

Trabajo Práctico N° 4

Rettore Ricardo

Legajo: VINFo4566



Junio de 2024

Seminario de Práctica de Informática

Docente: Hugo Fernando Frías

Módulo 4

Contenido

Título	4
Introducción.....	4
Justificación.	4
Definiciones del proyecto	4
Objetivo general del proyecto.....	4
Objetivos particulares:	5
Definiciones del sistema.....	5
Objetivo general del sistema	5
Elicitación.....	5
Conocimiento del negocio.....	9
Diagrama de Dominio	10
Propuesta de Solución.....	10
Propuesta funcional.....	10
Propuesta técnica.....	11
Diagrama de Arquitectura	11
Arquitectura de comunicaciones.....	11
Requerimientos.	13
Requerimientos Funcionales.....	13
Requerimientos No Funcionales.....	14
Requerimientos Futuros.....	14
Inicio del Análisis	15
Casos de Uso	15
Identificación de Actores.....	15
Trazabilidad	16
Definición de los Casos de Uso.....	17
Etapas de Análisis	20
Diagrama Secuencia CU07 – Gestionar Mantenimiento.....	20
Diagrama Secuencia CU08 – Gestionar Cuadrilla.....	21
Diagrama Secuencia CU09 – Gestionar Materiales	21
Diagrama Secuencia CU11 – Programar Mantenimiento.....	22
Diagrama Secuencia CU12 – Gestionar Fin Mantenimiento.....	22
Etapas de Diseño.....	23
Diagrama de Clases	23

Etapa de Implementación	24
Diagrama de Despliegue.....	24
Etapa de Pruebas	25
Plan de Prueba	25
Tratamiento de Defectos	29
Interfaz Gráfica.....	32
Definición de la base de datos para el prototipo	32
Detalle de Tablas	33
Manipulación de Tablas	36
Consulta de datos	38
Borrado de tablas.....	39
Desarrollo del Sistema	40
Análisis de las clases.....	41
Conexión con la Base de Datos MySQL	47

Título

Desarrollo del Sistema de gestión de mantenimiento de redes eléctricas de DistriSpark, impulsando la productividad.

Introducción.

La empresa DistriSpark es una empresa destinada a la distribución de energía eléctrica en la zona norte de la provincia de Entre Ríos, con el fuerte compromiso de optimizar sus procesos para contribuir adecuadamente al crecimiento productivo de la zona de incumbencia.

La empresa realiza mantenimientos en las líneas eléctricas y requiere de un software que le permita registrar y programar adecuadamente dichas tareas, con el fin de optimizar los tiempos y materiales utilizados, en virtud de, con el tiempo, lograr una base de conocimiento que le permita prever a largo plazo las inversiones necesarias para mantener en óptimas condiciones las redes eléctricas y contribuir al sector productivo y residencial con un servicio de calidad.

Justificación.

La red eléctrica que debe mantener y ampliar la empresa DistriSpark se extiende a lo largo de la zona norte de la provincia de Entre Ríos, contando con un total de 650 km de red de media tensión, de los cuales un 30% se encuentra en zonas urbanas y el restante 70% se extiende a lo largo de las vastas zonas rurales que posee el área de concesión.

Las tareas de mantenimiento son fundamentales para mantener el normal funcionamiento de las instalaciones. Una correcta gestión y registración de estas permite mejorar la gestión de tiempos de las cuadrillas, y la registración de los materiales asociados a los mantenimientos, para que, su posterior valorización derive en la determinación de los costos asociados.

Por otro lado, en Entre Ríos, el servicio de distribución de energía eléctrica es un servicio público esencial a cargo del estado provincial y cedido a distintas empresas destinadas a realizar dicha tarea bajo una estricta reglamentación.

Cada cinco años, cada distribuidora debe realizar un estudio económico de los costos incurridos en el mantenimiento y crecimiento de las redes por incremento de la demanda. Del estudio presentado ante el ente regulador, el mismo, otorga a la distribuidora un incremento en el valor asociado a la distribución, permitiendo a la distribuidora, previa audiencia pública, incorporar a la facturación de los usuarios el costo asociado al funcionamiento de la empresa.

En este sentido, el conocimiento del costo de mantenimiento y su proyección es de vital importancia para poder evaluar adecuadamente el costo de funcionamiento de la empresa y permitir su crecimiento.

Definiciones del proyecto

Objetivo general del proyecto

El sistema de gestión de mantenimiento permitirá optimizar la gestión de los trabajos de mantenimiento y consecuentemente valorizar las tareas permitiendo una proyección de costos a lo largo del tiempo, estimando las erogaciones necesarias para el correcto funcionamiento de las instalaciones.

Objetivos particulares:

- Planificar las tareas de las cuadrillas.
- Priorizar los trabajos de mantenimiento en base a una criticidad.
- Contabilizar los materiales utilizados en los mantenimientos ejecutados para facilitar la confección del estudio económico solicitado por el ente regulador.

Definiciones del sistema**Objetivo general del sistema**

El sistema deberá permitir la registración de los mantenimientos para su posterior planificación, afectar a una cuadrilla en particular y asociar los materiales necesarios para la realización de la tarea, permitiendo finalmente detallar si un trabajo fue ejecutado o, si por alguna razón, no fue realizado y debe ser reprogramado.

Alcance:

- Registrar el ciclo de vida de los trabajos de mantenimiento.
- Incorporar una prioridad a cada trabajo, optimizando la planificación.
- Permitir contar con el stock de materiales en stock para mantenimiento.
- Registrar los materiales asociados a los trabajos de mantenimiento.
- Registrar las cuadrillas que realizan cada mantenimiento.
- Registrar horas de traslado estimadas para arribar a la zona de trabajo.
- Registrar horas de trabajo estimada para realizar la tarea.
- Emitir reporte de los trabajos realizados y materiales utilizados.
- Imprimir los trabajos diarios a realizar.

Queda fuera de alcance del sistema:

- La confección de la interfaz con el sistema de almacenes, debiendo ingresar el listado de materiales de forma manual.
- No será contemplada la registración de los trabajos que surjan de la realización de reparaciones por fallas sucedidas en las instalaciones.
- El sistema no realizará la valoración de los materiales utilizados en los trabajos de mantenimiento, sólo se limitará a la contabilización de estos.
- No se contempla la confección de indicadores de gestión.

Elicitación.

Para entender el dominio del problema se realizó una entrevista con el jefe de Mantenimiento para entender en forma general el proceso de mantenimiento y conocer sus actores. Como resultado de la entrevista se documentó la minuta de la reunión.

Posteriormente se concertaron entrevistas con cada actor dentro del proceso de gestión de mantenimiento, en función de determinar las actividades específicas que cada uno de ellos realiza en el proceso. En cada entrevista se confeccionó en conjunto con el entrevistado un diagrama de bloques identificando la fuente de información con la que se comunica para poder realizar su actividad, el proceso que realiza con dicha información y a quién reporta el resultado de su actividad. Para ello se utilizó la herramienta Miró para la confección del proceso que ejecuta cada uno de los intervinientes.

Documentación Generada:

Documento	Descripción
MR001 - Entrevista jefe Mantenimiento	Identificación de los roles que participan del proceso.
RE001 - Jefe Mantenimiento	Relevamiento de tareas que ejecuta el jefe de Mantenimiento
RE002 - Analista Mantenimiento	Relevamiento de tareas que ejecuta el Analista de Mantenimiento
RE003 - Responsables Cuadrillas	Relevamiento de tareas que ejecutan las cuadrillas
Nota: MR (Minuta Reunión) - RE (Relevamiento)	

Tabla 1 - Activos de Elicitación

A continuación, se detalla el contenido de cada documento generado, de forma concisa y acotada, como resultado de cada interacción con los agentes involucrados. Estos documentos se envían por correo para la validación final de cada participante de la reunión.

MR001 - Entrevista jefe Mantenimiento

Personas que participan de los trabajos de mantenimiento:

- Cuadrilla
- Administrador de Mantenimiento
- Jefe de Mantenimiento

Necesidades detectadas:

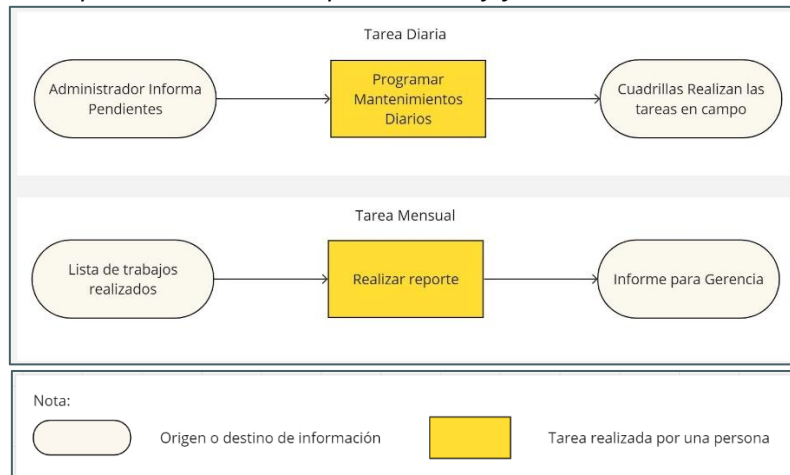
- Registrar los mantenimientos: los mantenimientos se llevan en un cuaderno, pero organizar la tarea y controlar la realización de las mismas es compleja e inadecuada.
- Priorizar los mantenimientos: los mantenimientos no se priorizan, dado que es complejo controlar todo lo pendiente.
- Ordenar el trabajo diario de las cuadrillas: a las cuadrillas se les asigna una serie de trabajos, pero requieren de alguien que les transmita cada uno de ellos.
- Agilizar extracción de reportes: Es compleja la realización de reportes gerenciales.

RE001 - Relevamiento Jefe Mantenimiento

- Rol: jefe mantenimiento
- Tarea Diaria: Especificar a cada contratista el trabajo que debe realizar durante el día laboral.
- Tarea Mensual: Mensualmente se debe entregar a la gerencia el reporte de los trabajos realizados y materiales utilizados.

Necesidades detectadas:

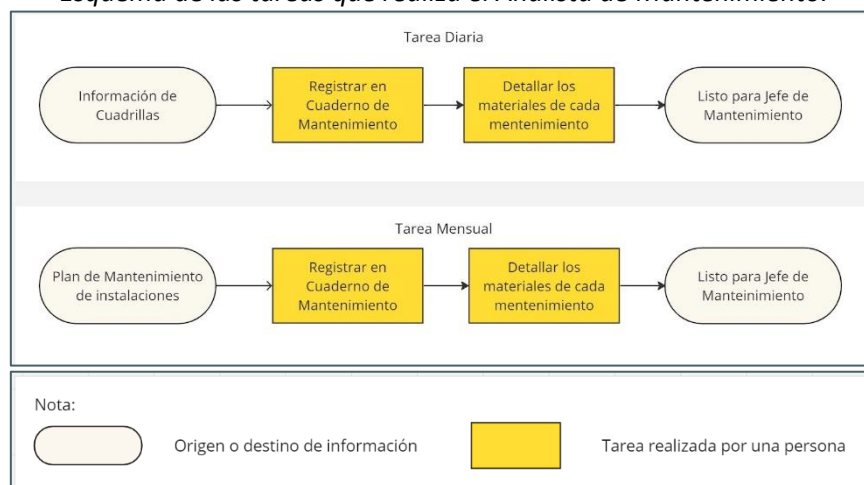
- Poder separar fácilmente los mantenimientos pendientes, los reprogramados y los cerrados para facilitar la organización y priorizar los trabajos reprogramados por sobre los que se encuentran pendientes.

Esquema de las tareas que realiza el jefe de Mantenimiento.*Figura 1 - Acciones del jefe de Mantenimiento***RE002 - Relevamiento Analista de Mantenimiento**

- Rol: Analista de Mantenimiento
- Tarea Diaria:
 - Escribir en el cuaderno de mantenimiento lo informado verbalmente por las cuadrillas sobre necesidades detectadas en campo.
 - Agregar a cada mantenimiento la lista de materiales necesarios para su realización.
- Tarea Mensual: Cargar al cuaderno de mantenimiento los trabajos necesarios según el plan de mantenimiento periódico de las instalaciones eléctricas.

Necesidades detectadas:

- Registrar los mantenimientos: dejar de utilizar cuadernos para cargar los mantenimientos a realizar.
- Poder tener identificados de alguna manera los trabajos correspondientes al plan de mantenimiento de las instalaciones de los mantenimientos detectados por inspección en campo de las cuadrillas.

Esquema de las tareas que realiza el Analista de Mantenimiento.*Figura 2 - Acciones del Analista de Mantenimiento*

RE003 - Relevamiento Responsables de Cuadrillas

- Rol: Responsable de cuadrilla
- Tarea Diaria:
 - Copiar del cuaderno los mantenimientos y materiales a realizar para las tareas del día.
 - Reportar los mantenimientos que se pudieron realizar y cuales no, indicando dicha tarea en los cuadernos de mantenimiento.
 - Informar al Analista de mantenimiento el resultado de la inspección visual del día anterior.

Necesidades detectadas:

- Poder imprimir la lista de mantenimiento y materiales para realizar el trabajo.
- Poder simplificar la búsqueda en el cuaderno para registrar el resultado de los mantenimientos.

Esquema de las tareas que realiza el responsable de Cuadrillas.

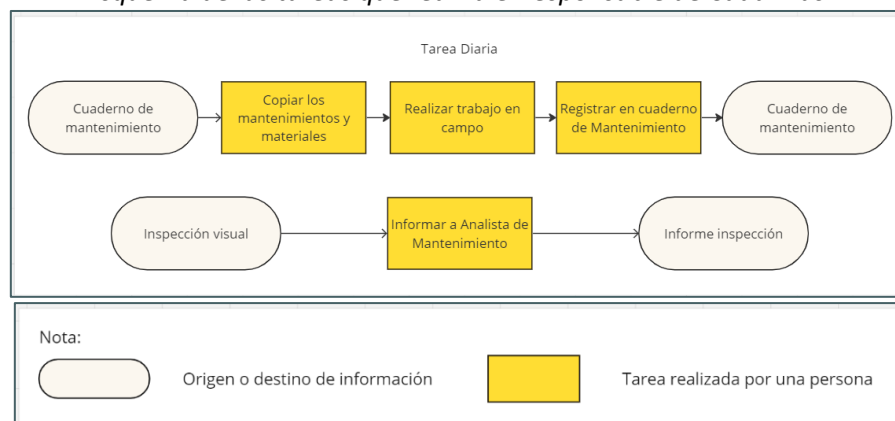


Figura 3 - Acciones del responsable de cuadrilla

Conocimiento del negocio.

Para mejorar el conocimiento del negocio y formalizar un proceso acordado entre los involucrados, se realizó el relevamiento final de las tareas e interacciones que existen entre los participantes del proceso.

Diagrama del proceso.

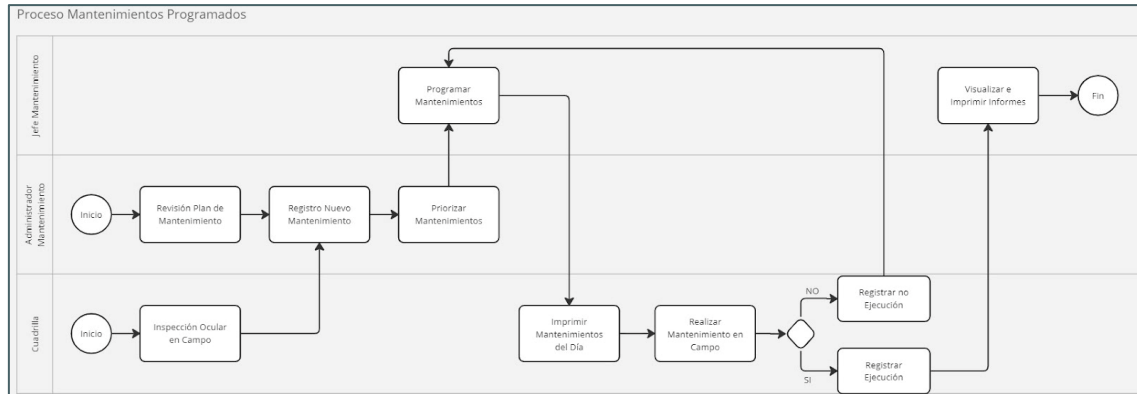


Figura 4 - Diagrama proceso Mantenimiento de Redes

Detalle del proceso.

