

UNIVERSIDAD ABIERTA INTERAMERICANA

INGENIERIA EN SISTEMAS INFORMATICOS

SEMINARIO DE APLICACIÓN PROFESIONAL

COMISION 5 A NOCHE

PROYECTO: “GEMOL”



AUTOR: ALICE, MARTIN MIGUEL.

PROFESORES: ING. ALEJANDRO SARTORIO.

ING. ALEJANDRO DE CASTRO.

ING. MARCELO VAQUERO.

AGOSTO 2018

**INDICE**

[10 Aspectos Descriptivos de la Solución Tecnológica 5](#_Toc520920739)

[10.1 Project Charter 5](#_Toc520920740)

[10.2 Scope Statement 9](#_Toc520920741)

[10.3 Registro de Stakeholders 11](#_Toc520920742)

[10.4 Introducción 13](#_Toc520920743)

[10.4.1 Propósito 14](#_Toc520920744)

[10.4.2 Alcance 14](#_Toc520920745)

[10.4.3 Definiciones, Acrónimos, y Abreviaciones 14](#_Toc520920746)

[10.5 Descripción de las partes interesadas y usuarios 15](#_Toc520920747)

[10.5.1 Resumen de las partes interesadas 15](#_Toc520920748)

[10.5.2 Entorno de Usuario 15](#_Toc520920749)

[10.5.3 Perfil de las partes interesadas 16](#_Toc520920750)

[10.6 Descripción Global del Producto 16](#_Toc520920751)

[10.6.1 Perspectiva del Producto 16](#_Toc520920752)

[10.6.2 Resumen de Características 16](#_Toc520920753)

[10.6.3 Suposiciones y Dependencias 16](#_Toc520920754)

[10.7 Descripción Detallada del Producto 17](#_Toc520920755)

[10.7.1 Estándares Aplicables 17](#_Toc520920756)

[10.7.2 Requisitos de Sistema 20](#_Toc520920757)

[10.7.3 Requisitos de Desempeño del sistema 20](#_Toc520920758)

[10.7.4 Requisitos de Entorno 20](#_Toc520920759)

[10.8 Requisitos de Documentación 20](#_Toc520920760)

[10.9 Guías de Instalación, Configuración y Fichero Léame 20](#_Toc520920761)

[10.10 Casos de Uso – Prototipos de Interfaz de Usuario 20](#_Toc520920762)

[10.11 Especificaciones de Casos de Uso 23](#_Toc520920763)

[10.12 Diagramas de Casos de Uso 29](#_Toc520920776)

[10.13 Modelado del Dominio 31](#_Toc520920779)

[10.14 Diagrama de Robustez 33](#_Toc520920782)

[10.15 Diagrama de Secuencia 36](#_Toc520920785)

[10.16 Diagrama de Clases 39](#_Toc520920788)

[10.17 Modelo de Datos 41](#_Toc520920791)

[10.18 Seguridad y Auditoría 42](#_Toc520920792)

[10.19 Pruebas Unitarias 43](#_Toc520920793)

[10.19.1 Prueba Caja Negra 43](#_Toc520920794)

[10.19.2 Prueba Caja Blanca 46](#_Toc520920795)

[10.20 Manual de Usuario 58](#_Toc520920796)

[10.20.1 Manual de Uso 58](#_Toc520920797)

[10.20.2 Manual de instalación 59](#_Toc520920798)

[10.21 Patrones de Diseño 59](#_Toc520920799)

[ANEXOS 61](#_Toc520920800)

1. Aspectos Descriptivos de la Solución Tecnológica
   1. Project Charter

|  |  |
| --- | --- |
| **NOMBRE DEL PROYECTO** | **SIGLAS DEL PROYECTO** |
| **Gestión de Mantenimiento On-Line** | **GeMOL** |
| **DESCRIPCIÓN DEL PROYECTO:** *QUÉ, QUIÉN, CÓMO, CUÁNDO Y DÓNDE? RESUMEN* | |
| GeMOL es un sistema el cual mediante una plataforma web brinda un servicio de gestión y planificación de tareas y recursos para el mantenimiento de equipamientos industriales. Está orientado a pequeñas y medianas empresas de distintos rubros como ser: construcción, vialidad, agroindustrial, metalmecánicas, municipios, navales, etc. La segmentación geográfica incluye la localidad de Rosario y alrededores. | |

|  |
| --- |
| **DEFINICIÓN DE REQUISITOS DEL PROYECTO:** |
| 1. El proyecto que dar un servicio para planificar tareas de mantenimiento en función de una rutina, un grupo de personas, una cantidad de tiempo, fechas, frecuencia de tareas, estrategias de cada frecuencia si es por cantidad de tiempo de uso, distancia recorrida o solo tiempo calendario transcurrido. No se incluye en este proyecto el módulo de materiales y repuestos. 2. Los activos son los equipos o máquinas propias de la empresa a los que se les hace el mantenimiento. Los mismos llevan un nombre, descripción específica y técnica con datos como marca, modelo, número de serie, estado actual, vida útil o garantías, valoración (con posibilidad de agregar mas datos según la necesidad del usuario). 3. Los activos forman parte de un árbol de activos. Esto significa que un activo, puede tener hijos y nietos respectivamente. Por ejemplo, un auto es el activo principal, como activo hijo puede ser el motor, y como activo nieto puede ser el burro de arranque, el cual también es hijo del motor. Es fundamental mantener el criterio de que todo aquello que es viable ser reparado se considera un activo, no así por ejemplo, tornillos, rodamientos, sellos, accesorios, etc. 4. Los equipos de trabajo con una lista de personas que trabajan para la empresa y ejecutan la mano de obra de mantenimiento. Esas personas tienes que estar registradas con nombre, apellido, perfil de mano de obra (eléctrico, mecánico, ayudante, etc, con posibilidad de cambio en un futuro) y costo por hora. 5. Existe algo que se denomina aviso de mantenimiento y es un documento que contiene información acerca de un activo que necesita ser atendido. Cuando se detecta un problema o falla se crea un aviso de mantenimiento para que el supervisor a cargo se entere del problema y quede registrado en el sistema. 6. Existe algo que se llama orden de trabajo y es un documento que contiene una serie de tareas de mantenimiento que deben realizarse sobre un activo. Ese documento indica qué tareas se tienen que hacer sobre un activo, que personas lo realizarán, y de forma muy escueta se permite colocar información de repuestos relevantes a la tarea con su costo asociado. También se indica en la orden cuánto tiempo tomará el trabajo y cuando será realizado. 7. Una orden de trabajo puede crearse a partir de un aviso de mantenimiento y estar asociados para cuando una orden de trabajo culmine, el aviso también. 8. La planificación de tareas es un documento que indica una lista de tareas de mantenimiento que se le van a realizar a los activos (máquinas). Esas tareas son realizadas según una frecuencia determinada, la misma puede ser por unidad de tiempo, por volumen de producción, distancias o otras unidades de medida que brinden flexibilidad. La unidad de medida de los activos debe ser registrada en el sistema de forma manual para que la lógica funcione. Según la información registrada y la medición tomada del activo, el sistema puede remitir órdenes de trabajo o avisos de mantenimiento según elija el usuario. Desde el punto de vista de los procesos estandarizados de mantenimiento, la planificación representa a los tipos de mantenimiento preventivos y predictivos (esto también puede cambiar según la necesidad del cliente). 9. El sistema tiene que reportar métricas (que a futuro pueden cambiar según la necesidad del usuario). Inicialmente se calcularán internamente las siguientes métricas:    1. Tiempo medio entre fallas: es el tiempo promedio que existe entre una falla y la siguiente, por activo.    2. Costo de mantenimiento: es el costo de mantenimientos realizados a cada activo por mes.    3. Tiempo de respuesta: es el tiempo que existe desde la creación de un aviso de mantenimiento correctivo y el cierre del mismo ya sea a través del cierre de una orden de trabajo o no. |

|  |  |  |
| --- | --- | --- |
| **OBJETIVOS DEL PROYECTO:** *METAS HACIA LAS CUALES SE DEBE DIRIGIR EL TRABAJO DEL PROYECTO EN TÉRMINOS DE*  *LA TRIPLE RESTRICCIÓN.* | | |
| ***CONCEPTO*** | ***OBJETIVOS*** | ***CRITERIO DE ÉXITO*** |
| ***1. ALCANCE*** | Implementar el sistema desarrollado de forma que cumpla con los requisitos antes descritos. | Es fundamental que el sistema genere y reporte las métricas que necesita el usuario. |
| ***2. TIEMPO*** | Fin del proyecto: 24 de Julio de 2018 | Avanzar en cada etapa del proyecto de acuerdo a lo planificado |
| ***3. COSTO*** | No menos de 6 horas semanales hasta la finalización del proyecto. | Establecer un horario fijo cada dia y establecer un objetivo diario |

