongoDB con registros que contienen Modificación: 2 y Estado: 'Vivo'.
	Se obtienen los registros con estas condiciones.
Cambio de entrada/salida y registro de cambios	Entradas: Registro con Modificación: 3
	Se cambia entrada/salida (de 1 a 3 o de 3 a 1) y se registra el cambio en registro_cambios.
Eliminar duplicados y registrar cambios	Entradas: Registros con el mismo RUT, entrada/salida, y fecha_reloj con una diferencia de tiempo pequeña.
	Los duplicados se marcan como 'Muerto' y se registran los cambios en registro_cambios.

6-. Posibles mejoras del sistema

- **Mejora de protección contra errores humanos del sistema:**
Esto es necesario para que el sistema sea tolerante a fallos de origen humano como el subir archivos en un orden incorrecto por ejemplo
- **Implementar nuevas reglas al sistema si es que existe otro caso particular que demore el trabajo de los funcionarios.**
Si se llegasen a considerar nuevas reglas de negocio de parte del cliente, estas deben ser incorporadas e integradas al sistema.
- **Mejorar la interfaz gráfica.**
Se debe considerar una mejora en la interfaz gráfica de modo que se ofrezca al usuario final una mejor experiencia de uso de la aplicación.
- **Añadir funcionalidades extras, especialmente enfocadas con el filtrado de la tabla y/o archivos referido a fecha.**
En caso de que se requiera, se deben implementar nuevas características referentes al funcionamiento de la tabla en la que se visualizan los datos.
- **Implementar lo necesario para un sistema de co-working fluido y estable.**
El cliente manifestó la posibilidad de incorporar un sistema de coworking. Por lo que se deben investigar alternativas para llevar a cabo esta característica.
- **Permitir que los cambios realizados puedan revertirse**
En caso de que el usuario encargado de modificar el archivo cometa un error, o bien, considere que el resultado de la modificación no fue el esperado, el sistema debe ser capaz de deshacer los cambios que fueron hechos volviendo al estado anterior a la modificación.
- **Añadir funciones de administrador respecto a la base de datos dentro del sistema.**
A modo de que el sistema no almacene demasiados registros que no serán útiles al momento de realizar la exportación de datos, el administrador tendrá la facultad de realizar tareas de mantenimiento, esta se tiene previsto que sea el borrar registros desde la base de datos de MongoDB.