- El Administrador de Mantenimiento verifica periódicamente el plan de mantenimiento de las instalaciones que así lo requieran y lo registrará en el sistema. Por otro lado, aprovechando que las cuadrillas recorren continuamente las líneas e instalaciones, van realizando una inspección ocular de las mismas, y, si se detecta una necesidad de mantenimiento, le informan al Administrador de Mantenimiento para que dicha tarea sea registrada también en el sistema.
- Diariamente, el Administrador de Mantenimiento prioriza los mantenimientos pendientes en función de la criticidad del problema, teniendo en cuenta la relevancia de la instalación y el impacto de una posible falla en el funcionamiento.
- El jefe de Mantenimiento revisa los mantenimientos registrados, y en función de la priorización de estos y de la disponibilidad de cuadrillas, planifica los mantenimientos que realizará cada cuadrilla, indicando el día en que se debe ejecutar la tarea. Además, en virtud de poder optimizar los trabajos a realizar cada día, registra el tiempo de traslado a cada punto donde se deben realizar los mantenimientos. Esto permite no programar trabajos que no será posible realizar por falta de tiempo.
- Las cuadrillas, diariamente, acceden al sistema y realizan una impresión del detalle de cada mantenimiento, incluyendo la lista de materiales que será necesario acarrear para realizar todos los mantenimientos asignados.
- Dependiendo de las circunstancias de carga momentánea de las instalaciones, la posibilidad efectiva de poder acceder a las instalaciones por cuestiones de tránsito o climáticas, puede suceder que un trabajo no sea factible de realizar, y otros serán realizados con normalidad.
- Finalizada la jornada laboral, cada cuadrilla vuelve a las oficinas de la empresa y el jefe de cuadrilla registrará cuales fueron los trabajos realizados y cuáles no fueron posible de realizar, indicando una justificación en este último caso.

- Todos los trabajos que no fueron realizados en campo serán devueltos al jefe de Mantenimiento para que re programe una fecha de realización.
- A medida que se requiere, el jefe de Mantenimiento visualizará y ejecutará reportes de los Mantenimientos ejecutados en campo y los materiales utilizados.

Diagrama de Dominio

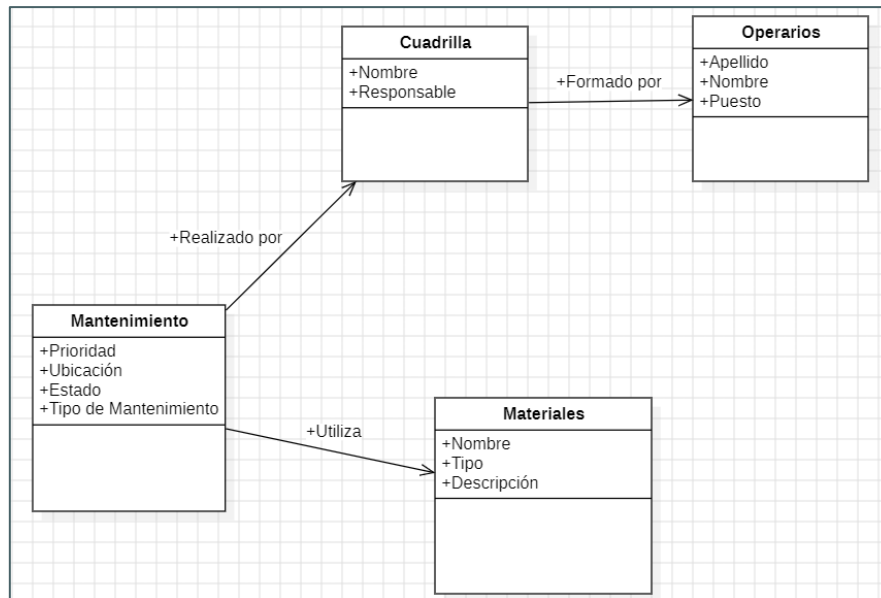


Figura 5 - Diagrama de Dominio de la Solución

Propuesta de Solución

En función del relevamiento realizado, se propone a continuación una solución para cubrir las necesidades detectadas.

Propuesta funcional

Se propone la realización de una aplicación web que permita al Analista de Mantenimiento cargar un registro por cada mantenimiento identificando para ello ciertos datos referidos al lugar y tarea a realizar, sumado a una prioridad y tipo de mantenimiento para distinguir a los mantenimientos que surgen del plan de mantenimiento anual de las instalaciones y del que, por inspección en campo de las cuadrillas, son relevados para realizar. Para cada mantenimiento cargado, el sistema permitirá asociar el grupo de materiales necesarios para realizar la tarea en campo.

Posteriormente, el sistema permitirá al jefe de Mantenimiento programar las fechas de cada uno, asignando la cuadrilla que realizará el mantenimiento, agregando además el tiempo de traslado que estima para arribar al sector de trabajo.

El jefe de cada cuadrilla podrá ingresar al sistema e imprimir los trabajos de mantenimiento y materiales asociados. Finalizado el trabajo en campo, el mismo jefe podrá indicar el estado final

del mantenimiento, especificando si el mismo fue finalizado o si, por el contrario, el trabajo debe ser reprogramado.

Propuesta técnica.

El software de Gestión de Mantenimiento será realizado en Java, permitiendo así que el mismo sea escalable.

Para la persistencia de los datos se implementará en MySQL, base de datos relacional con sobradas prestaciones de rendimiento para la aplicación a implementar y su posterior escalabilidad.

Para conectar a la aplicación con la base de datos, se implementará JDBC que es una interfaz de programación de aplicaciones estándar.

Diagrama de Arquitectura

La solución estará implementada en una arquitectura de 3 capas.

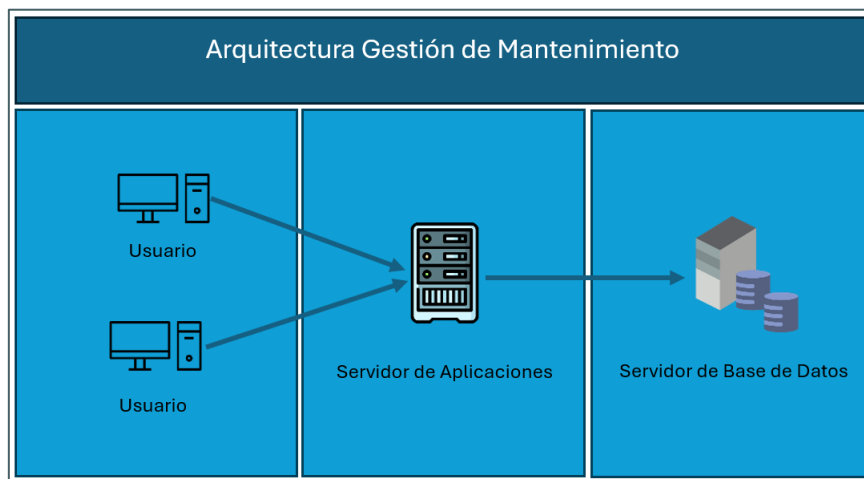


Figura 6 - Diagrama de Arquitectura de la solución.

Arquitectura de comunicaciones.

La empresa donde será implementada la aplicación cuenta con:

- Una oficina central donde se encuentran los sectores de Administración y Tecnología Informática
- Un galpón principal donde se encuentra la Oficina Central de Mantenimiento, junto con la oficina de una de las cuadrillas de mantenimiento y un almacén central de materiales.
- Un galpón secundario con oficina para cuadrillas y almacén secundario de materiales.

Dada la infraestructura edilicia de la empresa se propone el siguiente esquema de conexión entre las distintas oficinas donde se requiere acceso al sistema.

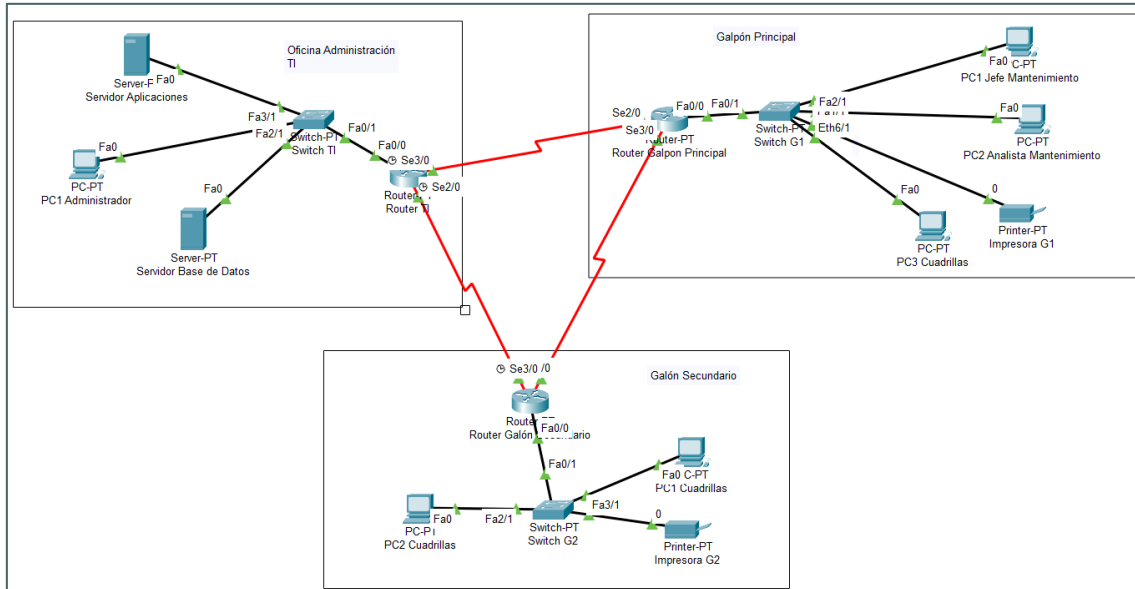
Diagrama de Arquitectura de Comunicaciones.

Figura 7 - Diagrama de arquitectura de comunicaciones de la solución.

- Cada locación se conectará mediante la utilización del proveedor de servicio de internet, solicitando que configure para la empresa una WAN privada para la interconexión entre los sitios.
- Dentro de cada locación se tenderá una red LAN para conectar los servidores, los ordenadores de usuarios e impresoras que formarán parte de la arquitectura de solución propuesta.

Requerimientos.

A continuación, se listarán todos los requerimientos funcionales (RF), no funcionales (RNF) y los requerimientos futuros (RFU) que serán incluidos en el presente documento, pero no serán tenidos en cuenta para esta implementación, quedando pendientes para futuras implementaciones.

Requerimientos Funcionales.

ID	Descripción
RF01	El sistema debe permitir ingresar a la aplicación mediante una validación de credenciales (usuario y contraseña).
RF02	El sistema limitar las acciones que cada operador del sistema puede realizar, dependiendo de su participación dentro del proceso de negocio.
RF03	El sistema debe permitir cargar materiales a una lista de materiales susceptibles de ser utilizados en los mantenimientos.
RF04	El sistema debe permitir cargar operarios de campo que realizan los mantenimientos, indicando su jerarquía (responsable, oficial, ayudante).
RF05	El sistema debe permitir eliminar un operario del sistema.
RF06	El sistema debe permitir crear cuadrillas.
RF07	El sistema debe permitir asignar operarios a las cuadrillas.
RF08	El sistema debe permitir eliminar operarios de las cuadrillas.
RF09	El sistema debe permitir registrar un nuevo mantenimiento en el sistema con estado "Pendiente".
RF10	El sistema debe permitir especificar si el mantenimiento corresponde al plan anual de mantenimiento o al resultado de la inspección en campo.
RF11	El sistema debe permitir asignarle una prioridad al trabajo de mantenimiento dependiendo de la urgencia en la realización del mantenimiento.
RF12	El sistema debe poder cambiar la prioridad asignada al trabajo.
RF13	El sistema debe poder borrar un mantenimiento cargado.
RF14	El sistema debe permitir modificar los datos de un mantenimiento cargado.
RF15	El sistema debe permitir Asignarle una cuadrilla al mantenimiento.
RF16	El sistema debe poder ingresar un tiempo de traslado al mantenimiento, tiempo que le insume a la cuadrilla arribar al lugar de trabajo.
RF17	El sistema debe permitir cambiar el tiempo de traslado asignado.
RF18	El sistema debe permitir agregar materiales a un mantenimiento registrado.
RF19	El sistema debe permitir ingresar una fecha de realización del trabajo cambiando el estado del mantenimiento a "Programado".
RF20	El sistema debe poder cambiar una fecha de realización del trabajo.
RF21	El sistema debe poder cambiar el estado del trabajo de mantenimiento a "Realizado" cuando el trabajo fue realizado.
RF22	El sistema debe poder cambiar el estado del trabajo de mantenimiento a "Reprogramar" cuando el trabajo no pudo ser realizado.
RF23	El sistema debe poder listar los mantenimientos por estado de estos.
RF24	El sistema debe poder imprimir la lista de trabajo de un día en particular asignado a una cuadrilla en particular con la lista de materiales asociada.
RF25	El sistema debe poder extraer en Excel los mantenimientos realizados en un período determinado.
RF26	El sistema debe permitir extraer en Excel el listado de mantenimientos Pendientes, Programados y Reprogramados en un rango de fechas.

Requerimientos No Funcionales.

ID	Descripción
RNF01	El sistema debe ser desarrollado en JAVA.
RNF02	El sistema debe utilizar para persistencia de datos MySQL.
RNF03	El sistema debe funcionar en entorno Windows.
RNF04	El sistema debe estar instalado en infraestructura propia de la empresa en las oficinas de centrales y ser accesible de cualquier edificio de la misma.
RNF05	El sistema debe ser construido utilizando el patrón MVC (Modelo – Vista – Controlador)

Requerimientos Futuros.

ID	Descripción
RFU01	El sistema debe interactuar con el sistema de almacenes para mantener la lista de materiales a utilizar en los mantenimientos.
RFU02	El sistema debe permitir que las cuadrillas carguen los mantenimientos que surgen de la inspección visual en campo, en un estado de precarga para luego ser aprobados por el Analista de Mantenimiento.
RFU03	El sistema debe poder funcionar en dispositivos móviles para que las cuadrillas puedan visualizar los mantenimientos y registrar el resultado del trabajo en campo.

Inicio del Análisis

Casos de Uso

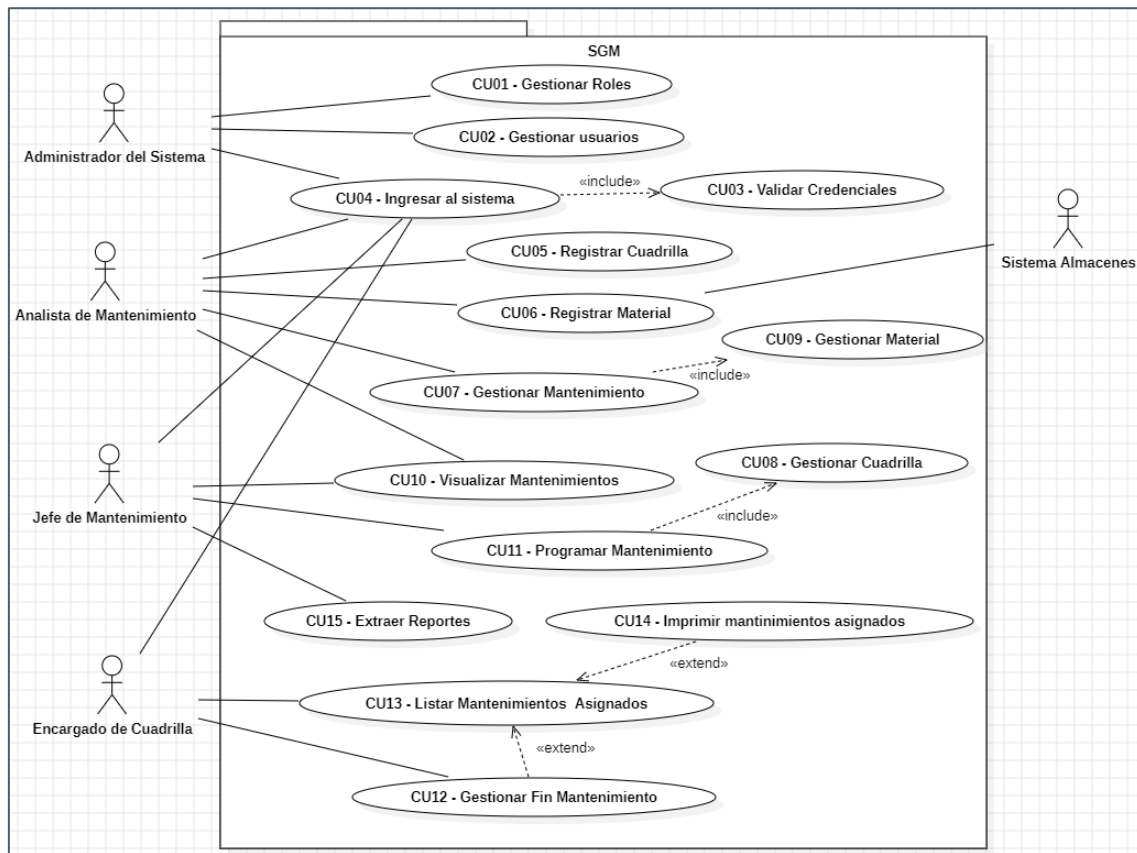


Figura 8 - Casos de Uso

Identificación de Actores

- Analista de Mantenimiento: Persona encargada de registrar los mantenimientos en el sistema.
- Jefe de Mantenimiento: Persona responsable programar el trabajo de las cuadrillas.
- Encargado de Cuadrilla: Persona a cargo del grupo de trabajo en campo que luego de realizar los trabajos en campo informa la finalización de estos.
- Administrador de Sistema: Persona encargada de generar roles y usuarios dentro del sistema, manteniendo el normal funcionamiento del mismo.