|  |
| --- |
| **FINALIDAD DEL PROYECTO:** *FIN ÚLTIMO, PROPÓSITO GENERAL, U OBJETIVO DE NIVEL SUPERIOR POR EL CUAL SE EJECUTA EL*  *PROYECTO. ENLACE CON PROGRAMAS, PORTAFOLIOS, O ESTRATEGIAS DE LA ORGANIZACIÓN.* |
| El objetivo general del proyecto es brindar una herramienta para lograr eficiencia en el mantenimiento de equipos industriales. Este objetivo está alineado con los objetivos principales de cualquier organización referidos a la rentabilidad, reducción de costos, mejora de la productividad y eficiencia operacional. |

|  |  |  |
| --- | --- | --- |
| **DESIGNACIÓN DEL PROJECT MANAGER DEL PROYECTO** | | |
| ***NOMBRE*** | Martin Alice | ***NIVELES DE AUTORIDAD*** |
| ***REPORTA A*** | Martin Alice | Único líder y ejecutor del proyecto. |
| ***SUPERVISA A*** | Martin Alice |

|  |  |
| --- | --- |
| **CRONOGRAMA DE HITOS DEL PROYECTO** | |
| ***HITO O EVENTO SIGNIFICATIVO*** | ***FECHA PROGRAMADA*** |
| Inicio del Proyecto. | 3 de Abril de 2018 |
| Análisis de requisitos | 10 de Abril de 2018 |
| Diseño de casos de uso core | 29 de Mayo de 2018 |
| Desarrollo | 19 de Junio de 2018 |
| Implementación | 26 de Junio de 2018 |
| Testing | 3 de Julio de 2018 |
| Fin del Proyecto. | 24 de Julio de 2018 |

|  |  |
| --- | --- |
| **ORGANIZACIONES O GRUPOS ORGANIZACIONALES QUE INTERVIENEN EN EL PROYECTO** | |
| ***ORGANIZACIÓN O GRUPO ORGANIZACIONAL*** | ***ROL QUE DESEMPEÑA*** |
| Universidad Abrierta Interamericana | Seguimiento y Evaluación |
| La empresa | Apoyo financiero y logístico |

|  |
| --- |
| **PRINCIPALES AMENAZAS DEL PROYECTO** *(RIESGOS NEGATIVOS)* |
| Los competidores que ofrecen aplicaciones de escritorio con las mismas características pueden pueden estar pensando en implementar una herramienta web similar a GeMOL.  Algunos clientes no demuestran interés en mejorar sus procesos de mantenimiento de equipos y otros trabajan solo a la rotura sin pensar en planificar o invertir en gestión. |

|  |
| --- |
| **PRINCIPALES OPORTUNIDADES DEL PROYECTO** *(RIESGOS POSITIVOS)* |
| El proyecto se distribuye de forma web, esto permite que el cliente acceda desde cualquier lugar y desde cualquier computadora.  Las prestaciones son las justas y necesarias, de fácil comprensión.  Evita costos de infraestructura de red por parte del cliente. |

|  |
| --- |
| **ASUNCIONES** |
| Se dispone del inmueble y requisitos de infraestructura necesarios para comenzar con el proyecto. Incluyendo las licencias de desarrollo y base de datos. |

|  |
| --- |
| **EXCLUSIONES DEL PROYECTO:** *ENTREGABLES, PROCESOS, ÁREAS, PROCEDIMIENTOS, CARACTERÍSTICAS,REQUISITOS, FUNCIONES, ESPECIALIDADES, FASES, ETAPAS, ESPACIOS FÍSICOS, VIRTUALES, REGIONES, ETC., QUE SONEXCLUSIONES CONOCIDAS Y NO SERÁN ABORDADAS POR EL PROYECTO, Y QUE POR LO TANTO DEBEN ESTAR CLARAMENTE ESTABLECIDAS PARA EVITAR INCORRECTAS INTERPRETACIONES ENTRE LOS TAKEHOLDERS DEL PROYECTO.* |
| En el proyecto no está incluido el módulo de alamacen e inventario. Los costos que se procesan en el sistema son relacionados a recursos humanos y tiempos.  Proveedores o empresas de servicio tercerizados no están incluidos en la lógica del sistema |

|  |  |
| --- | --- |
| **RESTRICCIONES DEL PROYECTO:** *FACTORES QUE LIMITAN EL RENDIMIENTO DEL PROYECTO, EL RENDIMIENTO DE UN PROCESO DEL PROYECTO, O LAS OPCIONES DE PLANIFICACIÓN DEL PROYECTO. PUEDEN APLICAR A LOS OBJETIVOS DEL PROYECTO O A LOS RECURSOS QUE SE EMPLEA EN EL PROYECTO.* | |
| ***INTERNOS A LA ORGANIZACIÓN*** | ***AMBIENTALES O EXTERNOS A LA ORGANIZACIÓN*** |
| Seguimiento e imputación de datos al sistema (sobre todo al inicio). Es necesario que el sistema cuente con datos de personas y equipos esenciales para el funcionamiento. | Proveedores y servicios tercerizados no implementado |
| Modulo de almacen no implementado |  |
| NIvel de educación informática por parte del cliente |  |

|  |  |
| --- | --- |
| **SUPUESTOS DEL PROYECTO:** *FACTORES QUE PARA PROPÓSITOS DE LA PLANIFICACIÓN DEL PROYECTO SE CONSIDERAN VERDADEROS, REALES O CIERTOS.* | |
| ***INTERNOS A LA ORGANIZACIÓN*** | ***AMBIENTALES O EXTERNOS A LA ORGANIZACIÓN*** |
| El cliente conoce y tiene amplia experiencia el proceso propio de su organización |  |

|  |  |  |
| --- | --- | --- |
| **PRESUPUESTO PRELIMINAR DEL PROYECTO:** | | |
| ***CONCEPTO*** | | ***MONTO ($)*** |
| ***1. PERSONAL*** | Capital humano para el diseño y desarrollo del sistema | $160.000 |
|  | $ |
| ***2. MATERIALES*** | materiales e insumos. bienes de consumo | $ 50.000 |
| ***3. MAQUINAS*** | Infraestructura de red e instalaciones | $120.000 |
| ***4. OTROS COSTOS*** | bienes de capital | $ 100.000 |
| ***TOTAL LÍNEA BASE*** | | **$ 430.000** |
| ***5. RESERVA DE CONTINGENCIA*** | retrasos e incontingencias | $ 80.000 |
| ***6. RESERVA DE GESTIÓN*** | gastos administrativos varios | $ 50.000 |
| ***TOTAL PRESUPUESTO*** | | **$ 560.000** |

|  |  |  |  |
| --- | --- | --- | --- |
| **SPONSOR QUE AUTORIZA EL PROYECTO** | | | |
| ***NOMBRE*** | ***EMPRESA*** | ***CARGO*** | ***FECHA*** |
| UAI | Universidad Abierta Interamericana | Evaluador | 19/04/2018 |

* 1. Scope Statement

|  |  |
| --- | --- |
| **DESCRIPCIÓN DEL ALCANCE DEL PRODUCTO** | |
| ***REQUISITOS QUE INCLUYE:*** *FUNCIONALIDAD QUE EL PRODUCTO POSEERA. CONDICIONES O CAPACIDADES QUE DEBE POSEER O SATISFACER EL PRODUCTO PARA CUMPLIR CON CONTRATOS, NORMAS, ESPECIFICACIONES, U OTROS DOCUMENTOS FORMALMENTE IMPUESTOS.* | ***REQUISITOS QUE EXCLUYE:*** *FUNCIONALIDAD QUE EL PRODUCTO NO POSEERA.* |
| Planificar tareas con frecuencia, rutinas, entidades afectadas, fechas de inicio y finalización, posibilidad de retener la planificación en el tiempo para luego ser disparada cuando el cliente desee. | Inventarios, control de stock y almacenes |
| Familia de activos hasta 3 niveles jerárquicos | Proveedores |
| Manejo de equipos de trabajo (personas) con costos según perfil y categoría de mano de obra | Servicios tercerizados |
| Documentos de registro de tareas, como avisos de fallas y órdenes de trabajos. |  |
| Registro de estados de equipos en unidad de medida (kilometraje, volumen, tiempo de operación, etc) para establecer la lógica del sistema de planificación. |  |
| Asociación entre avisos y órdenes de trabajo. |  |
| Aporte de métricas de control de costos, tiempos de respuesta, adherencia a planificación, resolución de fallas. |  |

|  |  |
| --- | --- |
| **CRITERIOS DE ACEPTACIÓN DEL PRODUCTO:** *ESPECIFICACIONES O REQUISITOS DE RENDIMIENTO, FUNCIONALIDAD, ETC., QUE DEBEN CUMPLIRSE PARA QUE SE ACEPTE EL PRODUCTO.* | |
| ***CONCEPTOS*** | ***CRITERIOS DE ACEPTACIÓN*** |
| Información del cliente dentro del sistema | Indispensable crear y cargar al sistema la familia de activos (padre, hijo y nieto) y las asociaciones entre los mismo |
| Información de los usuarios y perfiles | Definir perfiles para evitar la superposición de funciones y responsabilidades de acuerdo al proceso de mantenimiento. (El usuario que planifica no puede ejecutar) |
| Información de equipos de trabajo | Grupo de personas que trabajan para el mantenimiento con sus perfiles, categorías, rango horario y costo hora hombre. |
| Análisis de métricas básicas. | Implementación de métricas como: tiempo medio entre fallas, backlog de mantenimiento, costos de mantenimiento por equipo, etc. |
| Análisis de métricas propuestas por el cliente | Definir con el cliente qué métricas utilizar y sirven a la toma de decisiones del área |

|  |  |
| --- | --- |
| **ENTREGABLES DEL PROYECTO:** *PRODUCTOS ENTREGABLES INTERMEDIOS Y FINALES QUE SE GENERARÁN EN CADA FASE DEL PROYECTO.* | |
| ***FASE DEL PROYECTO*** | ***PRODUCTOS ENTREGABLES*** |
| Análisis de requisitos |  |
| Diseño estructural | Interfaz de usuario |
| Desarrollo |  |
| Implementación | Testing |
| Ajustes | Entrega |

* 1. Registro de Stakeholders

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IDENTIFICACION | | | | EVALUACION | | | CLASIFICACION | |
| NOMBRE | ORGANIZACIÓN | LOCALIZACION | ROL EN EL PROYECTO | EXPECTATIVAS PRINCIPALES | INFLUENCIA POTENCIAL | FASE DE MAYOR INTERES | INTERNO / EXTERNO | APOYO / NEUTRAL / OPOSITOR |
| Gerente General de la empresa | Gerencia | Empresa | Gestión y gerenciamiento de la empresa en el nivel superior.  Toma de decisiones. Definición de objetivos a largo plazo | Supervisión de la implementación del proyecto | Si | Implementación/Entrega | Interno | Apoyo |
| Supervisor/Jefe de Mantenimiento | Supervisión | Empresa | Supervisión y manejo del personal que realiza mantenimiento de equipos | Supervisión de la implementación del proyecto  Key User | Si | Implementación/Entrega/Mantenimiento | Interno | Apoyo |
| Planificador de Mantenimiento | Departamento de Mantenimiento | Empresa | Planner de Mantenimiento en equipos. Coordinador de tareas | Supervisión de la implementación del proyecto  Key User | Si | Implementación/Entrega/Mantenimiento | Interno | Apoyo |
| Ejecutantes de Mantenimiento | Departamento de Mantenimiento | Empresa | Grupo de trabajo que ejecuta tareas de mantenimiento | Supervisión de la implementación del proyecto  Key User | No | Implementación/Entrega/Mantenimiento | Interno | Apoyo |
| Operaciones de equipos | Cliente Interno | Empresa | Operador de los equipos mantenidos | Cliente interno | Si | Implementación/Entrega/Mantenimiento | Interno | Opositor |
| Servicios tercerizados | Departamento de Mantenimiento (terceros) | Empresa | Servicio de mano de obra contratada | Proveedor interno | No | Ninguna | Interno | Neutral |

* 2. Introducción

GEMOL es un proyecto que brinda soluciones a la gestión y planificación del mantenimiento de máquinas industriales y maquinaria pesada como cintas transportadoras, grúa y vehículos pesados utilizados en diferentes industrias, aportando al negocio todas las funciones en una sola aplicación web fácil de utilizar y de rápida registración.

La implementación de funcionalidades como planificar actividades preventivas y/o predictivas, monitorear situaciones correctivas y administrar recursos tan importantes como lo son el tiempo y el capital humano, está directamente relacionado a los costos globales de cualquier organización.

GEMOL es un B2B que se enfoca estratégicamente en insertar a aquellas organizaciones destinadas al mantenimiento industrial de maquinarias e instalaciones, a sumergirse en el mundo de la gestión y administración de su propio negocio de una manera sencilla a través de Internet.

La principal ventaja competitiva que presenta este proyecto, lo cual lo hace único a este producto, es que no existe actualmente en el mercado de la región sur de la provincia de Santa Fé, una plataforma web que brinde soluciones de gestión y planificación de mantenimiento tanto preventivo como predictivo.

Existen competidores que brindan servicios de similares características con la diferencia de que estas son aplicaciones de escritorio que requieren de procesos de instalación y adaptación demandantes de mucho tiempo y recursos para el cliente. Ello se traduce en costos altísimos. GEMOL busca diferenciarse de estos competidores aplicando los mismos beneficios para el cliente directamente desde una plataforma web, sin la necesidad de complejas instalaciones de software, costosas infraestructuras de redes, o capacitaciones extensas para los usuarios de la aplicación.

Al leer el mercado de la región y ver que no hay herramientas web para este rubro se observa una buena oportunidad para poder brindar una solución y facilitarle la gestión de los recursos a los clientes del rubro industrial, agro, portuarios, viales, navieras, construcción y otras.

El proyecto le permite al cliente asociarse con esta solución tecnológica pensando en que cuanto mayor tiempo de uso tiene el sistema, más experiencia y datos este contendrá, lo cual sirve para una mejor toma de decisiones por parte del cliente y gracias al proceso de información y reportes de métricas que el sistema pone a disposición.

Existen numerosos competidores que sugieren importantes y llamativos diseños web e incluso aplicaciones novedosas para el mundo actual. La propuesta se diferencia de estos en que brinda un servicio que ningún otro competidor en el mercado de la región lo hace. El masivo consumo de Internet abre un gran abanico de oportunidades para encontrar a clientes meta en busca de una solución a sus problemas de gestión y administración.

El proyecto le permite optimizar al cliente sus recursos más relevantes como lo son el tiempo, el capital humano y el dinero, los cuales se van a ver aprovechados luego de implementada esta solución tecnológica.

* + 1. Propósito

GEMOL es un e-business que brinda soluciones a la gestión y administración del mantenimiento de máquinas industriales como: grúas de cargas a granel, cintas transportadoras, vehículos de transporte pesado, equipos para la construcción, vialidad y todo tipo de máquinas agrícolas. Envuelve un universo de activos de mantenimientos que se utilizan en diferentes ámbitos operacionales: puertos, aeropuertos, construcción, transporte, agro y metalúrgicas. Aporta al negocio del mantenimiento de equipos, todas las funciones en una aplicación web, fácil de utilizar para el cliente, dinámica, de rápida registración en las bases de datos.

La implementación de funcionalidades como planificar actividades preventivas y predictivas, monitorear situaciones correctivas y administrar recursos tan importantes como lo son el tiempo y el capital humano, está directamente relacionado a los costos globales de cualquier organización.

* + 1. Alcance

El proyecto se encuadra en el modelo de negocios B2B (Business to Business) a raíz de que las funcionalidades principales del sistema son un beneficio y un servicio orientado a brindar valor para aquellas empresas que realizan el mantenimiento industrial de sus activos o recursos utilizados para llevar a cabo sus operaciones y/o negocios.

GEMOL es un negocio que brinda soluciones tecnologías a otros negocios de la rama industrial que no cuentan con un sistema de gestión para el mantenimiento y administración de sus recursos más relevantes o su actual sistema no cubre las necesidades y objetivos de la organización. La problemática del mercado analizado es que muchas empresas no realizan este tipo de gestiones lo cual genera costos que a simple vista no logran percibirse, y son costos que pueden verse reducidos solamente con una administración adecuada. Ante esta problemática, GEMOL permite a las empresas, tener una herramienta para facilitar esa gestión y administración de recursos.

El sistema contempla tanto el registro de equipos denominados como Activos del cliente, como la organización de sus tareas de mantenimiento.

* + 1. Definiciones, Acrónimos, y Abreviaciones

Los activos son los equipos, las maquinas o las instalaciones que son parte del patrimonio del cliente y que son afectadas al mantenimiento que administra el sistema. El punto de Medición es un parámetro que representa el estado de uso de un equipo o activo.

El disparo es un proceso mediante el cual un plan de mantenimiento es lanzado para ejecutar su función principal, generar tareas de mantenimiento de forma automática.