Actor fuera de alcance:

- Sistema Almacenes: Interfaz que mantiene automáticamente el listado de los materiales susceptibles de ser utilizados en los mantenimientos.

Trazabilidad

A continuación, se detallará la relación de los Casos de Uso confeccionados y los Requerimientos Funcionales detallados para asegurar el cubrimiento de los mismos.

Para ello centraremos el detalle en los casos de uso que nos permiten recorrer todo el ciclo de vida de un mantenimiento, desde su registración hasta su finalización.

Casos de Uso.

- CU07 – Gestionar Mantenimiento
- CU08 – Gestionar Cuadrilla
- CU09 – Gestionar Material
- CU11 – Programar Mantenimientos
- CU12 – Gestionar Fin Mantenimiento

Se detalla seguidamente los Requerimientos cubiertos por los Casos de Uso listados.

ID RQM	ID Caso Uso	Actor Principal	Paquete de Análisis	Descripción
RF09	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe permitir registrar un nuevo mantenimiento en el sistema con estado "Pendiente".
RF10	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe permitir especificar si el mantenimiento corresponde al plan anual de mantenimiento o al resultado de la inspección en campo.
RF11	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe permitir asignarle una prioridad al trabajo de mantenimiento dependiendo de la urgencia en la realización del mantenimiento.
RF12	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe poder cambiar la prioridad asignada al trabajo.
RF13	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe poder borrar un mantenimiento cargado.
RF14	CU07	Analista Mantenimiento	Mantenimiento	El sistema debe permitir modificar los datos de un mantenimiento cargado.
RF15	CU08	Jefe Mantenimiento	Mantenimiento	El sistema debe permitir Asignarle una cuadrilla al mantenimiento.
RF16	CU11	Jefe Mantenimiento	Mantenimiento	El sistema debe poder ingresar un tiempo de traslado al mantenimiento, tiempo que le insume a la cuadrilla arribar al lugar de trabajo.
RF17	CU11	Jefe Mantenimiento	Mantenimiento	El sistema debe permitir cambiar el tiempo de traslado asignado.
RF18	CU09	Analista Mantenimiento	Mantenimiento	El sistema debe permitir agregar materiales a un mantenimiento.
RF19	CU11	Jefe Mantenimiento	Mantenimiento	El sistema debe permitir ingresar una fecha de realización del trabajo

				cambiando el estado del mantenimiento a "Programado".
RF20	CU11	Jefe Mantenimiento	Mantenimiento	El sistema debe poder cambiar una fecha de realización del trabajo.
RF21	CU12	Encargado de Cuadrilla	Mantenimiento	El sistema debe poder cambiar el estado del trabajo de mantenimiento a "Realizado" cuando el trabajo fue realizado.
RF22	CU12	Encargado de Cuadrilla	Mantenimiento	El sistema debe poder cambiar el estado del trabajo de mantenimiento a "Reprogramar" cuando el trabajo no pudo ser realizado.

Definición de los Casos de Uso.

ID Caso de Uso	CU07	
Nombre	Gestionar Mantenimiento	
Descripción	El sistema permitir registrar un nuevo mantenimiento en estado "Pendiente".	
Referencia	RF09, RF10, RF11, RF12, RF13, RF14	
Actores	Analista de Mantenimiento	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea.	
Flujo Principal	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Mantenimiento".
	2	El usuario ingresa "Agregar Nuevo Mantenimiento" para ingresar un nuevo mantenimiento.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> • Instalación para mantener. • Prioridad. • Tipo de Mantenimiento. • Descripción del mantenimiento.
	4	Gestionar Materiales
Postcondición	El sistema ha agregado un mantenimiento en estado "Pendiente".	
Flujo Secundario 1	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Mantenimiento".
	2	El usuario ingresa "Modificar" para modificar los datos de un mantenimiento.
	3	El usuario ingresa el ID del mantenimiento a modificar.
	4	El usuario completa la información requerida: <ul style="list-style-type: none"> • Instalación para mantener. • Prioridad. • Tipo de Mantenimiento. Descripción del mantenimiento.
	5	Gestionar Materiales

Postcondición	El sistema ha actualizado el registro del mantenimiento.	
Flujo Secundario 2	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Mantenimiento".
	2	El usuario ingresa "Borrar" para eliminar el mantenimiento.
	3	El usuario ingresa el ID del mantenimiento a eliminar.
Postcondición	El sistema ha eliminado el mantenimiento.	
Excepciones	No se contempla	

ID Caso de Uso	CU08	
Nombre	Gestionar Cuadrilla	
Descripción	El sistema debe permitir Asignarle una cuadrilla al mantenimiento.	
Referencia	RF15	
Actores	Jefe de Mantenimiento	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El pedido de Mantenimiento se encuentra en Estado "Pendiente".	
Flujo Principal	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Cuadrilla".
	2	El usuario ingresa "Asignar Cuadrilla" para ingresar un nuevo mantenimiento.
	3	El usuario ingresa el código de cuadrilla a asignar.
Postcondición	El sistema ha asignado una cuadrilla a un mantenimiento en estado "Pendiente".	
Excepciones	No se contempla	

ID Caso de Uso	CU09	
Nombre	Gestionar Material	
Descripción	El sistema permitir gestionar los materiales asociados a un pedido de mantenimiento.	
Referencia	RF18	
Actores	Analista de Mantenimiento	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El usuario ingresó a Gestionar Mantenimiento.	
Flujo Principal	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Materiales".
	2	El usuario ingresa los códigos y cantidades de los materiales a utilizar en el mantenimiento.
Postcondición	El sistema ha agregado los materiales al mantenimiento.	
Excepciones	No se contempla	

ID Caso de Uso	CU11	
Nombre	Programar Mantenimiento	
Descripción	El sistema permitir realizar la programación de un pedido de mantenimiento en estado "Pendiente".	
Referencia	RF16, RF17, RF19, RF20	
Actores	Jefe de Mantenimiento	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El pedido se encuentra en estado "Pendiente".	
Flujo Principal	Acción	Descripción
	1	El usuario ingresa la opción "Programar Mantenimiento".
	2	El usuario ingresa el código del mantenimiento para programar.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> • Tiempo de traslado. • Fecha de programación.
	4	Gestionar Cuadrilla
Postcondición	El sistema ha agregado la información ingresada y pasado el pedido de mantenimiento a estado "Programado".	
Excepciones	No se contempla	

ID Caso de Uso	CU12	
Nombre	Gestionar Fin Mantenimiento	
Descripción	El sistema permite gestionar el resultado de un mantenimiento.	
Referencia	RF21, RF22	
Actores	Encargado de Cuadrilla	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El mantenimiento se encuentra en estado "Programado".	
Flujo Principal	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Fin Mantenimiento".
	2	El usuario ingresa el código del mantenimiento a gestionar.
	3	El usuario ingresa "Trabajo Realizado".
Postcondición	El sistema ha cambiado el estado del pedido de mantenimiento a "Realizado".	
Flujo Secundario 1	Acción	Descripción
	1	El usuario ingresa la opción "Gestionar Fin Mantenimiento".
	2	El usuario ingresa el código del mantenimiento a gestionar.
	3	El usuario ingresa "Trabajo No Realizado".
Postcondición	El sistema ha cambiado el estado del pedido de mantenimiento a "Reprogramar" y borra la fecha de programación y la cuadrilla asignada.	
Excepciones	No se contempla	

Etapa de Análisis

En la etapa de análisis se contemplarán los casos de uso definidos como pertenecientes al ciclo de vida de los mantenimientos y que fueron detallados en la etapa anterior, a saber:

Casos de Uso.

- CU07 – Gestionar Mantenimiento
- CU08 – Gestionar Cuadrilla
- CU09 – Gestionar Material
- CU11 – Programar Mantenimientos
- CU12 – Gestionar Fin Mantenimiento

A continuación, para detallar la interacción y mensajes entre los objetos que formarán parte del desarrollo, se presentan los diagramas de secuencia de cada caso de uso.

Diagrama Secuencia CU07 – Gestionar Mantenimiento

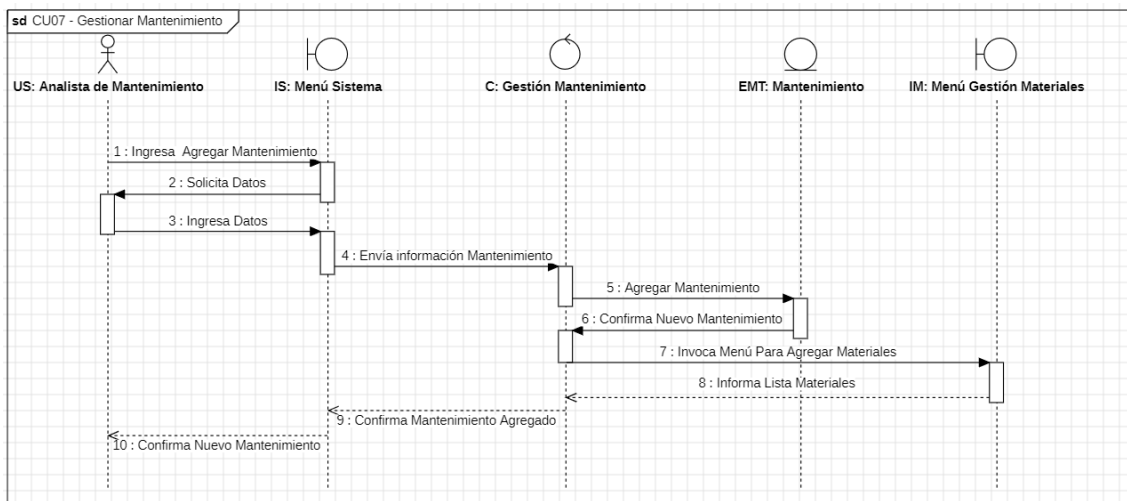


Figura 9 - Diagrama Secuencia CU07

Diagrama Secuencia CUo8 – Gestionar Cuadrilla

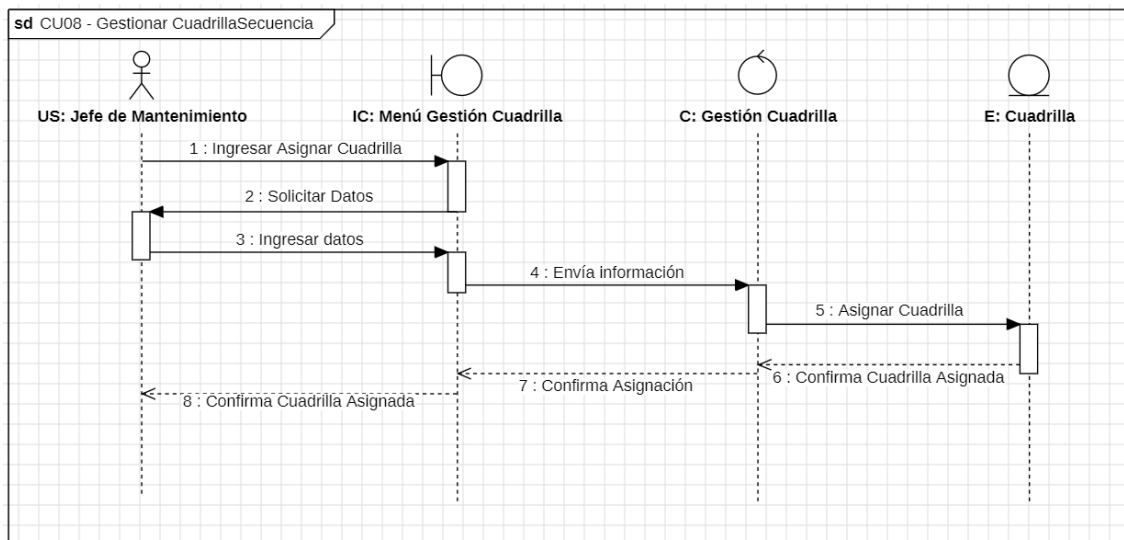


Figura 10 - Diagrama Secuencia CUo8

Diagrama Secuencia CUo9 – Gestionar Materiales

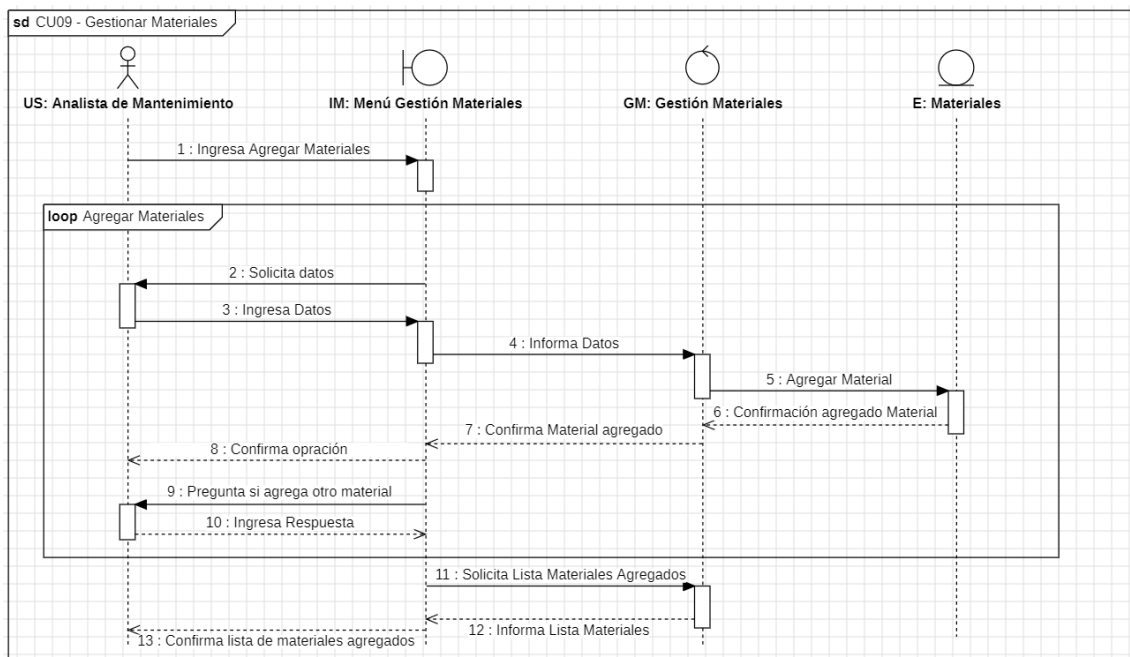


Figura 11 - Diagrama Secuencia CUo9

Diagrama Secuencia CU11 – Programar Mantenimiento

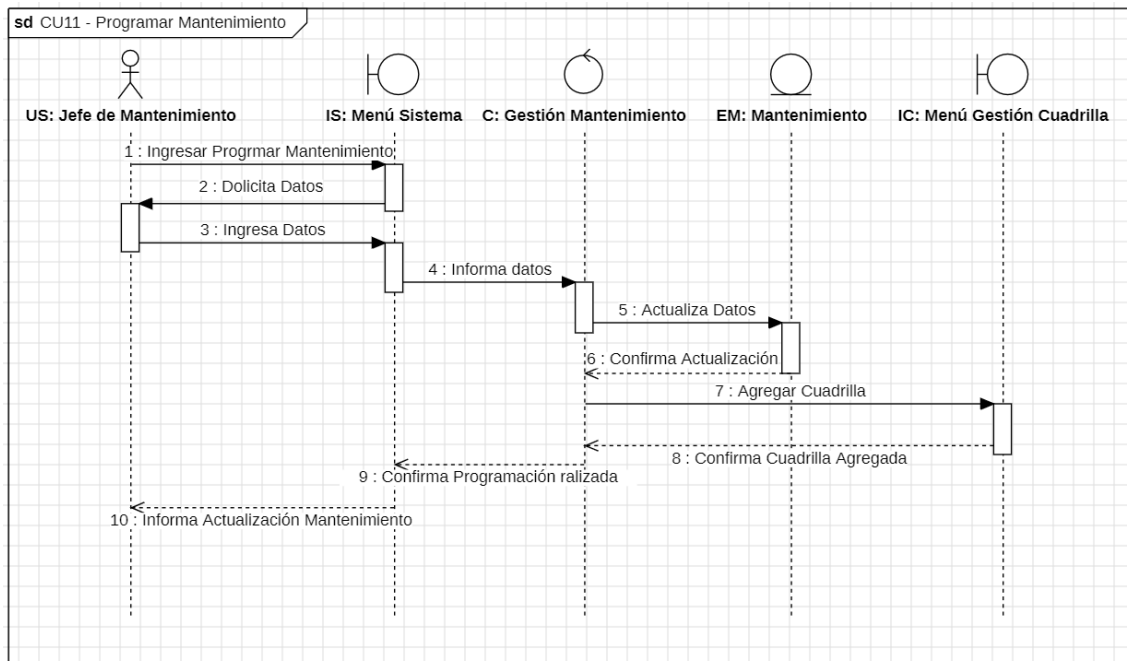


Figura 12 - Diagrama Secuencia CU11

Diagrama Secuencia CU12 – Gestionar Fin Mantenimiento

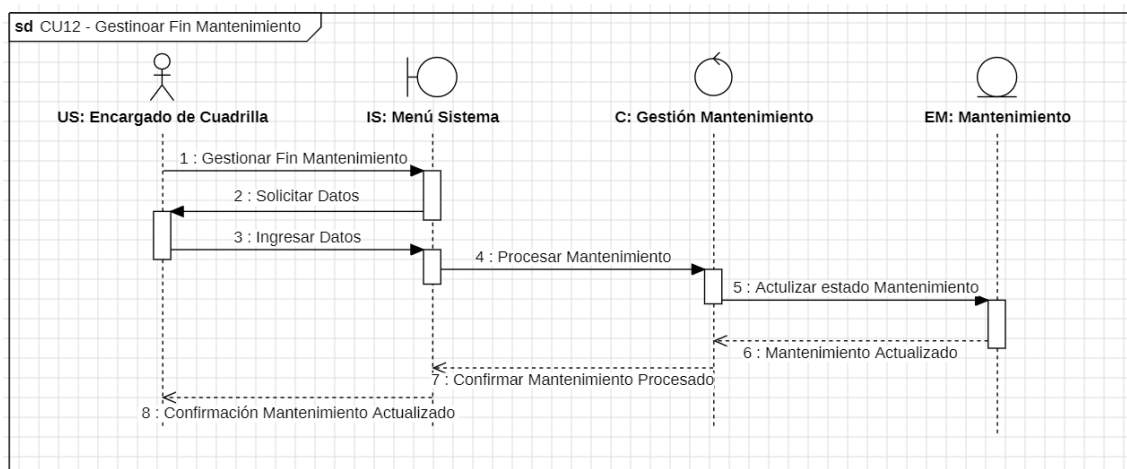


Figura 13 - Diagrama Secuencia CU12

Etapa de Diseño

Diagrama de Clases

A continuación, se presenta un diagrama de clases para describir las clases definidas y la relación entre ellas.

Para el caso de estudio, la clase GestionarMantenimiento es la responsable de mostrar el menú principal de la gestión de mantenimientos.

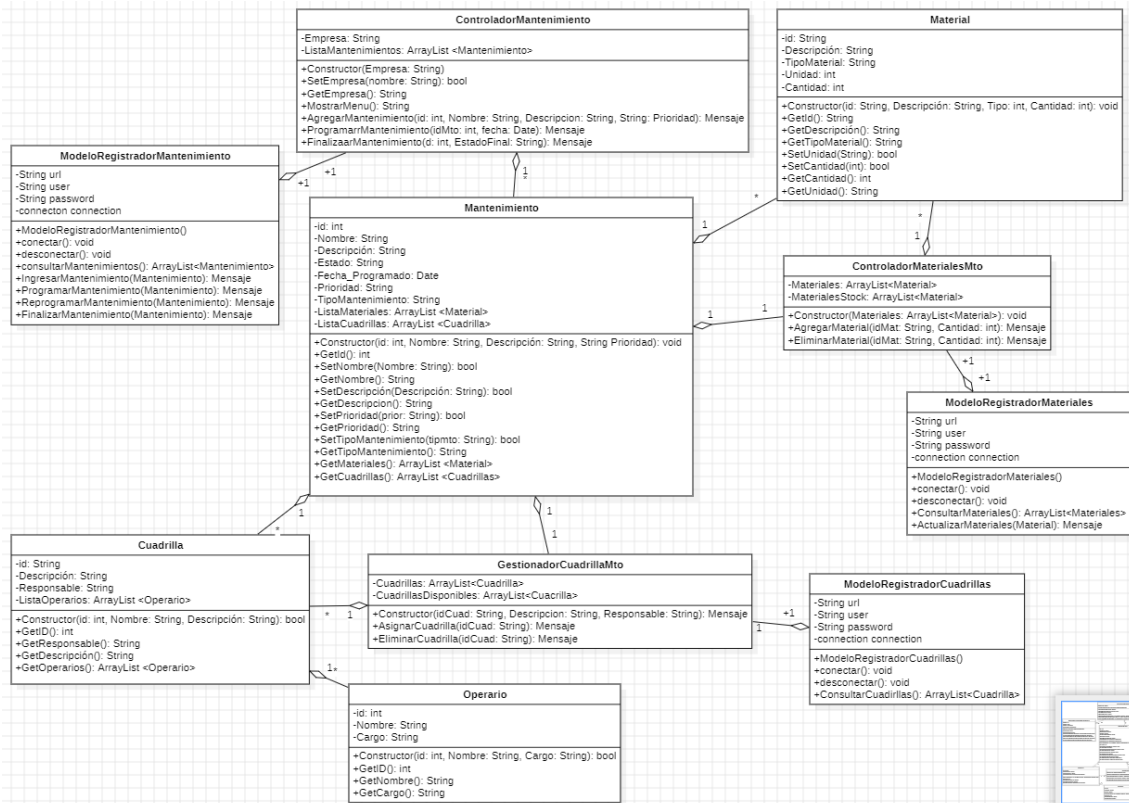


Figura 14 - Diagrama de Clases

Etapa de Implementación

Diagrama de Despliegue

Partiendo de la arquitectura diseñada para la implementación del sistema y sumado al cumplimiento de los requerimientos no funcionales que se detallan a continuación, se presenta el diagrama de despliegue.

Requerimientos no funcionales:

ID	Descripción
RNF01	El sistema debe ser desarrollado en JAVA.
RNF02	El sistema debe utilizar para persistencia de datos MySQL.
RNF03	El sistema debe funcionar en entorno Windows.
RNF04	El sistema debe estar instalado en infraestructura propia de la empresa en las oficinas de centrales y ser accesible de cualquier edificio de la misma.
RNF05	El sistema debe ser construido utilizando el patrón MVC (Modelo – Vista – Controlador)

Diagrama de Despliegue

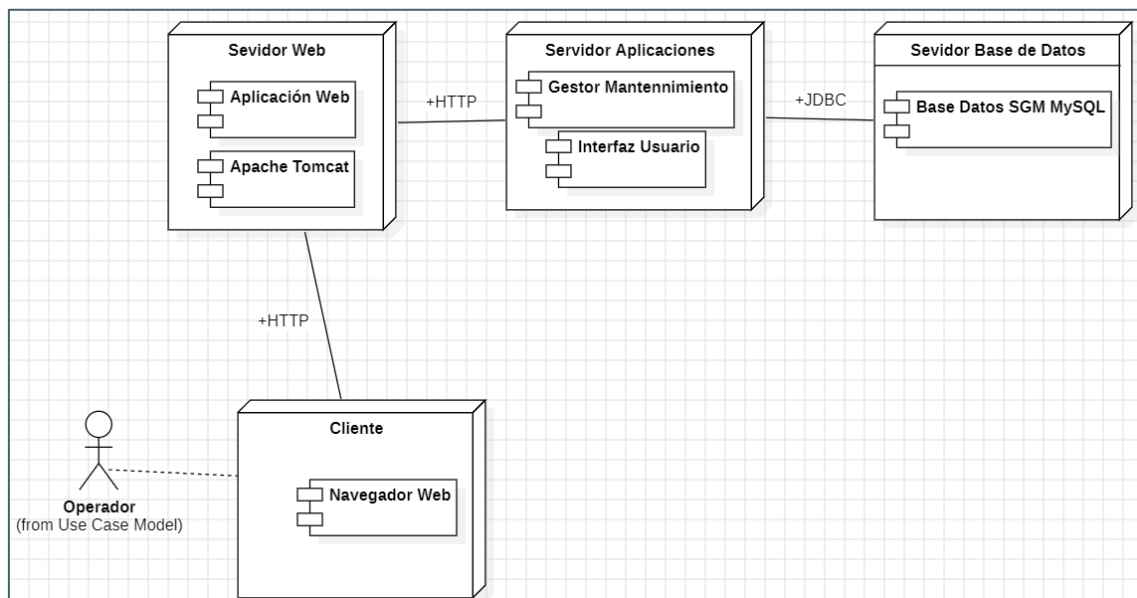


Figura 15 - Diagrama de despliegue

Los usuarios podrán conectarse vía navegador web desde cualquier computadora conectada a la red LAN propia de la empresa a un servidor Web donde se encontrará desplegada la Aplicación Web. Este último estará conectado a un servidor de aplicaciones que finalmente se conectará vía JDBC a la base de datos situada en un servidor para tal fin.

Etapa de Pruebas

Plan de Prueba

Cada una de las pruebas será identificada unívocamente por su código de ID que mantendrá la siguiente nomenclatura:

- Las dos primeras letras del código indican que se trata de un caso de prueba. Se utilizarán las tras CP.
- Dos dígitos indicando el número de caso de uso.
- Por último, dos dígitos indicando el número de prueba.

Solo se reflejarán en el plan de pruebas aquellas que correspondan a los casos de uso que contemplen el ciclo de vida de los Mantenimientos previstos en el apartado de Trazabilidad.

Caso Uso	ID Prueba	Tipo Prueba	Técnica	Descripción
CU07	CP0701	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo principal (Agregar Mantenimiento) del caso de uso.
CU07	CP0702	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo secundario 1 (Modificar Mantenimiento) del caso de uso.
CU07	CP0703	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo secundario 2 (Borrar Mantenimiento) del caso de uso.
CU08	CP0801	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo principal (Gestionar Cuadrilla) del caso de uso.
CU09	CP0901	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo principal (Gestionar Material) del caso de uso.
CU11	CP1101	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo principal (Programar Mantenimiento) del caso de uso.
CU12	CP1201	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo principal (Trabajo Realizado) del caso de uso.
CU12	CP1202	Sistema	Prueba Funcional	Verificación de cumplimiento del flujo secundario (Trabajo no realizado) del caso de uso.

Las pruebas previstas se realizarán sobre el entorno de TEST, teniendo promovido a dicho entorno los desarrollos necesarios que cubren los Casos de Uso previstos.

A continuación, se detallan los Casos de Prueba previstos en el “Camino Feliz” del proceso de gestión de los mantenimientos y que abarca el ciclo de vida completo del mismo.

ID Caso Prueba	CU0701	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Agregar Mantenimiento) del caso de uso.	
Referencia	Caso de Uso CU07 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea de Agregar Mantenimiento	
Datos de entrada	<ul style="list-style-type: none"> • Instalación para mantener: "Distribuidor Las Tunas" • Prioridad: "Alta" • Tipo Mantenimiento: "Por Inspección" • Descripción: "Cambio de 8 postes de madera y 5 crucetas en derivación estancia Los Teros" 	
Acciones	#	Descripción
	1	El usuario ingresa la opción "Gestionar Mantenimiento".
	2	El usuario ingresa "Agregar Nuevo Mantenimiento" para ingresar un nuevo mantenimiento.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> • Instalación para mantener. • Prioridad. • Tipo de Mantenimiento. • Descripción del mantenimiento.
Resultado Esperado	El sistema ha agregado un mantenimiento en estado "Pendiente" e informa al usuario. Inmediatamente el sistema muestra el menú para agregar materiales.	

ID Caso Prueba	CU0801	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Gestionar Cuadrilla) del caso de uso.	
Referencia	Caso de Uso CU08 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El pedido de Mantenimiento se encuentra en Estado "Pendiente".	
Datos de entrada	<ul style="list-style-type: none"> • Código de Cuadrilla: "CH01" 	
Acciones	#	Descripción
	1	El usuario ingresa la opción "Gestionar Cuadrilla".
	2	El usuario ingresa "Asignar Cuadrilla" para ingresar un nuevo mantenimiento.
	3	El usuario ingresa el código de cuadrilla a asignar.
Resultado Esperado	El sistema ha agregado una cuadrilla al mantenimiento e informa al usuario.	

ID Caso Prueba	CU0901	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Gestionar Material) del caso de uso.	
Referencia	Caso de Uso CU09 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El usuario ingresó a Gestionar Mantenimiento.	
Datos de entrada	<ul style="list-style-type: none"> • Código de Material: “MS08” • Cantidad: 2 • Código de Material: “A13” • Cantidad: 6 	
Acciones	#	Descripción
	1	El usuario ingresa la opción “Gestionar Materiales”.
	2	El usuario ingresa el código de material “MS08” y cantidad 2 a utilizar en el mantenimiento.
	3	El usuario ingresa a “Agregar Nuevo Material”
	4	El usuario ingresa el código de material “A13” y cantidad 6 a utilizar en el mantenimiento.
Resultado Esperado	El sistema ha agregado ambos materiales al mantenimiento.	

ID Caso Prueba	CU1101	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Programar Mantenimiento) del caso de uso.	
Referencia	Caso de Uso CU11 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El pedido se encuentra en estado “Pendiente”.	
Datos de entrada	<ul style="list-style-type: none"> • Código Mantenimiento: 1 • Fecha: 25/05/2024 • Tiempo traslado: 1.5 hs 	
Acciones	#	Descripción
	1	El usuario ingresa la opción “Programar Mantenimiento”.
	2	El usuario ingresa el código del mantenimiento “1” para programar.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> • Tiempo de traslado. • Fecha de programación.
	4	El usuario ingresa el código de material “A13” a utilizar en el mantenimiento.
Resultado Esperado	El sistema ha programado el mantenimiento e informado al usuario. A continuación el sistema muestra el menú para agregar Cuadrilla.	

ID Caso Prueba	CU1201	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Trabajo Realizado) del caso de uso.	
Referencia	Caso de Uso CU12 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El mantenimiento se encuentra en estado “Programado”.	
Datos de entrada	<ul style="list-style-type: none"> Código Mantenimiento: 1 	
Acciones	#	Descripción
	1	El usuario ingresa la opción “Gestionar Fin Mantenimiento”.
	2	El usuario ingresa el código del mantenimiento “1” para programar.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> “Trabajo Realizado”
Resultado Esperado	El sistema ha pasado a Estado “Finalizado” el mantenimiento e informado al usuario.	

ID Caso Prueba	CU1201	
Tipo Prueba	Sistema	
Descripción	Verificación de cumplimiento del flujo principal (Trabajo Realizado) del caso de uso.	
Referencia	Caso de Uso CU12 – Flujo Principal	
Precondición	El usuario ha ingresado al sistema con sus credenciales. El usuario tiene los permisos necesarios para realizar la tarea. El mantenimiento se encuentra en estado “Programado”.	
Datos de entrada	<ul style="list-style-type: none"> Código Mantenimiento: 2 	
Acciones	#	Descripción
	1	El usuario ingresa la opción “Gestionar Fin Mantenimiento”.
	2	El usuario ingresa el código del mantenimiento “2” para programar.
	3	El usuario completa la información requerida: <ul style="list-style-type: none"> “Trabajo No Realizado”
Resultado Esperado	El sistema ha pasado a Estado “Pendiente” el mantenimiento e informado al usuario.	

Tratamiento de Defectos

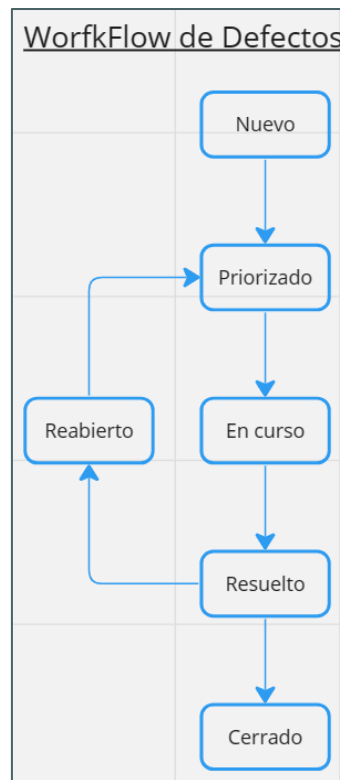
Para identificar y describir a cada defecto (incidente) reportado por el grupo de testeo, se enumeran a continuación los atributos a considerar para cada uno de ellos:

Atributo	Descripción
ID Incidente	Código único de identificación del incidente
Resumen	Breve descripción de la prueba donde se detecta el incidente
Componente	Componente al cual pertenece la funcionalidad sobre la que se realizó el caso de prueba
ID Caso de Prueba	Código único de identificación del caso de prueba que se está testeando y sobre el cual se detecta el incidente
Versión de software	Identificación de la versión de software que se está sometiendo a prueba
Usuario de creación	Nombre y Apellido del tester que detecta el incidente
Fecha y hora de creación	Fecha y hora de la registración del incidente
Fecha y hora de última actualización	Fecha y hora de la última actualización del registro del incidente
Estado	Código que identifica la situación del registro del incidente. Permite establecer al grupo de usuarios que tienen la responsabilidad de realizar la próxima acción
Asignado a	Persona a la cual está asignado el incidente
Síntoma	Respuesta del sistema ante la aparición del defecto
Severidad	Nivel de criticidad que presenta la falla respecto de la funcionalidad bajo testeo
Prioridad	Nivel de importancia del incidente. Establece la prioridad de la corrección del incidente
Resolución	Tipificación de la solución implementada en el incidente
Fecha y Hora de resolución	Fecha y hora en que el equipo de desarrollo realizó la modificación del software que soluciona el incidente

A continuación, se profundiza sobre algunos atributos para ordenar la gestión de los incidentes.