* 1. Descripción de las partes interesadas y usuarios
     1. Resumen de las partes interesadas

|  |  |  |
| --- | --- | --- |
| *Puesto* | *Descripción* | *Responsabilidad* |
| Gerente de Empresa Cliente | Gerencia | Gestión y gerenciamiento de la empresa en el nivel superior.  Toma de decisiones.  Definición de objetivos a largo y mediano plazo. |
| Supervisor de Mantenimiento | Supervisión y Control del personal de Mantenimiento. | Control de personal.  Ejecución de tareas.  Seguimiento del sistema. |
| Planner de Mantenimiento | Gestión de planificación y coordinación de tareas. | Coordinar tareas de mantenimiento.  Soporte al Supervisor de Mantenimiento.  Planificación de mantenimiento.  Seguimiento del sistema. |
| Ejecutante de Mantenimiento | Ejecutar actividades de mantenimiento. | Atender los planes de mantenimiento.  Ejecución y verificación de las tareas de mantenimiento.  Seguimiento del sistema. |
| Cliente interno (Operaciones de la empresa) | Utilizador de los activos | Utilizar los activos de la empresa con responsabilidad.  Informar al área de Mantenimiento antes fallas de equipos. |

* + 1. Entorno de Usuario

Los usuarios utilizan un entorno gráfico Web mediante los navegadores de Internet Explorer, Chrome y/o Firefox sobre los cuales el usuario puede ingresar en forma segura.

Los reportes tendrán formato de salida HTML y/o Qlik View, para la ayuda en la toma de decisiones.

* + 1. Perfil de las partes interesadas

|  |  |
| --- | --- |
| *Puesto* | *Perfil* |
| Gerente de Empresa Cliente | Gerenciamiento de Empresa. Emprendedor. Nivel superior de decisiones. Conocimientos de Negocio Avanzado. Conocimiento bajo en uso de sistemas transaccionales. |
| Supervisor de Mantenimiento | Nivel básico de uso de sistema de gestión de mantenimiento. Usuario avanzando. |
| Planner de Mantenimiento | Alto nivel de uso sistemas transaccionales. Monitoreo continuo en campo con soporte de aplicaciones. Usuario avanzando. |
| Ejecutante de Mantenimiento | Nivel inferior de uso de sistema de registración de actividades. Usuario básico. |
| Cliente interno (Operaciones de la empresa) | Usuario que no utiliza el sistema. |

* 1. Descripción Global del Producto
     1. Perspectiva del Producto

GeMOL permite planificar tareas de mantenimiento, administrar horas hombre y disponibilidad de equipos. La posibilidad de obtener reportes de KPI’s permitirá obtener ventajas sobre competidores y proporcionar información de valor para los clientes internos. La gestión del negocio mediante GeMOL permitirá concentrar el esfuerzo de niveles medios en gestión y análisis de información.

* + 1. Resumen de Características

Las principales características del GeMOL son:

* Gestión de Activos (altas, bajas, modificaciones, evaluación de costos).
* Gestión de Equipos de trabajo (altas, bajas, modificaciones de datos del personal asociado al mantenimiento, costos por hora hombre y categorías).
* Gestión de Planificación de tareas con cronogramas y frecuencias de acción.
* Gestión de Avisos de mantenimiento correctivo.
* Gestión de Ordenes de trabajo para posteriores análisis.
* Reportes de indicadores.
  + 1. Suposiciones y Dependencias

Se asume que el cliente proveerá de la siguiente información:

* Árbol de activos padre e hijos.
* Grupos de trabajos.
* Categorías de trabajo.
* Costos de activos y de horas hombre.
  1. Descripción Detallada del Producto

GeMOL se implementa en una plataforma Web que consiste en dos esquemas diferenciados:

* Frontend: Interfaz web desde donde acceden los diferentes usuarios. La misma consta de una página principal a la que accede mediante un menú general a la actividad que quiera realizarse. Para validar los permisos el usuario deberá ingresar con su nombre y contraseña.

Cada perfil de usuario accede a las funciones del sistema según sus permisos asignados.

* Backend: consta de los procesos, objetos y servicios que se ejecutan para guardar y proveer los datos que los distintos usuarios requieran. Los servicios constituyen la interfaz de comunicación con el Frontend. Estos servicios invocan otros procesos que consolidan datos en la Base de Datos y devuelven los resultados.
  + 1. Estándares Aplicables

La aplicación se desarrolla con el Framework Symfony en su versión 4 el cual representa la siguiente arquitectura:

**Elementos de la arquitectura**

**Cliente:** es quien requiere de algo: un servicio, información, interactuar con el sistema.

**Vista:** es la forma de presentar en la pantalla las respuestas solicitadas por el cliente.

**Plantilla:** es un modelo o template de Vistas.

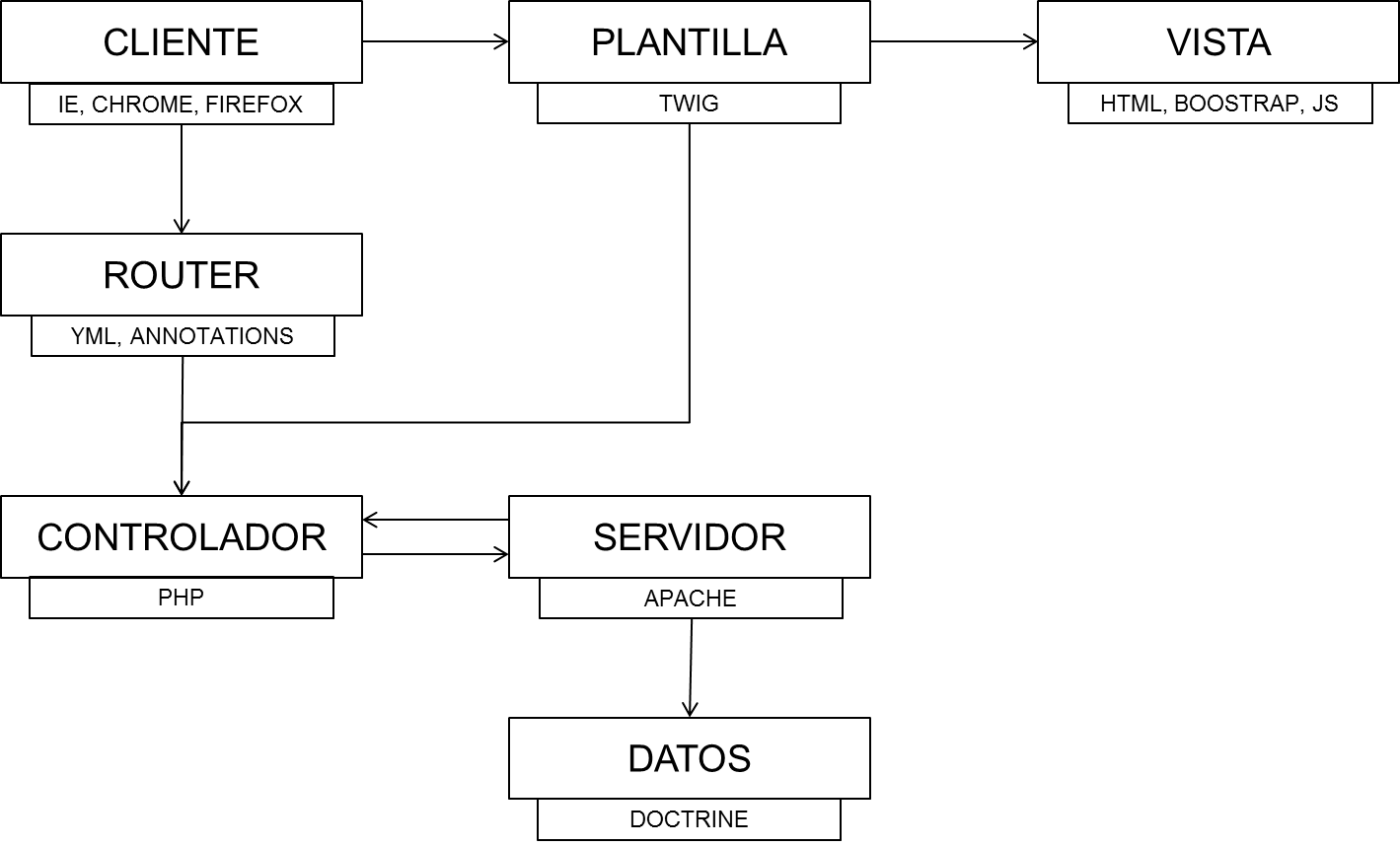
**Router:** es un servicio de ruteo de las direcciones URL que conforman la página web incluyendo acceso a los controladores y métodos.