Estado de Incidencia

A continuación, se detalla el WorkFlow de los estados de los defectos detectados con sus correspondientes transiciones de estado.

Descripción de estados:

Estado	Descripción
Nuevo	Al detectar un incidente en las pruebas, el mismo será registrado por el Tester en la herramienta de gestión de incidencias y su estado inicial será Nuevo. Los incidentes en este estado están disponibles para que el Comité de Priorización de Incidentes (CPI) analice el defecto y priorice al mismo
Priorizado	En este estado se encuentran los incidentes que fueron analizados y priorizados por el CPI y están disponibles para ser corregidos
En Curso	En este estado se encuentran todos los incidentes que están siendo resueltos por los desarrolladores
Resuelto	En este estado se encuentran todos los incidentes que fueron resueltos por los desarrolladores y se encuentran listos para ser probados
Cerrado	En este estado se encuentran todos los incidentes cuya corrección fue probada dando un resultado satisfactorio
Reabierto	En este estado se encuentran los incidentes que, habiendo sido resueltos con anterioridad, fueron detectados nuevamente en las pruebas. Los incidentes en este estado se encuentran disponibles para ser analizados y priorizados por el CPI

Prioridad de Incidencia

EL CPI deberá analizar los defectos nuevos y reabiertos, estableciendo una prioridad para cada uno, que será tenida en cuenta al momento de establecer el orden de las correcciones por parte del equipo de desarrollo.

Descripción de prioridades:

Prioridad	Descripción
Urgente	La resolución del incidente debe ser inmediata
Alta	El incidente debe ser solucionado lo antes posible
Media	El incidente debe ser solucionado en algún momento
Baja	El incidente debe ser solucionado en última instancia

Severidad de Incidencia

La severidad se determina en función del impacto del incidente en la funcionalidad bajo testeo.

Descripción de prioridades:

Severidad	Descripción
Bloqueante	Incidente que impide continuar con el proceso, bloqueando al usuario
Crítica	Incidente que permite continuar con el proceso, pero genera inconsistencia en datos que imposibilitan el normal funcionamiento del sistema
Mayor	Incidente que permite continuar con el proceso a través de un workaround
Menor	Incidente generado por cambios de apariencia o usabilidad. Sin influencia en las operaciones diarias. Cambios de logos, parámetros, colores, etc.

Resolución de Incidencia

La resolución es un campo que deben completar los desarrolladores en base a la corrección u observación realizada y que solucione el incidente reportado.

Descripción de resolución:

Resolución	Descripción
Corregido	Se realizó una modificación en la codificación del software para solucionar el incidente
Duplicado	Ya existe un incidente abierto por el mismo inconveniente
Parametrización	El incidente fue resuelto modificando una parametrización del software sin modificación de código fuente
Funciona según diseño	Se verifica que el sistema funciona según el diseño del mismo. Se deberá revisar si el caso de prueba responde al comportamiento definido en el requerimiento

Interfaz Gráfica

A continuación, se presenta un prototipo de interfaz gráfica para la ventana principal del sistema, para ser presentada al usuario.

ID	Prioridad	Tipo	Estado	Nombre	Descripción	Fecha Programado
1	Alta	Por Inspección	Pendiente	Distribuidor Las Tunas	Cambio de 8 postes de madera y 5 crucetas en derivación estancia Los Teros	

Figura 16 - Interfaz Ventana Principal

En esta ventana, según el usuario perfil de usuario que haya ingresado al sistema, se presentarán habilitadas aquellas opciones a las que el perfil posee permisos. Además, los usuarios podrán visualizar los mantenimientos que, según su estado, tengan acceso a poder visualizar y gestionar.

Definición de la base de datos para el prototipo

A continuación, se detalla mediante un diagrama de entidad relación la estructura de la Base de Datos que soportará el sistema a desarrollar.

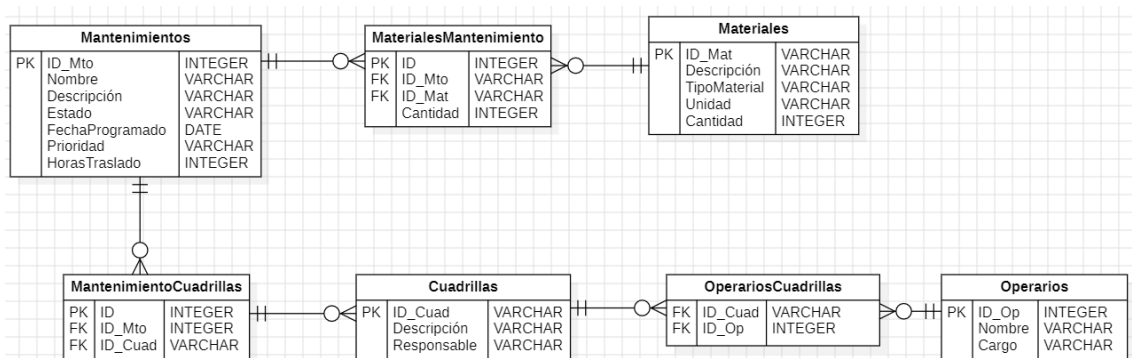


Figura 17 - Diagrama Entidad Relación del sistema

Detalle de Tablas

Tabla Mantenimiento: Contiene la información de cada mantenimiento desglosado en los siguientes campos:

- ID: código único de identificación que es además clave primaria de la tabla
- Nombre: Nombre del mantenimiento
- Descripción: Detalle de las tareas a realizar.
- Estado: estado del mantenimiento (Pendiente, Programado, Realizado)
- Fecha Programado: Fecha donde se establece realizar el trabajo.
- Prioridad: importancia del mantenimiento (Alta, Media, Baja)
- Tipo de Mantenimiento: describe el origen del mantenimiento (por plan anual de mantenimiento o por inspección)

Script de creación:



```
1 CREATE TABLE `mantenimiento` (  
2     `ID` int NOT NULL,  
3     `Nombre` varchar(45) NOT NULL,  
4     `Descripcion` varchar(1000) NOT NULL,  
5     `Estado` varchar(15) NOT NULL,  
6     `FechaProgramado` date DEFAULT NULL,  
7     `Prioridad` varchar(10) DEFAULT NULL,  
8     `TipoMantenimiento` varchar(20) DEFAULT NULL,  
9     PRIMARY KEY (`ID`)  
10 ) ;
```

Figura 18 - Script Creación Tabla Mantenimiento

Tabla Materiales: Contiene la información de los materiales desglosado en los siguientes campos:

- IDMat: código único de identificación que es además clave primaria de la tabla
- Descripción: Detalle del material.
- Tipo de Material: describe el nivel de tensión al que pertenece.

Script de creación:



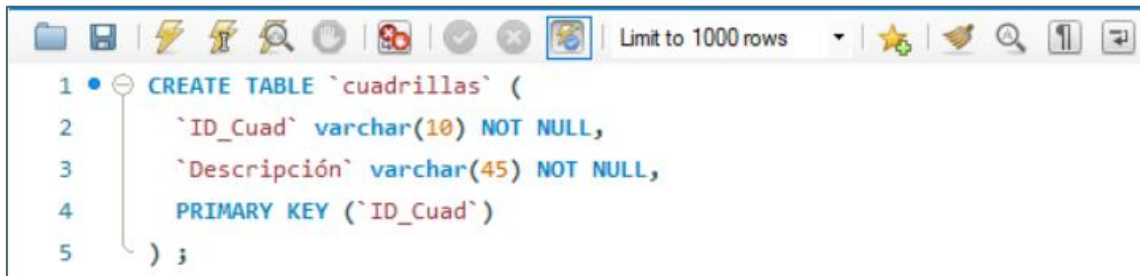
```
1 CREATE TABLE `materiales` (  
2     `IDMat` varchar(10) NOT NULL,  
3     `Descripcion` varchar(45) NOT NULL,  
4     `TipoMaterial` varchar(45) NOT NULL,  
5     PRIMARY KEY (`IDMat`)  
6 ) ;
```

Figura 19 - Script creación tabla materiales

Tabla Cuadrillas: Contiene la información de cada cuadrilla desglosado en los siguientes campos:

- ID_Cuad: código único de identificación que es además clave primaria de la tabla
- Descripción: Detalle de la cuadrilla.

Script de creación:



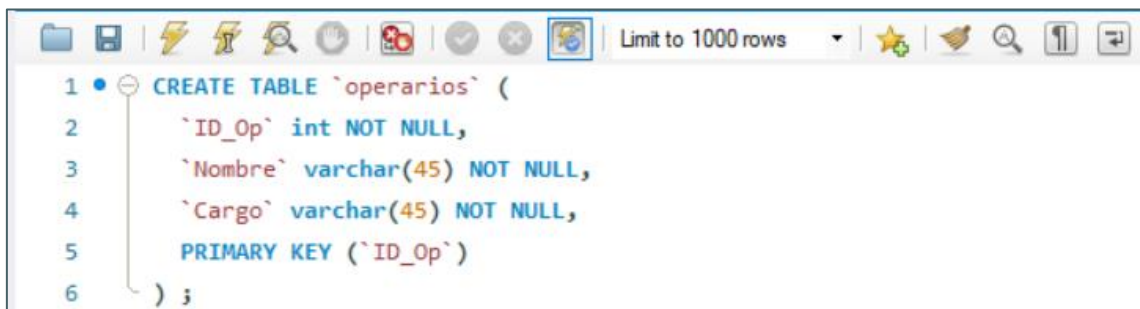
```
1 CREATE TABLE `cuadrillas` (  
2     `ID_Cuad` varchar(10) NOT NULL,  
3     `Descripción` varchar(45) NOT NULL,  
4     PRIMARY KEY (`ID_Cuad`)  
5 ) ;
```

Figura 20 - Script creación tabla Cuadrillas

Tabla Operarios: Contiene la información de cada operario que compone a las cuadrillas, desglosado en los siguientes campos:

- ID_Op: código único de identificación que es además clave primaria de la tabla
- Nombre: Nombre del operario
- Cargo: puesto que ocupa.

Script de creación:

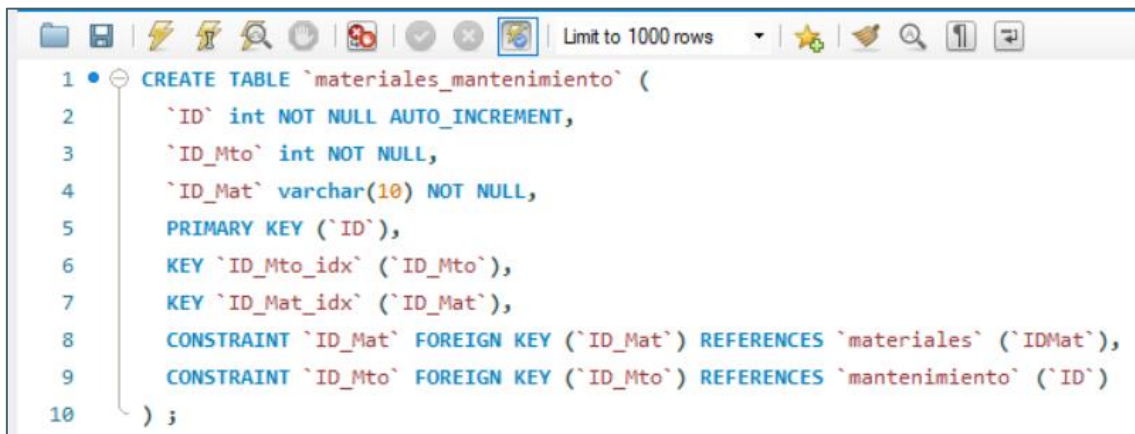


```
1 CREATE TABLE `operarios` (  
2     `ID_Op` int NOT NULL,  
3     `Nombre` varchar(45) NOT NULL,  
4     `Cargo` varchar(45) NOT NULL,  
5     PRIMARY KEY (`ID_Op`)  
6 ) ;
```

Figura 21 - Script Creación tabla operarios

Tabla MaterialesMantenimiento: Contiene la relación de qué materiales están asociados a cada mantenimiento, desglosado en los siguientes campos:

- ID: código único de identificación que es además clave primaria de la tabla
- ID_Mto: Id del mantenimiento. Este campo es clave foránea con la tabla Mantenimiento.
- ID_Mat: Id del Material. Este campo es clave foránea con la tabla Materiales.

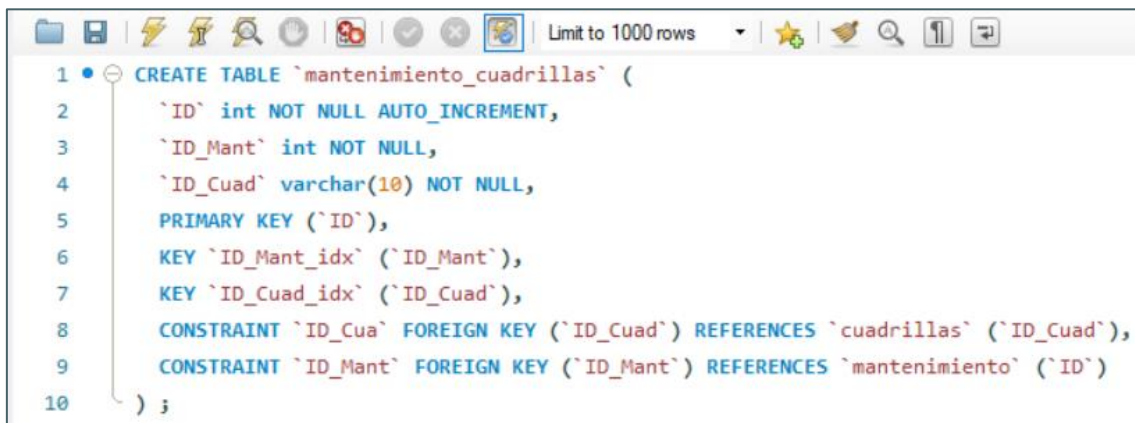
Script de creación:

```
1 CREATE TABLE `materiales_mantenimiento` (  
2   `ID` int NOT NULL AUTO_INCREMENT,  
3   `ID_Mto` int NOT NULL,  
4   `ID_Mat` varchar(10) NOT NULL,  
5   PRIMARY KEY (`ID`),  
6   KEY `ID_Mto_idx` (`ID_Mto`),  
7   KEY `ID_Mat_idx` (`ID_Mat`),  
8   CONSTRAINT `ID_Mat` FOREIGN KEY (`ID_Mat`) REFERENCES `materiales` (`IDMat`),  
9   CONSTRAINT `ID_Mto` FOREIGN KEY (`ID_Mto`) REFERENCES `mantenimiento` (`ID`)  
10  ) ;
```

Figura 22 - Script creación tabla materiales mantenimiento

Tabla MantenimientoCuadrilla: Contiene la relación de qué cuadrillas están asociadas a cada mantenimiento, desglosado en los siguientes campos:

- ID: código único de identificación que es además clave primaria de la tabla
- ID_Mto: Id del mantenimiento. Este campo es clave foránea con la tabla Mantenimiento.
- ID_Cuad: Id de la Cuadrilla. Este campo es clave foránea con la tabla Cuadrilla.