**Controlador:** es la lógica del sistema propiamente dicha.

**Servidor:** es el encargado de recibir las peticiones del cliente y obtener la información de la base de datos.

**Datos:** es la base de datos.

**Diagrama canónico**



**Especificación 2MIL**

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Vista**  html,boostrap,js  manejar.html  manejar.boostrap  manejar.js |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Router**  Controlador,yml  direccionar.controlador  manejar.yml |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Controlador**  Servidor,php  hablar.servidor  manejar.php |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Servidor**  Controlador,Datos,apache  hablar.controlador  hablar.datos  manejar.apache |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Datos**  doctrine  manejar.doctrine  response.servidor |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Plantilla**  Vista,Controlador,twig  hablar.vista  hablar.controlador  manejar.twig |

|  |  |
| --- | --- |
| **Module**  **inports**  **exportsproc** | **Cliente**  Plantilla,Router,browser  hablar.plantilla  escuchar.router  manejar.browser |

* + 1. Requisitos de Sistema

Requerimientos Funcionales

* ABM de Equipos/Activos.
* ABM de Grupos de Trabajo.
* Gestión de Planificación.
* ABM de Avisos.
* ABM de Ordenes de trabajo.
* Gestión de Ordenes de trabajo.
* Gestión de Avisos.
* Gestión de Puntos de Medición.
* Gestión de KPI’s

Requerimientos No Funcionales

* Software actualizado para navegar en el sitio.
* Gestión de usuarios (autenticación, perfiles y permisos).
* Gestión de Auditoría.
* Aspectos indispensables de usabilidad.
  + 1. Requisitos de Desempeño del sistema

Los requisitos del desempeño del sistema no presentan funciones de emergencia. La mayoría de las transacciones dependen el estado de conexión a Internet y accesos al servidor. Los tiempos de espera no son excesivos teniendo en cuenta el tipo de transacciones.

* + 1. Requisitos de Entorno

Se debe contar con un navegador Web como Firefox, Chrome o Internet Explorer para poder hacer uso de las funciones del sistema.

Los requisitos para el servidor de la empresa, constan de espacio en disco rígido con suficiente velocidad para implementar RAID y asegurar de esta forma el backup de la información.

* 1. Requisitos de Documentación

Se pondrá a disposición del cliente un Manual de Usuario de cada funcionalidad del sistema con el detalle suficiente para su operatividad. El manual estará disponible en formato PDF como así también online para poder consultarlo vía Web en la sección de Ayuda On Line.

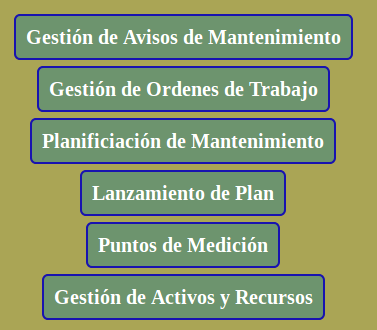
* 1. Guías de Instalación, Configuración y Fichero Léame

Para la implementación de GeMOL no es necesaria la instalación de complementos adicionales a los anteriormente mencionados.

La asignación de perfiles y permisos es definida con el cliente en reuniones presenciales y realizadas con anterioridad.

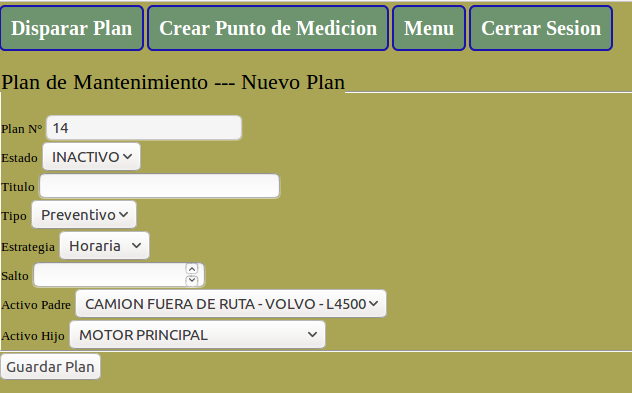
* 1. Casos de Uso – Prototipos de Interfaz de Usuario

Menu de gestiones con las funciones principales del sistema

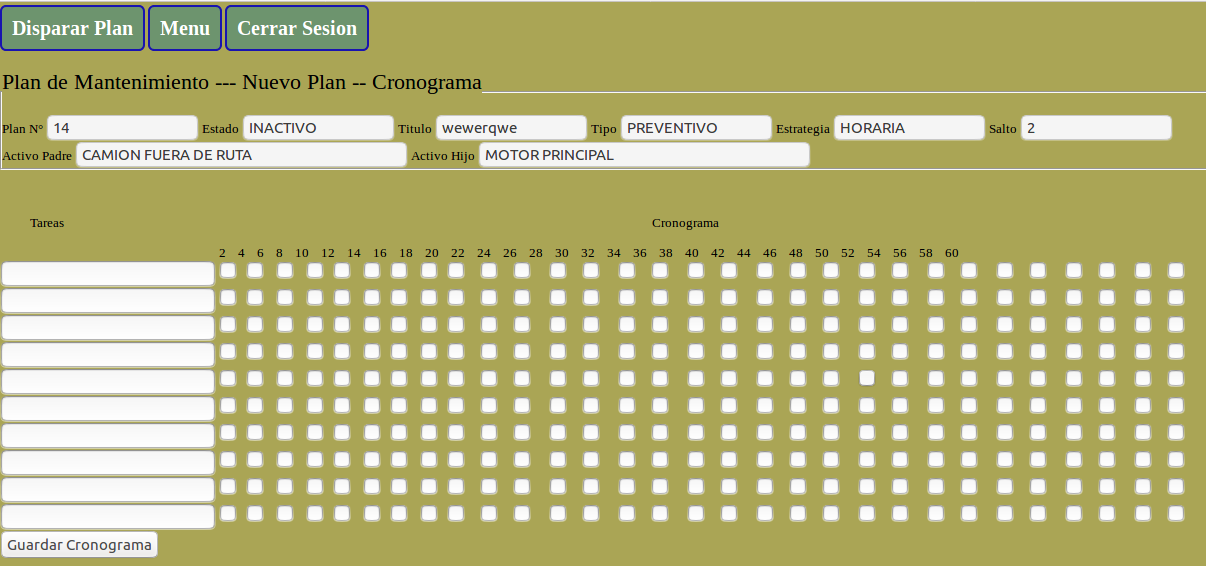


**CU01 – PLANIFICAR MANTENIMIENTO**

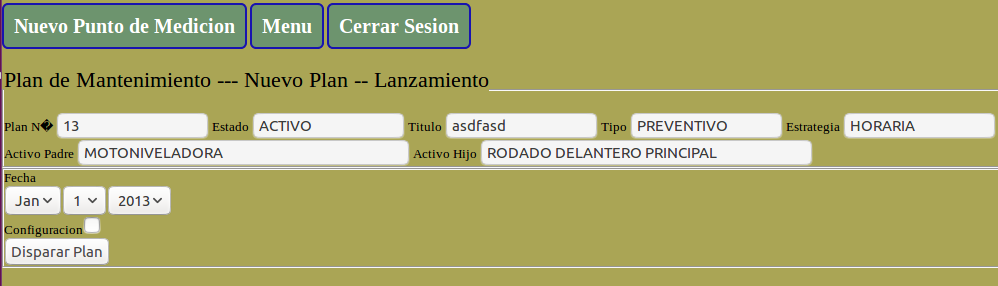
Creación de un nuevo Plan de Mantenimiento. Registra los datos de encabezado importantes para la creacion del plan.



Registración del cronograma del plan de mantenimiento donde se indican las tareas y frecuencia de ejecución de cada tarea.



Lanzamiento de plan posterior a su creación. Confirma que el gestor de plan comienza a actuar.



Carga de punto de medición para los planes de mantenimiento que se encuentran activados.

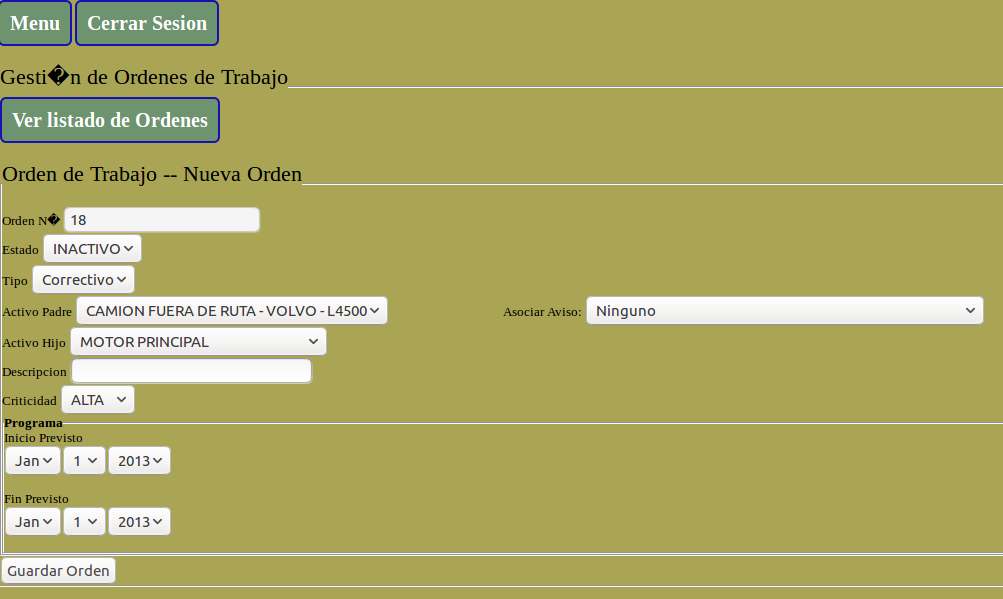


**CU01 – ORDENES DE TRABO/AVISO DE MANTENIMIENTO**

Creación manual de avisos de mantenimiento



Creación manual de ordenes de trabajo con asociación a avisos de mantenimiento.



* 1. Especificaciones de Casos de Uso

CU01 – PLANIFICAR MANTENIMIENTO

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Planificar Mantenimiento** | | **Versión: 1.0** |
| ID de caso de uso: | CU-01 – Planificar Mantenimiento | | |
| Creado por: | Martin Alice | | |
| Fecha de creación: | 04/07/2018 | | |
| Actor primario: | Analista Planificador | | |
| Actor secundario: | Ejecutante de Mantenimiento | | |
| Descripción: | Permite registrar y planificar tareas de mantenimiento que necesitan ser efectuadas sobre los equipos. En el formulario o pantalla de planificar mantenimiento se deben ingresar datos de equipos (padre y/o hijo), tareas a ser realizadas, frecuencia de cada tarea, tipo de mantenimiento (preventivo o predictivo), fecha de lanzamiento. Una vez validada la información colocada en el formulario se registra y guarda el plan estimado y se continua con el disparo del plan donde el usuario debe indicar cuando (fecha) desea que el plan comience a generar tareas en base a una configuración elegida (avisos u ordenes de mantenimiento) propuesta por el mismo usuario.  El usuario Ejecutante de Mantenimiento registra los puntos de medición de los equipos en relación a las estrategia de frecuencia definida. | | |
|
|
|
|
|
| Pre – condiciones: | El usuario debe autenticarse en el sistema con el perfil y los permisos correspondientes a las funciones que puede hacer el analista planificador y ejecutante de mantenimiento respectivamente. El sistema debe contar con información de tipos de mantenimiento, equipos (padre y/o hijo) y estrategias de frecuencia prestablecidas.  El plan se inicia con un estado “inactivo”. | | |
| Post – condiciones: | El plan que es guardado cambia automáticamente a un estado “inactivo” pasara a su estado “activo” una vez que el usuario cambia la configuración del plan en el formulario de disparo de plan.  El plan debe ser alimentado con los puntos de medición por parte del Ejecutor de mantenimiento. | | |
| Disparador | El usuario selecciona la la opción "Planificar Mantenimiento" del menú. | | |
| Escenario Principal de Éxito | El éxito del funcionamiento del caso de uso depende del registro de los puntos de medición. Si estos son cargados correctamente el funcionamiento del módulo está asegurado. | | |
| **Flujo Normal de Caso de Uso (Actor Principal)** | | | |
| **Usuario** | | **Sistema** | |
| Desde la pantalla de Planificar Mantenimiento el analista planificador debe registrar: tipo de plan, equipos padre y/o hijo, estrategia de frecuencia, tareas, y frecuencia por cada tarea. | | El sistema guarda los datos ingresados y llama a la pantalla de disparos. | |
| En la pantalla de disparo, el usuario debe ingresar el plan que desea activar, la fecha de lanzamiento, la configuración en función de si desea que se generen avisos u órdenes de mantenimiento para la autogeneración de tareas. | | El sistema valida los datos ingresados, guarda los datos del disparo y activa la lógica del negocio. | |
|
| **Flujo Alternativo 1 de Caso de Uso**  **Registro de Disparo Pospuesto (Actor Principal)** | | | |
| Desde la pantalla de Planificar Mantenimiento el analista planificador debe registrar: tipo de plan, equipos padre y/o hijo, estrategia de frecuencia, tareas, y frecuencia por cada tarea. | | El sistema guarda los datos ingresados y llama a la pantalla de disparos donde el usuario puede salir del formulario para registrar el disparo del plan en otro momento. | |
| **Flujo Alternativo 2 de Caso de Uso**  **Registro de Punto de Medición (Actor Secundario)** | | | |
| Desde la pantalla de Punto de Medición el ejecutante de mantenimiento debe registrar el valor numérico que representa el uso o avance del plan y sus equipos padre y/o hijo. | | El sistema guarda los datos ingresados y llama al procedimiento que verifica el estado anterior del punto de medición, lo compara con el nuevo, y toma la decisión de autogenerar tareas transformadas en avisos de mantenimiento u órdenes de trabajo según fue configurado en el plan. | |

CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Ordenar/Avisar tareas de Mantenimiento** | | **Versión: 1.0** |
| ID de caso de uso: | CU-02 – Ordenar/Avisar tareas de Mantenimiento | | |
| Creado por: | Martin Alice | | |
| Fecha de creación: | 04/07/2018 | | |
| Actor primario: | Supervisor de Mantenimiento | | |
| Actor secundario: | Ejecutante de Mantenimiento | | |
| Descripción: | Permite registrar ordenes de trabajo detallando tareas, recursos materiales, recursos humanos y los costos asociados a estos. En el formulario o pantalla de orden de trabajo se deben ingresar datos de equipos (padre y/o hijo), tareas a ser realizadas, tipo de mantenimiento (correctivo, preventivo o predictivo), fecha programada de inicio y final. Una vez validada la información colocada en el formulario se registra y guarda la orden y se continua con la registración de los recursos materiales. En dicha pantalla el usuario podrá registrar elementos materiales o servicios tercerizados que se realizan en la actividad principal, indicando cantidades y costos que luego el sistema computará para el presupuesto de la orden de trabajo. A continuación el sistema guarda y muestra el formulario de Recursos Humanos. En este formulario el usuario registrar las tareas detallas de la orden y el asigna a cada tarea un operario y la cantidad de horas que dedica a la tareas. La información se guarda y conforma el costo total de la orden en función de los materiales registrados y el total de horas de acuerdo a cada categoría de operario. La orden de trabajo también puede asociarse con un aviso de mantenimiento. Esto quiere decir que cuando una orden se dé por cerrada, el sistema cerrara el aviso automáticamente.  Ademas permite registrar avisos de mantenimiento donde se registra información de una falla o avería en algún equipo. El usuario debe registrar en el formulario, equipos padre y/o hijo, y criticidad de atendimiento. | | |
|
|
|
|
|
| Pre – condiciones: | El usuario debe autenticarse en el sistema con el perfil y los permisos correspondientes a las funciones que puede hacer el Supervisor y Ejecutante de mantenimiento respectivamente. El sistema debe contar con información de tipos de mantenimiento, equipos (padre y/o hijo) y lista de avisos disponibles para asociar a una orden.  La orden se inicia con un estado “abierto”.  El aviso se inicia con un estado “pendiente”. | | |
| Post – condiciones: | El aviso que es asociado con una orden cambia automáticamente a un estado “abierto” cuando la orden es registrada en el sistema. | | |
| Disparador | El usuario selecciona la la opción "Gestión de Ordenes de Trabajos" del menú.  El usuario selecciona la la opción "Avisos de Mantenimiento" del menú. | | |
| **Flujo Normal de Caso de Uso (Actor Principal)** | | | |
| **Usuario** | | **Sistema** | |
| Desde la pantalla de Orden de Trabajo: se debe registrar tipo de plan, equipos padre y/o hijo, asociación de aviso, tareas, recursos materiales, recursos humanos, y fecha de inicio y fin programados. | | El sistema guarda los datos ingresados y muestra un mensaje que los datos se guardaron correctamente. | |
| En la pantalla de Aviso, el usuario debe registrar equipos padre y/o hijo, criticidad de atendimiento y descripción de la avería. | | El sistema guarda los datos ingresados y muestra un mensaje que los datos se guardaron correctamente. | |
|
| **Flujo Alternativo 1 de Caso de Uso**  **Registro de Orden RM y RH (Actor Principal)** | | | |
| Luego de registrar los datos principales de la orden, se desplega la pantalla de agregar Recursos Materiales donde el usuario debe registrar: lista de materiales y servicios con cantidad y costos de cada uno. | | El sistema guarda los datos ingresados y llama a la pantalla de Recursos Humanos. donde el usuario puede salir del formulario para registrar el disparo del plan en otro momento. | |
| Luego de registrar los datos de recursos materiales, se desplega la pantalla de agregar Recursos Humanos donde el usuario debe registrar: lista detallada de tareas, operarios y horas previstas a trabajar. | | El sistema guarda los datos ingresados y muestra un mensaje informado que los datos se guardaron correctamente. | |
| **Flujo Alternativo 2 de Caso de Uso**  **Registro de Punto de Medición (Actor Secundario)** | | | |
| Desde la pantalla de Aviso de Mantenimiento el ejecutante de mantenimiento debe registrar el equipo padre y/o hijo que posee una falla, descripción detallada de la falla y criticidad de antendimiento. | | El sistema guarda los datos ingresados y muestra un mensaje informado que los datos se guardaron correctamente. | |

10. Usar una herramienta de modelado que sostenga la vinculación y la trazabilidad entre los requerimientos y los casos de uso.