Script de creación:

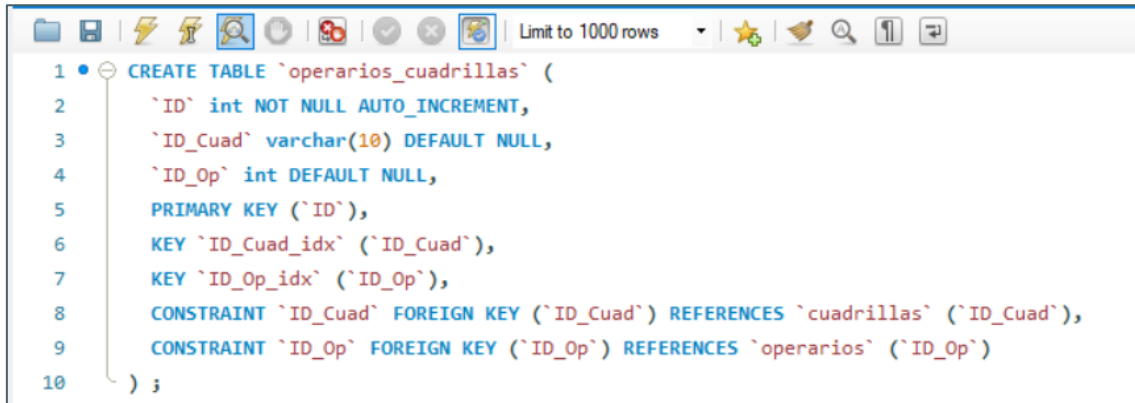
```
1 CREATE TABLE `mantenimiento_cuadrillas` (  
2   `ID` int NOT NULL AUTO_INCREMENT,  
3   `ID_Mant` int NOT NULL,  
4   `ID_Cuad` varchar(10) NOT NULL,  
5   PRIMARY KEY (`ID`),  
6   KEY `ID_Mant_idx` (`ID_Mant`),  
7   KEY `ID_Cuad_idx` (`ID_Cuad`),  
8   CONSTRAINT `ID_Cua` FOREIGN KEY (`ID_Cuad`) REFERENCES `cuadrillas` (`ID_Cuad`),  
9   CONSTRAINT `ID_Mant` FOREIGN KEY (`ID_Mant`) REFERENCES `mantenimiento` (`ID`)  
10  ) ;
```

Figura 23 - Script creación tabla mantenimiento cuadrillas

Tabla OperariosCuadrilla: Contiene la relación de qué operarios están asociadas a cada Cuadrilla, desglosado en los siguientes campos:

- ID: código único de identificación que es además clave primaria de la tabla
- ID_Cuad: Id de la Cuadrilla. Este campo es clave foránea con la tabla Cuadrilla.
- ID_Op: Id del operario. Este campo es clave foránea con la tabla Operarios.

Script de creación:



```

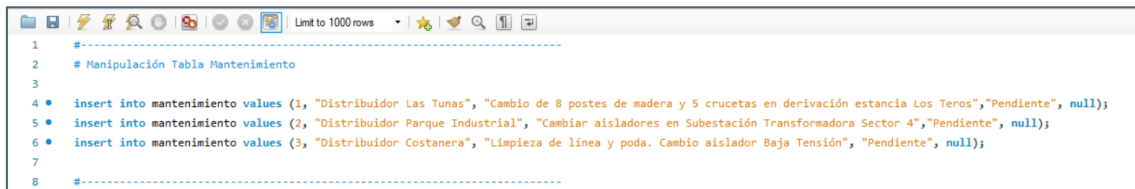
1 CREATE TABLE `operarios_cuadrillas` (
2   `ID` int NOT NULL AUTO_INCREMENT,
3   `ID_Cuad` varchar(10) DEFAULT NULL,
4   `ID_Op` int DEFAULT NULL,
5   PRIMARY KEY (`ID`),
6   KEY `ID_Cuad_idx` (`ID_Cuad`),
7   KEY `ID_Op_idx` (`ID_Op`),
8   CONSTRAINT `ID_Cuad` FOREIGN KEY (`ID_Cuad`) REFERENCES `cuadrillas` (`ID_Cuad`),
9   CONSTRAINT `ID_Op` FOREIGN KEY (`ID_Op`) REFERENCES `operarios` (`ID_Op`)
10  ) ;

```

Figura 24 - Script creación tabla operarios cuadrillas

Manipulación de Tablas

Insertión en Tablas:



```

1 #-----
2 # Manipulación Tabla Mantenimiento
3 #-----
4 insert into mantenimiento values (1, "Distribuidor Las Tunas", "Cambio de 8 postes de madera y 5 crucetas en derivación estancia Los Teros", "Pendiente", null);
5 insert into mantenimiento values (2, "Distribuidor Parque Industrial", "Cambiar aisladores en Subestación Transformadora Sector 4", "Pendiente", null);
6 insert into mantenimiento values (3, "Distribuidor Costanera", "Limpieza de línea y poda. Cambio aislador Baja Tensión", "Pendiente", null);
7 #-----
8

```



```

8 #-----
9 # Manipulación Tabla Mantenimiento
10 #-----
11 insert into materiales values ("MS08", "Poste Madera 8 metros", "Baja Tensión");
12 insert into materiales values ("MS11", "Poste Madera 11 metros", "Media Tensión");
13 insert into materiales values ("HS08", "Columna Hormigón 8 metros", "Baja Tensión");
14 insert into materiales values ("HS11", "Columna Hormigón 11 metros", "Media Tensión");
15 insert into materiales values ("CRM", "Cruceta de Madera", "Media Tensión");
16 insert into materiales values ("CRH", "Cruceta de Hormigón", "Media Tensión");
17 insert into materiales values ("CRB", "Cruceta de Madera Baja Tensión", "Baja Tensión");
18 insert into materiales values ("A13", "Aislador 13 kV", "Media Tensión");
19 insert into materiales values ("A33", "Aislador 33 kV", "Media Tensión");
20 insert into materiales values ("ABT", "Aislador Baja Tensión", "Baja Tensión");
21 #-----

```



```
24 #-----
25 # Manipulación Tabla Materiales_Mantenimiento
26
27 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 1, "MS08");
28 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 1, "MS08");
29 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 1, "CRM");
30 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 1, "CRM");
31 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 2, "A13");
32 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 2, "A13");
33 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 2, "A13");
34 • insert into materiales_mantenimiento (ID_Mto, ID_Mat) values ( 3, "ABT");
35
36 #-----
```

```
39 #-----
40 # Manipulación Tabla Operarios
41
42 • insert into operarios values (28676538, "Rettore Ricardo", "Encargado");
43 • insert into operarios values (26589145, "López José", "Encargado");
44 • insert into operarios values (36569458, "Landra Pablo", "Oficial");
45 • insert into operarios values (40256895, "Ríos Héctor", "Oficial");
46 • insert into operarios values (42571023, "Hunz Leandro", "Ayudante");
47 • insert into operarios values (46125784, "Vega Ignacio", "Ayudante");
48
49 #-----
```

```
51 #-----
52 # Manipulación Tabla Cuadrillas
53
54 • insert into cuadrillas values ( "CH01", "Cuadrilla con Hidro H100");
55 • insert into cuadrillas values ( "CP01", "Cuadrilla poda con Hidro H10");
56
57 #-----
58
```

```

60  #-----
61  # Manipulación Tabla Operarios_Cuadrillas
62
63  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CH01", 28676538);
64  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CH01", 36569458);
65  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CH01", 42571023);
66  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CP01", 26589145);
67  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CP01", 40256895);
68  • insert into operarios_cuadrillas (ID_Cuad, ID_Op) values ( "CP01", 46125784);
69
70  #-----

```

```

73  #-----
74  # Manipulación Tabla Mantenimiento_Cuadrillas
75
76  • insert into mantenimiento_cuadrillas (ID_Mant, ID_Cuad) values (1, "CH01");
77  • insert into mantenimiento_cuadrillas (ID_Mant, ID_Cuad) values (2, "CH01");
78
79  #-----

```

Consulta de datos

Query 1 Script Insert tablas Script Creación Tablas Borrado de Tablas Consulta de datos

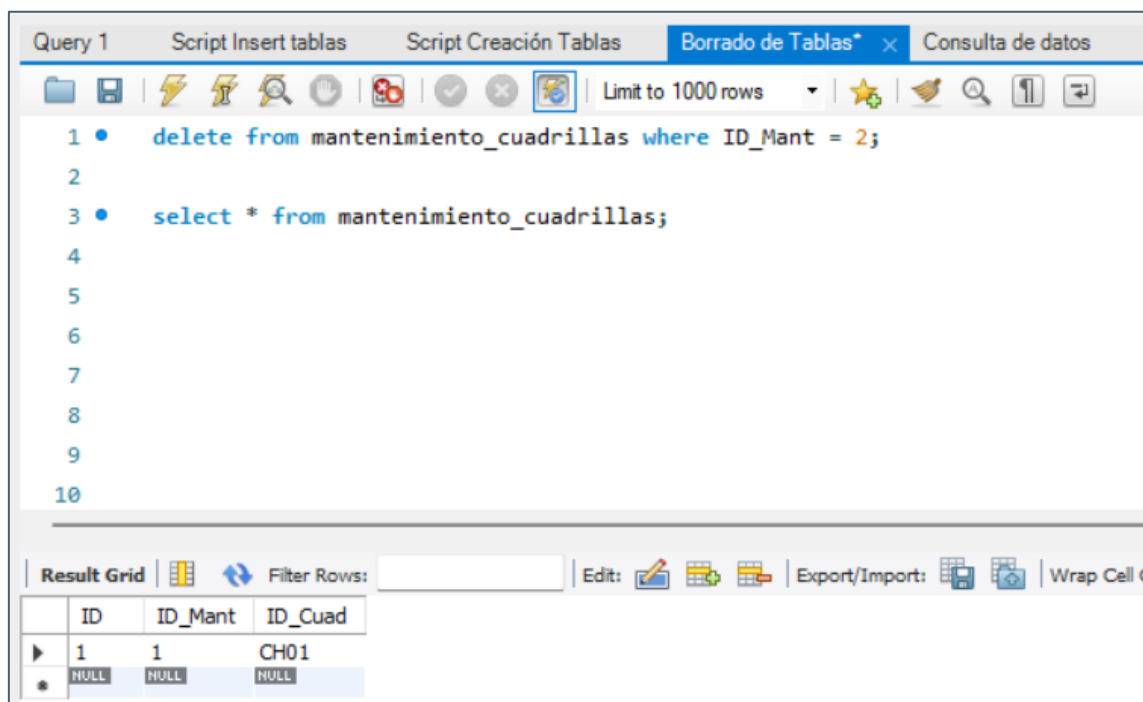
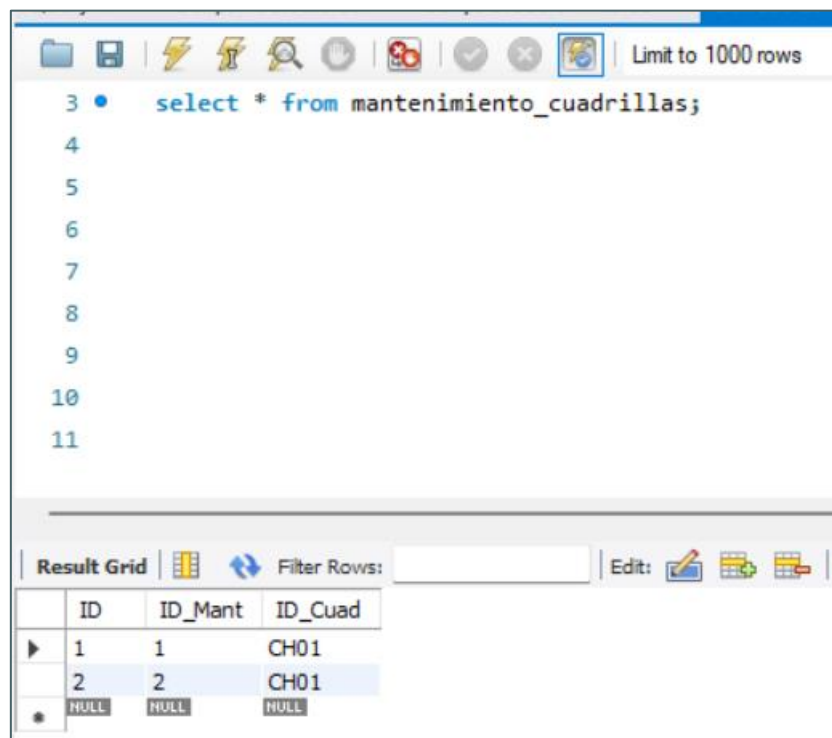
```

1  # consulta de datos de los materiales asociados al mantenimiento 1
2  • select mat.IDMat as "Código", mat.Descripcion as "Descripción", count(*) as cantidad
3  from materiales_mantenimiento as matmto, mantenimiento as mto, materiales as mat
4  where mto.ID = 1
5  and mto.id = matmto.ID_Mto
6  and matmto.ID_Mat = mat.IDMat
7  group by mat.IDMat, mat.Descripcion;
8
9

```

Result Grid Filter Rows: Export: Wrap Cell Content: [IA](#)

	Código	Descripción	cantidad
▶	MS08	Poste Madera 8 metros	2
	CRM	Cruceta de Madera	2

Borrado de tablas

Desarrollo del Sistema

El sistema será diseñado utilizando JAVA, de forma tal de dar cumplimiento al requerimiento no funcional.

ID	Descripción
RNF01	El sistema debe ser desarrollado en JAVA.

El diseño de las clases estará dividido en tres grupos que se detallan a continuación:

- Tipo Interfaz/Vista: son aquellas clases destinadas a realizar las interfaces con el usuario y se encargan de invocar a las clases de Tipo Controlador y pasarle la información requerida en cadenas de texto.
- Tipo Controlador: son aquellas clases que contiene las reglas de negocio e implementan la verificación de datos y control de las acciones sobre las clases de Tipo Modelo.
- Tipo Modelo: son las clases que representan a los objetos del negocio.

A continuación, se listan las clases definidas en el diseño por tipo;

- Tipo Interfaz/Vista:
 - Clase MenuPrincipal: Se encarga de interactuar con el usuario, mostrando el menú de las distintas acciones a realizar con los Mantenimientos (Agregar Mantenimiento, Programar Mantenimiento y Finalizar Mantenimiento).
 - Clase MenuMaterialesMto: Se encarga de interactuar con el usuario, mostrando el menú de las distintas acciones para agregar y eliminar materiales de un Mantenimiento en particular.
 - Clase MenuCuadrillas: Se encarga de interactuar con el usuario, mostrando el menú de las distintas acciones para asignar y desasignar cuadrillas de un Mantenimiento en particular.
- Tipo Controlador:
 - Clase MenuPrincipalSGM: es la clase encargada de Gestionar los Mantenimientos, que recibe información ingresada por el usuario, valida la consistencia de los tipos de datos, valida la consistencia de la información ingresada y aplica las reglas de negocio para la generación, programación y finalización de los mantenimientos. Esta clase cuenta dentro de sus atributos con una estructura de datos que contiene a Objetos de tipo Mantenimiento. Finalmente, esta clase envía información a través de mensajes, una clase simple creada para informar los resultados de la operación.
 - Clase ControladorMaterialesMto: es la clase que recibe información ingresada por el usuario, valida la consistencia de los tipos de datos, valida la consistencia de la información ingresada y aplica las reglas de negocio para agregar y eliminar materiales de un mantenimiento. Esta clase cuenta dentro de sus atributos con una estructura de datos que contiene a Objetos de tipo Material. Finalmente, esta clase envía información a través de mensajes, una clase simple creada para informar los resultados de la operación.
 - Clase ControladorCuadrillasMto: es la clase que recibe información ingresada por el usuario, valida la consistencia de los tipos de datos, valida la consistencia de la información ingresada y aplica las reglas de negocio para asignar y desasignar materiales de un mantenimiento. Esta clase cuenta dentro de sus

atributos con una estructura de datos que contiene a Objetos de tipo Cuadrilla. Finalmente, esta clase envía información a través de mensajes, una clase simple creada para informar los resultados de la operación.

- Tipo Modelo:
 - Clase Mantenimiento: Es la clase que modela a cada mantenimiento con sus atributos y métodos, sin relacionarse con el usuario y sin conocer las reglas de negocio. Esta clase es contenedora de objetos tipo Material y de tipo Cuadrilla.
 - Clase Materiales: Es la clase que modela a cada material con sus atributos y métodos, sin relacionarse con el usuario y sin conocer las reglas de negocio.
 - Clase Cuadrilla: Es la clase que modela a cada cuadrilla con sus atributos y métodos, sin relacionarse con el usuario y sin conocer las reglas de negocio.

Nota: para acotar el tamaño del prototipo, se ha decidido prescindir de la clase Operario.

En este diseño solo es necesario implementar relación de agregación entre los objetos intervinientes, sin necesidad de implementar Herencia ni Polimorfismo.

Todas las clases en Java heredan de la clase Object, por lo cual se han redefinido en algunas clases los métodos toString y equals.

Análisis de las clases.

Clases Tipo Modelo.

Clase Mantenimiento.