9. Enlazar los requerimientos con los casos mediante drag & drop.

8. Evitar requerimientos disfuncionales separando detalles funcionales de tu especificación de comportamiento.

7. Escribir al menos un caso de prueba para cada requerimiento.

6. Tratar los requerimientos como si fueran ciudadanos de primera clase en el modelo.

5. Distinguí los diferentes tipos de requerimientos.

4. Evitar el síndrome del gran documento monolítico.

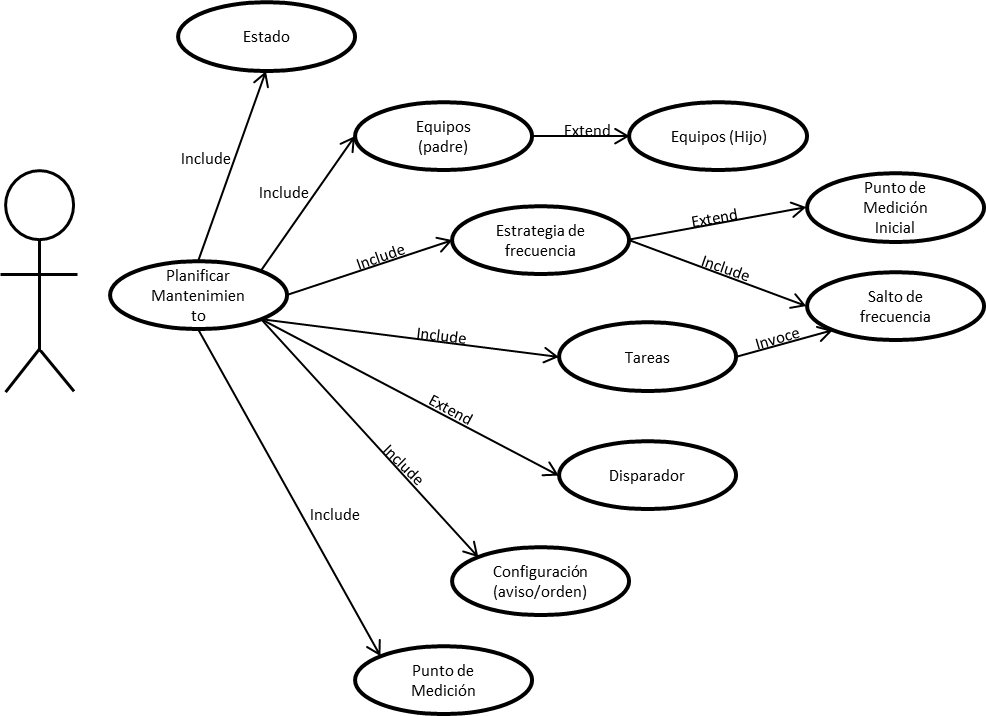
3. Crear estimaciones desde los casos de uso, NO desde los requerimientos funcionales.

2. No le temas a los ejemplos cuando escribas requerimientos funcionales.

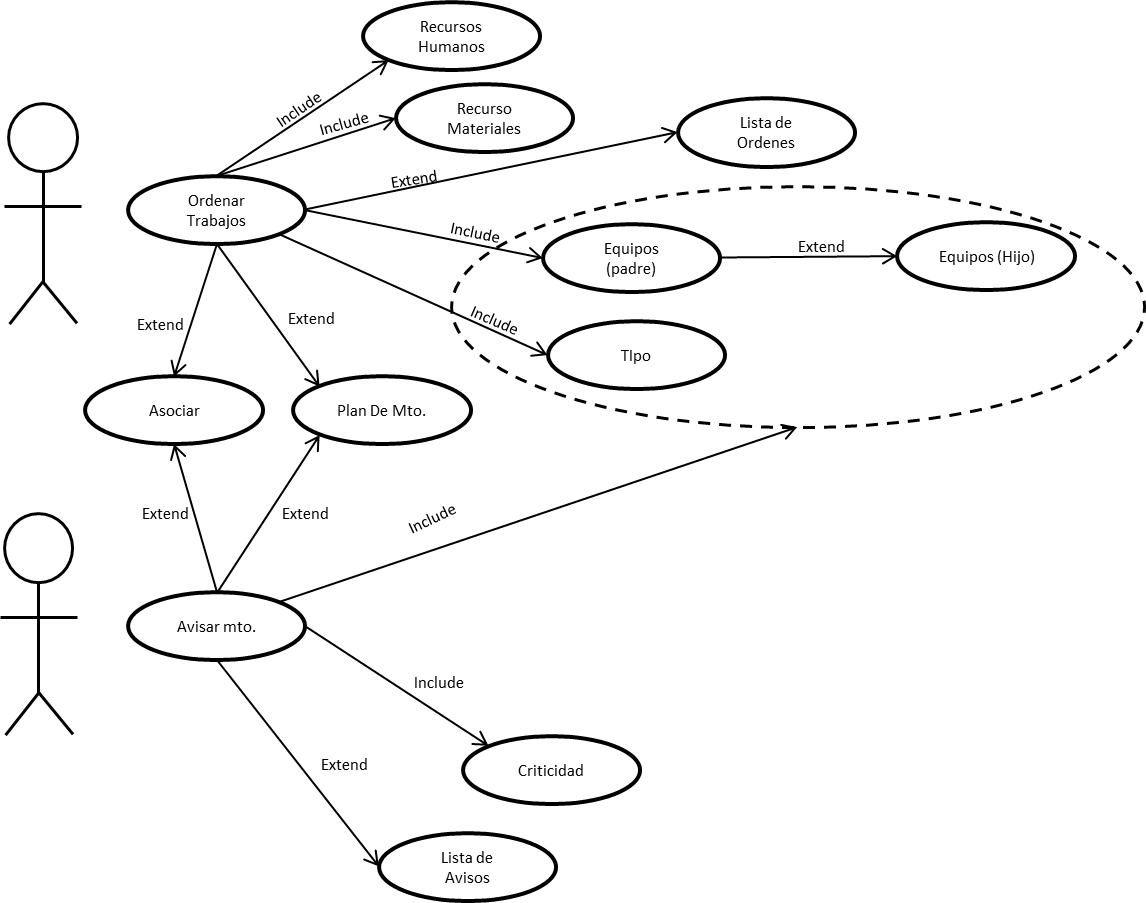
1. No hagas de los requerimientos funcionales una declaración de moda técnica.

* 1. Diagramas de Casos de Uso

CU01 – PLANIFICAR MANTENIMIENTO



CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO



Autoevaluación Casos de Uso

10. Siga la regla de los dos párrafos.

9. Organice sus casos de uso con actores

8. Escriba sus casos de uso en voz activa.

7. Escriba sus casos de uso usando un flujo de evento/respuesta, describiendo los dos lados de la conversación entre el usuario y el sistema.