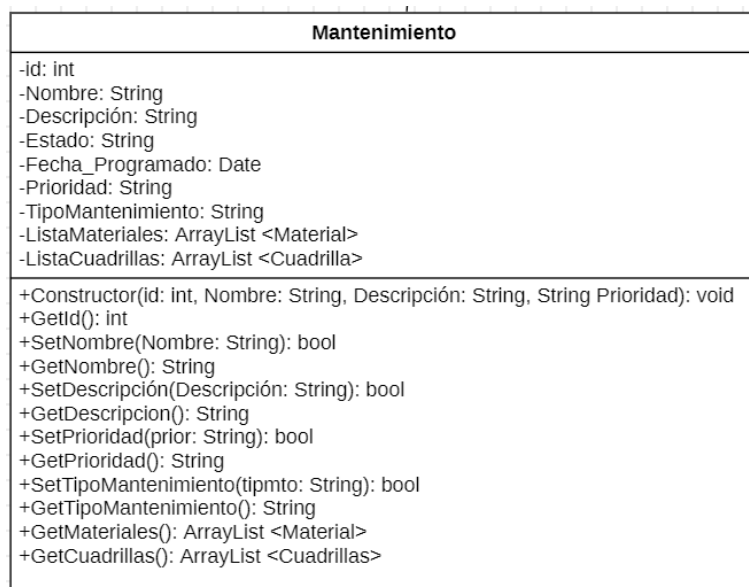


Figura 25 – Diagrama clase Mantenimiento

La clase Mantenimiento tiene una relación de agregación con las clases Material y Cuadrilla. Todos los atributos están encapsulados, utilizando el modificador private, pudiendo acceder a ellos sólo a través de sus métodos públicos que fueron definidos con el modificador public. Dentro de sus atributos se visualizan dos ArrayList para contener dichos objetos. Básicamente esta clase posee setters y getters, dado que es una clase Tipo Modelo.

Atributos de la clase.

```

public class Mantenimiento {

    /**
     * Declaración de atributos encapsulados con modificador private
     */

    private final int id;
    private String nombre;
    private String descripcion;
    private String estado;
    private String tipo;
    private String prioridad;
    private int horasTraslado;
    private Date fechaProgramado;
    private ArrayList <Material> materiales;
    private ArrayList <Cuadrilla> cuadrillas;
}

```

Figura 26 - Definición de los atributos de la clase Mantenimiento

Constructor de la clase.

```

public Mantenimiento(int id, String nombre, String descripcion, String tipo, String prioridad) {
    this.id = id;
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.tipo = tipo;
    this.prioridad = prioridad;
    this.estado = "Pendiente";
    this.horasTraslado = 0;
    this.materiales = new ArrayList<>();
    this.cuadrillas = new ArrayList<>();
}

```

Figura 27 - Constructor de la clase

El constructor de la clase es el primer método que se ejecuta al crear el objeto. Se identifica porque posee el mismo nombre que la clase. En este caso no es necesario sobrecargarlo, es decir, definir métodos con el mismo nombre y diferentes argumentos.

Métodos de la clase.

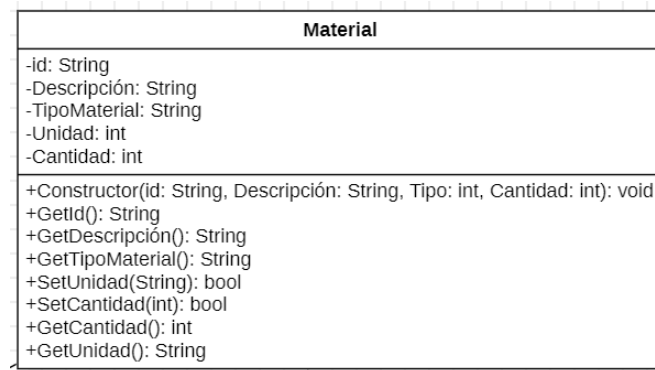
```

/**
 * Setters y Getters
 */

public String getNombre() { ...3 lines }
public void setNombre(String nombre) { ...3 lines }
public String getDescripcion() { ...3 lines }
public void setDescripcion(String descripcion) { ...3 lines }
public String getEstado() { ...3 lines }
public void setEstado(String estado) { ...3 lines }
public String getTipo() { ...3 lines }
public void setTipo(String tipo) { ...3 lines }
public int getId() { ...3 lines }
public String getPrioridad() { ...3 lines }
public void setPrioridad(String prioridad) { ...3 lines }
public int getHorasTraslado() { ...3 lines }
public void setHorasTraslado(int horasTraslado) { ...3 lines }
public Date getFechaProgramado() { ...3 lines }
public void setFechaProgramado(Date fechaProgramado) { ...3 lines }

```

Figura 28 - Setters y Getters

Clase Material*Figura 29 - Diagrama clase Material*

La clase Material es contenida, pero no contiene a otras clases. Todos los atributos están encapsulados, utilizando el modificador private, pudiendo acceder a ellos sólo a través de sus métodos públicos que fueron definidos con el modificador public. Básicamente esta clase posee setters y getters, dado que es una clase Tipo Modelo.

Atributos de la clase.

```
public class Material {  
  
    /*  
     * Declaración de atributos encapsulados con modificador private  
     */  
  
    private final String id;  
    private String descripcion;  
    private String tipo;  
    private String unidad;  
    private int cantidad;  
}
```

*Figura 30 - Definición de atributos*Constructor de la clase.

```
public Material(String id, String descripcion, String tipo, String unidad, int cantidad) {  
    this.id = id;  
    this.descripcion = descripcion;  
    this.tipo = tipo;  
    this.unidad = unidad;  
    this.cantidad = cantidad;  
}
```

Figura 31 - Constructor clase Material

Clases Tipo Controlador

A continuación, se presenta una de las tres clases tipo controlador.

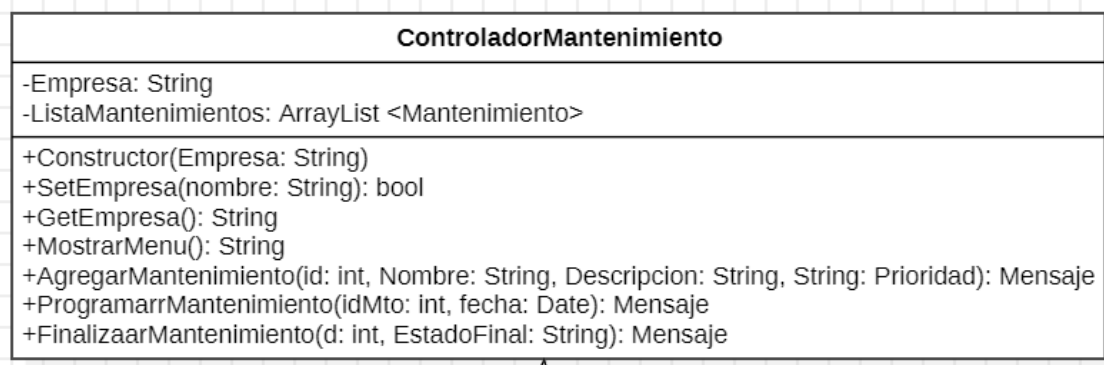
Clase ControladorMantenimiento

Figura 32 - Diagrama clase ControladorMantenimiento

La clase ControladorMantenimiento es la clase que tiene las reglas de negocio y gestiona el agregado de mantenimientos, la programación y finalización de estos. Posee dentro de sus atributos una lista con todos los mantenimientos.

Métodos de la clase.

Los Métodos centrales de la clase gestionan toda la información controlando los datos y manejando las excepciones que puedan llegar a surgir en función de que el sistema no falle y permita al usuario manejar el software con fluidez. Esta clase no se comunica directamente con el usuario ya que no tiene la capacidad de interactúa con el mismo, sino que se comunica a través de mensajes con otra clase destinada a realizar la interfaz con el usuario.

Gestión de la información y control de excepciones.

```
public Mensaje agregarMantenimiento(String cod, String nombre, String descripcion, String tipo, String prioridad) {
    Mensaje msg = new Mensaje();
    try {
        // Validar que el código del mantenimiento sea un número entero positivo
        Integer.valueOf(cod);
        // Valida que la prioridad esté dentro de los valores esperados
        if (prioridad.equals("Alta") || prioridad.equals("Media") || prioridad.equals("Baja")) {
            // Valida que el mantenimiento no exista previamente
            if (!mantenimientos.contains(new Mantenimiento(Integer.parseInt(cod), nombre, descripcion, tipo, prioridad))) {
                // Agrega el mantenimiento a la lista
                mantenimientos.add(new Mantenimiento(Integer.parseInt(cod), nombre, descripcion, tipo, prioridad));
                msg.SetMensaje(true, "Mantenimiento agregado exitosamente.");
                // Invoca a la interfaz para agregar materiales
                MenuMaterialesMto menu = new MenuMaterialesMto(mantenimientos.get(mantenimientos.size()-1));
                menu.ejecutarMenu();
            } else {
                // Captura excepción si el mantenimiento ya existe
                throw new IllegalArgumentException("Error: Ya existe un mantenimiento con este código.");
            }
        } else {
            // Setea mensaje de error si la prioridad no es válida
            msg.SetMensaje(false, "La prioridad debe ser 'Alta', 'Media' o 'Baja'.");
        }
    } catch (NumberFormatException e) {
        // Captura el error si el código del mantenimiento no es un número entero positivo
        msg.SetMensaje(false, "Error: El código del mantenimiento debe ser un número.");
    } catch (IllegalArgumentException e) {
        // Captura el error si el mantenimiento ya existe
        msg.SetMensaje(false, e.getMessage());
    }
}
```

Figura 33 - Método agregarMantenimiento

El tratamiento de los errores y excepciones se realiza a través de verificaciones y lanzamiento de excepciones con su correspondiente manejo a través de bloques try catch.

Invocación a interfaz de usuario-

Para cumplir lo definido en el Diagrama de Secuencia visto anteriormente, la clase ControladorMantenimiento invoca a la interfaz de usuario para agregar materiales.

Específicamente, se crea una instancia de la clase MenuMaterialesMto pasando al constructor de la clase interfaz la lista de materiales del mantenimiento para su gestión.

```
// Invoca a la interfaz para agregar materiales
MenuMaterialesMto menu = new MenuMaterialesMto(mantenimientos.get(mantenimientos.size()-1));
menu.ejecutarMenu();
```

Figura 34 - Invocar a la interfaz de usuario para agregar materiales

Además, en el método de ProgramarMantenimiento, se crea una instancia de la clase MenuCuadrillaMto pasando al constructor los parámetros necesarios.

```
// Invoca a la interfaz para asignar cuadrilla
MenuCuadrillaMto menuCuad = new MenuCuadrillaMto(mto);
menuCuad.ejecutarMenu();
```

Figura 35 - Invocar a la interfaz de usuario para agregar cuadrillas.

Clases Tipo Vista

A continuación, se presenta una de las tres clases tipo vista.

Clase MenuPrincipalSGM

```
private void mostrarMenu() {
    System.out.println("\n-----");
    System.out.println("\nSistema Mantenimiento " + gestor.getEmpresa() + ":");
    System.out.println("1. Agregar Nuevo Mantenimiento");
    System.out.println("2. Programar Mantenimiento");
    System.out.println("3. Finalizar Mantenimiento");
    System.out.println("4. Mostrar Mantenimientos");
    System.out.println("5. Detallar Mantenimiento");
    System.out.println("0. Salir");
    System.out.println("\n-----");
    System.out.print("Elige una opción: ");
}
```

Figura 36 - Menú que es mostrado al usuario

Esta clase es la encargada de dialogar con el usuario y enviarle los datos ingresados a la clase ControladorMantenimiento para que gestione. A su vez, espera el mensaje de la clase ControladorMantenimiento para informarle al usuario el resultado de la operación que intenta realizar.

```
private void agregarMantenimiento() {
    // Implementa la lógica para agregar un nuevo mantenimiento
    System.out.print("\033[H\033[2J");
    System.out.flush();
    System.out.println("Agregar Nuevo Mantenimiento");
    // ...
    Mensaje msg = new Mensaje();
    System.out.print("Ingrese el código del mantenimiento: ");
    String cod = new Scanner(System.in).nextLine();
    System.out.print("Ingrese el nombre del mantenimiento: ");
    String nombre = new Scanner(System.in).nextLine();
    System.out.print("Ingrese la descripción del mantenimiento: ");
    String descripcion = new Scanner(System.in).nextLine();
    System.out.print("Ingrese el tipo de mantenimiento: ");
    String tipo = new Scanner(System.in).nextLine();
    System.out.print("Ingrese la prioridad del mantenimiento: ");
    String prioridad = new Scanner(System.in).nextLine();
    msg = gestor.agregarMantenimiento(cod, nombre, descripcion, tipo, prioridad);
    if (msg.getResultado()) {
        System.out.println("SIIII -> " + msg.getMensaje());
    } else {
        System.out.println("NOOOO -> " + msg.getMensaje());
    }
}
```

Figura 37 - Método que solicita datos al usuario y le informa los resultados reportados por la clase ControladorMantenimiento

Conexión con la Base de Datos MySQL

Para diseñar los objetos que realizarán la interacción con la BD, se seguirá implementando el modelo MVC (Modelo Vista Controlador).

Como se ha detallado en el apartado anterior de diseños de clases, para cumplir con el modelo MVC, se han generado tres grupos que se detallan a continuación:

- Tipo Interfaz/Vista: son aquellas clases destinadas a realizar las interfaces con el usuario y se encargan de invocar a las clases de Tipo Controlador y pasarle la información requerida en cadenas de texto.
- Tipo Controlador: son aquellas clases que contiene las reglas de negocio e implementan la verificación de datos y control de las acciones sobre las clases de Tipo Modelo.
- Tipo Modelo: son las clases que representan a los objetos del negocio.

Las clases destinadas a interactuar con la base de datos, si bien se han agrupado en un paquete llamado “conexionBD”, pertenecen al grupo de clases de Tipo Modelo. Estas clases son las encargadas de manejar y controlar la interacción con la base de datos para mantener la persistencia de la información.

Para el prototipo inicial, se han creado tres clases que a continuación se detallan:

Clase ModeloRegistradorMantenimiento

Esta clase tiene por función agregar a la base de datos los nuevos mantenimientos a la base de datos, actualizar los datos al programar los mantenimientos y finalmente actualizar los mantenimientos en la base de datos al momento de finalizar los mantenimientos.

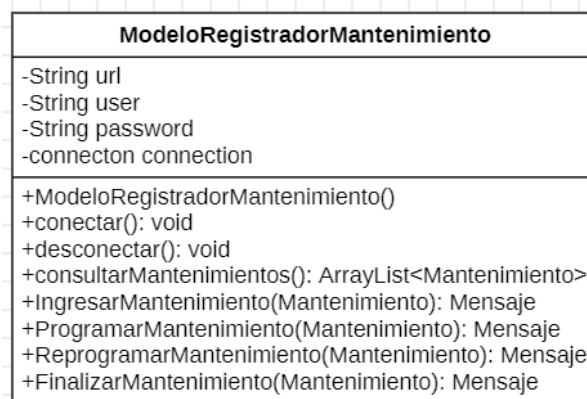


Figura 38 - Diagrama clase ModeloRegistradorMantenimiento

Esta clase establece el auto commit en falso para poder registrar la información en la base de datos; si la actualización es exitosa se realiza un commit y en caso contrario se realiza un rollback para mantener la consistencia de la información. Además, esta clase lanza excepciones de tipo SQLException para el control de las excepciones.

Atributos de la clase.

```
private final String url = "jdbc:mysql://localhost:3306/sgm?zeroDateTimeBehavior=CONVERT_TO_NULL";
private final String user = "Administrador";
private final String password = "@dminS212024";
private Connection connection;
```

Figura 39 - Atributos clase ModeloRegistradorMantenimiento

En el método conectar, se establece la conexión con la base de datos.

```
// Método para conectar a la BD
public void conectar() throws SQLException {
    try {
        // Establecer la conexión

        connection = DriverManager.getConnection(url, user, password);
        connection.setAutoCommit(false);

    } finally {

    }
}
```

Figura 40 - Método Conectar clase ModeloRegistradorMantenimiento

El método consultarMantenimientos, que lanza excepciones tipo SQLException, se conecta a la base de datos y realiza una consulta para determinar todos los mantenimientos existentes en la base de datos.

Para ello utiliza la consulta “select * from mantenimientos”, y el resultado la almacena en un ResultSet.

Luego, mediante un ciclo, comienza la carga de los mantenimientos en un ArrayList.

```
public ArrayList<Mantenimiento> ConsultarMantenimientos() throws SQLException {
    ArrayList<Mantenimiento> listaMantenimientos = new ArrayList<>();

    java.sql.Statement sqlMto = null;
    java.sql.PreparedStatement sqlMat = null;
    java.sql.PreparedStatement sqlCuad = null;

    try {
        this.conectar();

        String sql = "SELECT * FROM mantenimiento";
        sqlMto = connection.createStatement();
        java.sql.ResultSet Mtos = sqlMto.executeQuery(sql);
        while (Mtos.next()) {
            Mantenimiento mantenimiento = new Mantenimiento(Mtos.getInt("ID"), Mtos.getString("Nombre"), Mtos.getString("Desc
            mantenimiento.setEstado(Mtos.getString("Estado"));
            mantenimiento.setFechaProgramado(Mtos.getDate("FechaProgramado"));
            mantenimiento.setHorasTraslado(Mtos.getInt("HorasTraslado"));
        }
    }
}
```

Figura 41 - Primera porción de código del método ConsultarMantenimientos

Dentro del mismo ciclo, comienza la carga de los materiales asociados a cada mantenimiento realizando una consulta que integra información de las tablas relacionadas Mantenimiento, Materiales_mantenimientos y Materiales. A continuación realiza un ciclo para agregar a cada Mantenimiento los materiales asociados.

```
// Cargar los materiales del mantenimiento
String sqlmat = ""
        select mat.ID_Mat , m.Descripcion, m.TipoMaterial , m.Unidad , mat.Cantidad
        from materiales_mantenimiento as mat, mantenimiento as mto, materiales as m
        where mto.ID = mat.ID_Mto
        and mat.ID_Mat = m.IDMat
        and mto.ID = ?""";

sqlMat = connection.prepareStatement(sqlmat);
sqlMat.setInt(1, Mtos.getInt("ID"));
java.sql.ResultSet Mat = sqlMat.executeQuery();
ArrayList<Material> materiales = new ArrayList<>();
materiales = mantenimiento.getMaterialesMto();
while (Mat.next()) {
    materiales.add(new Material(Mat.getString("ID_Mat"), Mat.getString("Descripcion"), Mat.getString("TipoMaterial")
}
```

A continuación, realiza la carga de las cuadrillas asociadas a los mantenimientos, realizando una consulta a las tablas relacionadas Mantenimiento, Mantenimiento_cuadrillas y Cuadrillas. Finalmente agrega el mantenimiento con toda su información a una lista para finalmente retornarla.

```
String grymat = ""
        select mcuad.ID_Cuad IDCuad, cuad.Descripcion Descripcion, cuad.Responsable Responsable
        from mantenimiento mto, mantenimiento_cuadrillas mcuad, cuadrillas cuad
        where mto.ID = mcuad.ID_Mant
        and mcuad.ID_Cuad = cuad.ID_Cuad
        and mto.ID = ?""";

sqlCuad = connection.prepareStatement(grymat);
sqlCuad.setInt(1, Mtos.getInt("ID"));
java.sql.ResultSet Cuad = sqlCuad.executeQuery();
ArrayList<Cuadrilla> Cuadrillas = new ArrayList<>();
Cuadrillas = mantenimiento.getCuadrillas();
while (Cuad.next()) {
    Cuadrillas.add(new Cuadrilla (Cuad.getString("IDCuad"), Cuad.getString("Descripcion"), Cuad.getString("Responsa
}

// Agrego el mantenimiento con todos sus atributos al ArrayList
listaMantenimientos.add(mantenimiento);
}

this.desconectar();

} finally {
    if (sqlMto != null) {
        sqlMto.close();
    }
    if (sqlMat != null) {
        sqlMat.close();
    }
    if (sqlCuad != null) {
        sqlCuad.close();
    }
}
return listaMantenimientos;
```

El método IngresarMantenimiento, recibe como parámetro un mantenimiento con toda su información. Primeramente inserta el mantenimiento con sus datos a la base de datos mediante una sentencia insert.

```
public Mensaje IngresarMantenimiento (Mantenimiento mto) throws SQLException {
    Mensaje msg = new Mensaje();
    msg.SetMensaje(true, "Mantenimiento agregado!");

    java.sql.PreparedStatement sqlMto = null;
    java.sql.PreparedStatement sqlMat = null;

    try {
        this.conectar();
        // insertar mantenimiento
        String sqlmto = ""
            + "insert into mantenimiento (ID,Nombre,Descripcion,Estado,Prioridad,TipoMantenimiento)"
            + "values (?, ?, ?, ?, ?, ?)"";

        sqlMto = connection.prepareStatement(sqlmto);
        sqlMto.setInt(1, mto.getId());
        sqlMto.setString(2, mto.getNombre());
        sqlMto.setString(3, mto.getDescripcion());
        sqlMto.setString(4, mto.getEstado());
        sqlMto.setString(5, mto.getPrioridad());
        sqlMto.setString(6, mto.getTipo());
        sqlMto.executeUpdate();
    }
```

A continuación, recorre la lista de materiales asociados al mantenimiento para realizar los insert correspondientes.

```
// insertar materiales
String sqlmat = ""
    + "insert into materiales_mantenimiento (ID_Mto, ID_Mat, Cantidad)"
    + "values (?, ?, ?)"";

sqlMat = connection.prepareStatement(sqlmat);
sqlMat.setInt(1, mto.getId());

for (Material material : mto.getMaterialesMto()) {
    sqlMat.setString(2, material.getId());
    sqlMat.setInt(3, material.getCantidad());
    sqlMat.addBatch();
}

sqlMat.executeBatch();
```

Finalmente si la información fue correctamente guardada realiza el commit, caso contrario captura la excepción y realiza el rollback.

```
connection.commit();
this.desconectar();

}
catch (SQLException e) {
    msg.SetMensaje(false, "Error: No fue posible agregar el mantenimiento. " + e.getMessage());
    connection.rollback();
}
```

El método ProgramarMatneinimiento, realiza la actualización de los datos particulares del mantenimiento previamente ingresado mediante la sentencia update.

```
public Mensaje ProgramarMantenimiento (Mantenimiento mto) throws SQLException {
    Mensaje msg = new Mensaje();
    msg.SetMensaje(true, "Mantenimiento Programado!");

    java.sql.PreparedStatement sqlMto = null;
    java.sql.PreparedStatement sqlCuad = null;

    try {
        this.conectar();

        // actualizar mantenimiento
        String sqlmto = ""
            + "UPDATE mantenimiento"
            + "SET FechaProgramado = ?, HorasTraslado = ?, Estado = ?"
            + "WHERE ID = ?"";

        sqlMto = connection.prepareStatement(sqlmto);
        sqlMto.setDate(1, new java.sql.Date(mto.getFechaProgramado().getTime()));
        sqlMto.setInt(2, mto.getHorasTraslado());
        sqlMto.setString(3, mto.getEstado());
        sqlMto.setInt(4, mto.getId());

        sqlMto.executeUpdate();
    }
```

Luego, inserta en la base de datos las cuadrillas asociadas al mantenimiento.

```
// insertar cuadrillas
String sqlcuad = ""
    + "insert into mantenimiento_cuadrillas (ID_Mant, ID_Cuad)"
    + "values (?,?)"";

sqlCuad = connection.prepareStatement(sqlcuad);
sqlCuad.setInt(1, mto.getId());

for (Cuadrilla cuadrilla : mto.getCuadrillas()) {
    sqlCuad.setString(2, cuadrilla.getIdCuad());
    sqlCuad.addBatch();
}

sqlCuad.executeBatch();
```

Finalmente, si la información fue correctamente guardada realiza el commit, caso contrario el rollback.

```
connection.commit();
this.desconectar();

}
catch (SQLException e) {
    msg.SetMensaje(false, "Error: No fue posible Programar el mantenimiento. " + e.getMessage());
    connection.rollback();
}
```

El método ReprogramarMatneinimiento, realiza la actualización de los datos particulares del mantenimiento previamente ingresado mediante la sentencia update.

```
public Mensaje ReprogramarMantenimiento (Mantenimiento mto) throws SQLException {
    Mensaje msg = new Mensaje();
    msg.SetMensaje(true, "Mantenimiento Actualizado!");

    java.sql.PreparedStatement sqlMto = null;
    java.sql.PreparedStatement sqlCuad = null;

    try {
        this.conectar();

        // actualizar mantenimiento
        String sqlmto = ""
            + "UPDATE mantenimiento"
            + "SET FechaProgramado = null, HorasTraslado = ?, Estado = ?"
            + "WHERE ID = ?"";

        sqlMto = connection.prepareStatement(sqlmto);
        sqlMto.setInt(1, mto.getHorasTraslado());
        sqlMto.setString(2, mto.getEstado());
        sqlMto.setInt(3, mto.getId());

        sqlMto.executeUpdate();

    }
}
```

Luego, elimina a las cuadrillas asociadas al mantenimiento mediante una sentencia delete.

```
// borrar cuadrillas
String sqlcuad = ""
    + "delete from mantenimiento_cuadrillas"
    + "where ID_Mant = ?"";

sqlCuad = connection.prepareStatement(sqlcuad);
sqlCuad.setInt(1, mto.getId());

sqlCuad.executeUpdate();
```

Finalmente, si la información fue correctamente guardada realiza el commit, caso contrario el rollback.

```
connection.commit();
this.desconectar();

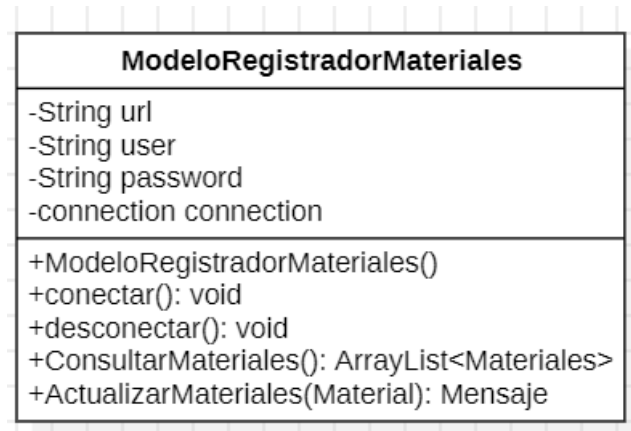
}
catch (SQLException e) {
    msg.SetMensaje(false, "Error: No fue posible Actualizar el mantenimiento. " + e.getMessage());
    connection.rollback();
}
```


El método FinalizarMatneinimiento, realiza la actualización de los datos particulares del mantenimiento previamente ingresado mediante la sentencia update. Finalmente, si la información fue correctamente guardada realiza el commit, caso contrario el rollback.

```
public Mensaje FinalizarMantenimiento (Mantenimiento mto) throws SQLException {  
    Mensaje msg = new Mensaje();  
    msg.SetMensaje(true, "Mantenimiento Actualizado!");  
  
    java.sql.PreparedStatement sqlMto = null;  
  
    try {  
        this.conectar();  
  
        // actualizar mantenimiento  
        String sqlmto = ""  
            + "UPDATE mantenimiento  
            + "SET Estado = ?  
            + "WHERE ID = ?"";  
  
        sqlMto = connection.prepareStatement(sqlmto);  
        sqlMto.setString(1,mto.getEstado());  
        sqlMto.setInt(2,mto.getId());  
  
        sqlMto.executeUpdate();  
  
        connection.commit();  
        this.desconectar();  
    }  
    catch (SQLException e) {  
        msg.SetMensaje(false, "Error: No fue posible Actualizar el mantenimiento. " + e.getMessage());  
    }  
}
```

Clase ModeloRegistradorMateriales

La clase ModeloRegistradorMateriales es la encargada de mantener actualizada la lista de materiales de stock almacenada en la tabla materiales.



Esta clase establece el auto commit en falso para poder registrar la información en la base de datos; si la actualización es exitosa se realiza un commit y en caso contrario se realiza un rollback para mantener la consistencia de la información. Además, esta clase lanza excepciones de tipo `SQLException` para el control de las excepciones.

Atributos de la clase.

```
private final String url = "jdbc:mysql://localhost:3306/sgm?zeroDateTimeBehavior=CONVERT_TO_NULL";
private final String user = "Administrador";
private final String password = "@dminS212024";
private Connection connection;
```

En el método conectar, se establece la conexión con la base de datos.

```
// Método para conectar a la BD
public void conectar() throws SQLException {
    try {
        // Establecer la conexión

        connection = DriverManager.getConnection(url, user, password);
        connection.setAutoCommit(false);

    } finally {

    }
}
```

El método `consultarMateriales`, que lanza excepciones tipo `SQLException`, se conecta a la base de datos y realiza una consulta para determinar todos los materiales existentes en stock almacenados en la base de datos.

Para ello utiliza la consulta “`select * from materiales`”, y el resultado la almacena en un `ResultSet`. Luego, mediante un ciclo, comienza la carga de los materiales en un `ArrayList`.

```

public ArrayList<Material> ConsultarMateriales() throws SQLException {
    ArrayList<Material> listaMateriales = new ArrayList<>();

    java.sql.Statement sqlMat = null;

    try {
        this.conectar();

        // Cargar los materiales del stock
        String sqlmat = "select * from materiales";

        sqlMat = connection.createStatement();
        java.sql.ResultSet Mat = sqlMat.executeQuery(sqlmat);

        while (Mat.next()) {
            listaMateriales.add(new Material(Mat.getString("IDMat"), Mat.getString("Descripcion"), Mat.getString("TipoMat")
        )

        this.desconectar();
    } finally {
        if (sqlMat != null) {
            sqlMat.close();
        }
    }
    return listaMateriales;
}

```

El método ActualizarMateriaels, realiza la actualización de las cantidades de materiales de stock a medida que son utilizados en cada mantenimiento.

```

public Mensaje ActualizarMaterialStock(Material material) throws SQLException {
    Mensaje msg = new Mensaje();
    java.sql.PreparedStatement sqlMat = null;

    try {
        this.conectar();
        // Cargar los materiales del stock
        String sqlmat = ""
            + "UPDATE materiales"
            + "SET Cantidad = ?"
            + "WHERE IDMat = ?"";

        sqlMat = connection.prepareStatement(sqlmat);
        sqlMat.setInt(1, material.getCantidad());
        sqlMat.setString(2, material.getId());

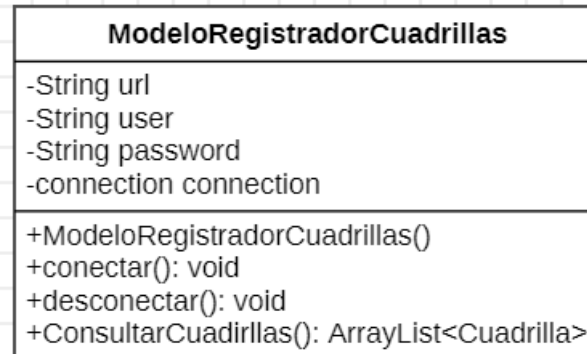
        sqlMat.executeUpdate();
        connection.commit();
        this.desconectar();
        msg.SetMensaje(true, "Material actualizado con éxito.");
    }
    catch (SQLException e) {
        msg.SetMensaje(false, "Error al actualizar stock. " + e.getLocalizedMessage());
    }
    catch (Exception e){
        msg.SetMensaje(false, e.getMessage());
    }
    finally {
        if (sqlMat != null) {
            sqlMat.close();
        }
    }

    return msg;
}

```

Clase ModeloRegistradorCuadrillas

La clase ModeloRegistradorCuadrillas es la encargada de mantener actualizada la lista de cuadrillas disponibles almacenada en la tabla cuadrillas.



Esta clase establece el auto commit en falso para poder registrar la información en la base de datos; si la actualización es exitosa se realiza un commit y en caso contrario se realiza un rollback para mantener la consistencia de la información. Además, esta clase lanza excepciones de tipo `SQLException` para el control de las excepciones.

Atributos de la clase.

```
private final String url = "jdbc:mysql://localhost:3306/sgm?zeroDateTimeBehavior=CONVERT_TO_NULL";
private final String user = "Administrador";
private final String password = "@dminS212024";
private Connection connection;
```

En el método conectar, se establece la conexión con la base de datos.

```
// Método para conectar a la BD
public void conectar() throws SQLException {
    try {
        // Establecer la conexión

        connection = DriverManager.getConnection(url, user, password);
        connection.setAutoCommit(false);

    } finally {

    }
}
```

El método consultarCuadrillas, que lanza excepciones tipo `SQLException`, se conecta a la base de datos y realiza una consulta para determinar todas las cuadrillas disponibles almacenados en la base de datos.

Para ello utiliza la consulta "select * from cuadrillas", y el resultado la almacena en un `ResultSet`. Luego, mediante un ciclo, comienza la carga de las cuadrillas en un `ArrayList`.

```
public ArrayList<Cuadrilla> ConsultarCuadrillas() throws SQLException {
    ArrayList <Cuadrilla> listaCuadrillas = new ArrayList<>();

    java.sql.Statement sqlCuad = null;

    try {
        this.conectar();

        // Cargar los materiales del stock
        String sqlcuad = "select * from cuadrillas";

        sqlCuad = connection.createStatement();
        java.sql.ResultSet Cuad = sqlCuad.executeQuery(sqlcuad);

        while (Cuad.next()) {
            listaCuadrillas.add(new Cuadrilla(Cuad.getString("ID_Cuad"), Cuad.getString("Descripcion"), Cuad.getString("Re
        })

        this.desconectar();

    } finally {
        if (sqlCuad != null) {
            sqlCuad.close();
        }
    }

    return listaCuadrillas;
}
```