6. Use prototipos GUI y maquetas de pantalla.

5. Recuerde que sus casos de uso son realmente una especificación de comportamiento en tiempo real.

4. Escriba el caso de uso en el contexto del modelo de objetos.

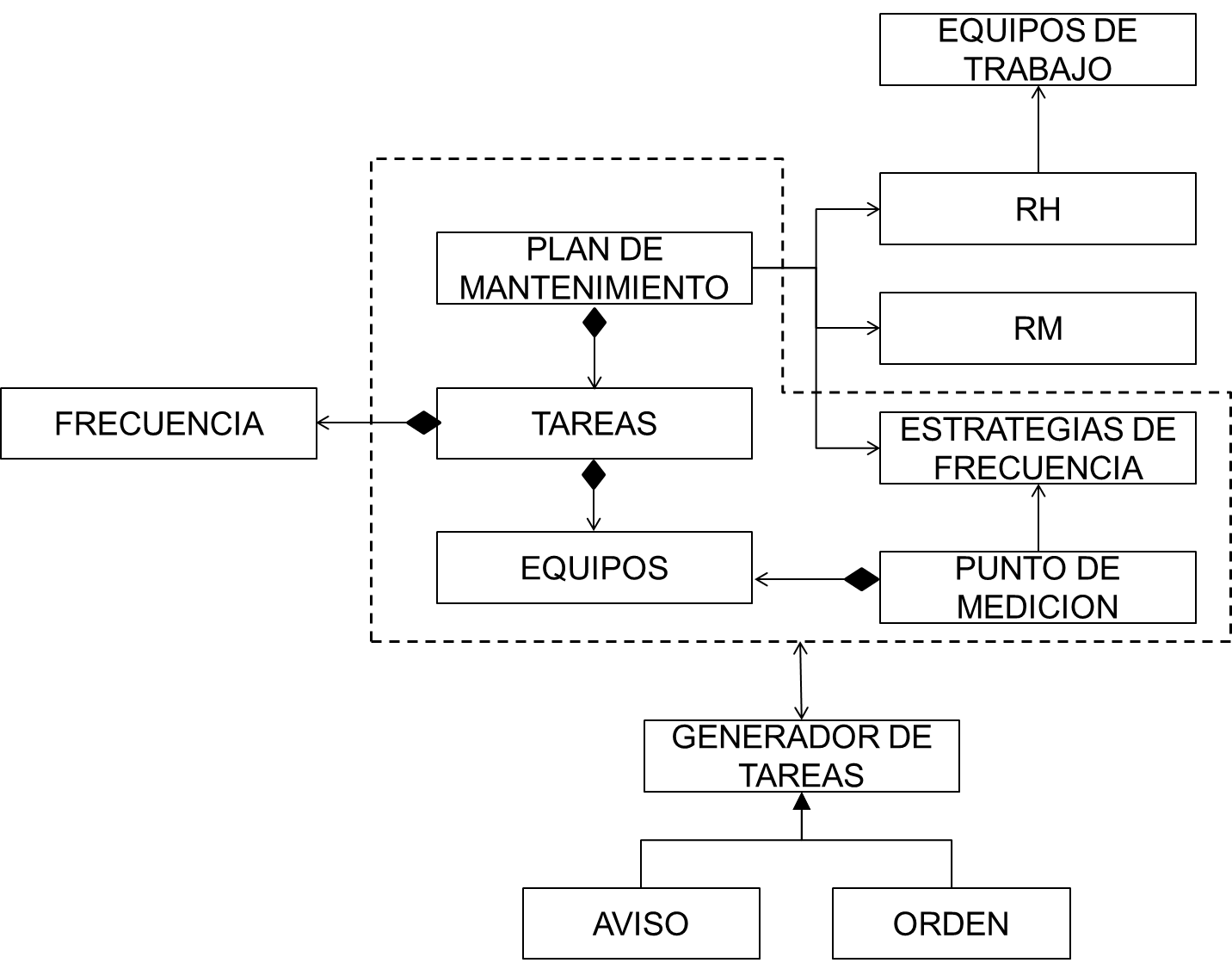
3. Escriba sus casos de uso usando una estructura de oración sustantivo-verbo-sustantivo.

2. Referencie las clases del dominio por nombre.

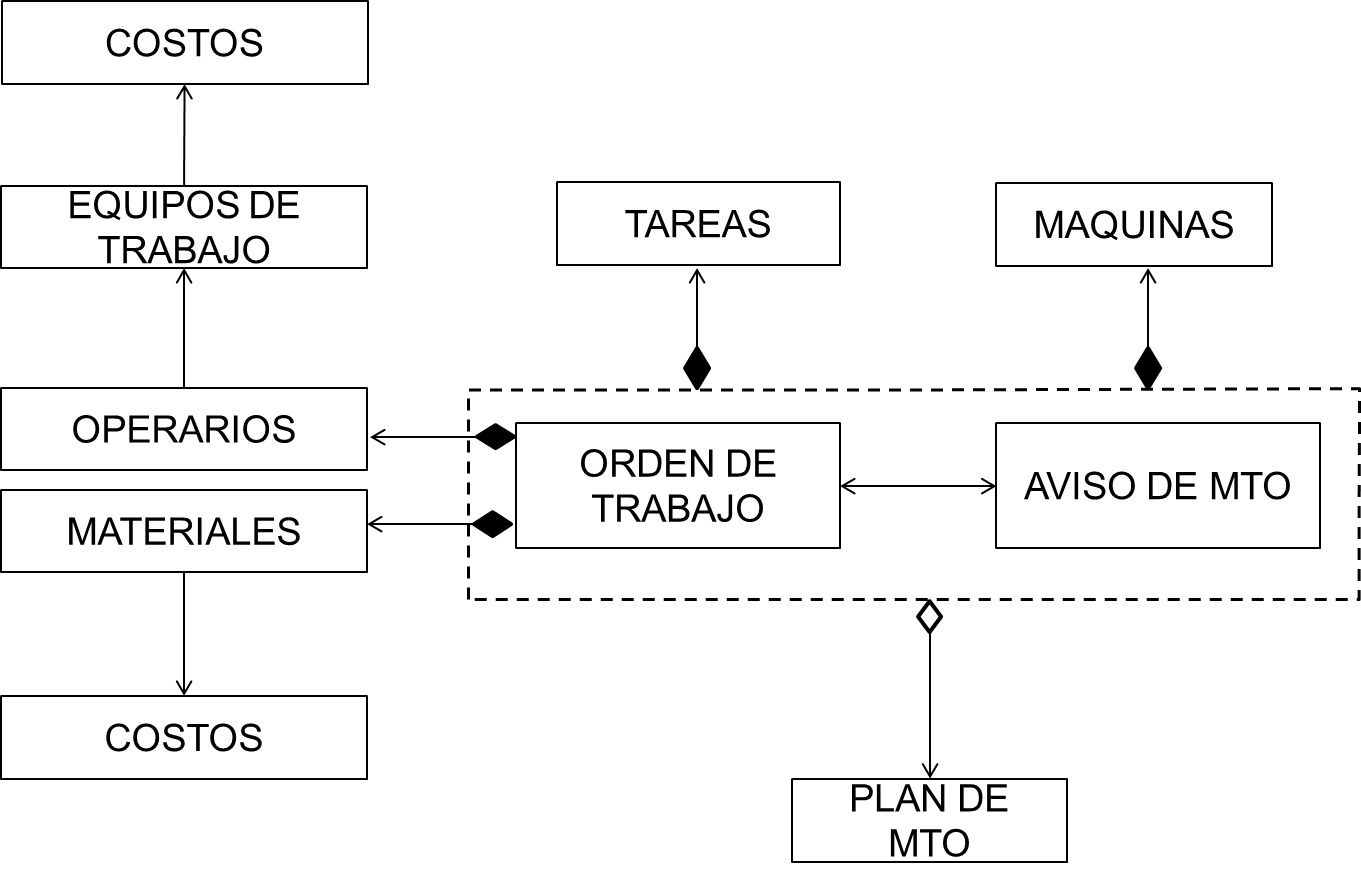
1. Referencie las clases límite (ej. pantallas) por nombre

* 1. Modelado del Dominio

CU01 – PLANIFICAR MANTENIMIENTO



CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO



Autoevaluación del modelado de dominio

10. Focalice en objetos del mundo real.

9. Use las relaciones de generalización y agregación para mostrar como los objetos se relacionan entre ellos.

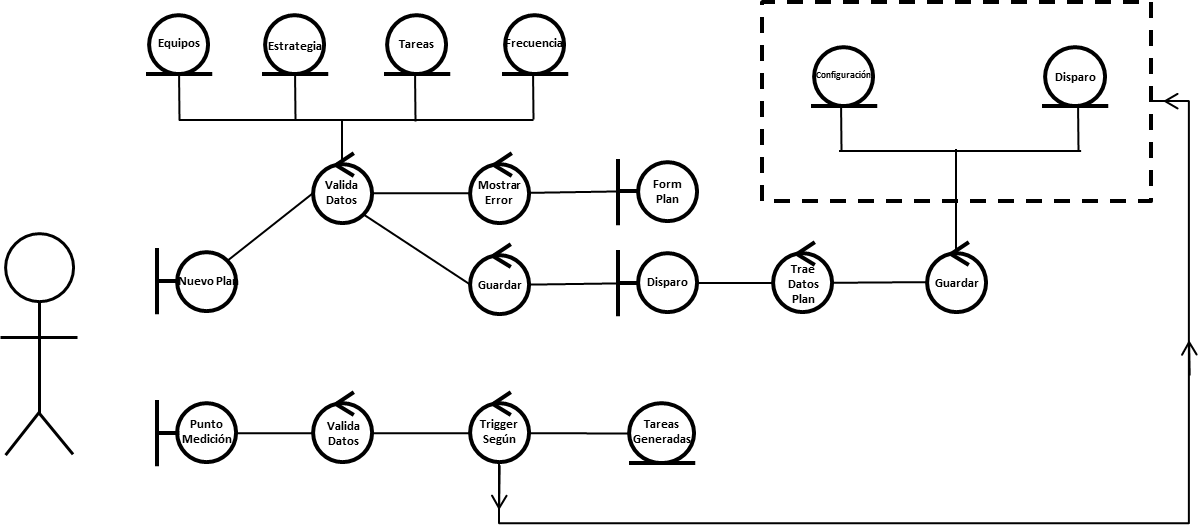
8. Limite los esfuerzos de modelado del dominio inicial a un par de horas.

7. No confunda un objeto (que representa una sola instancia) con una tabla de base de datos (que contiene una colección de cosas).

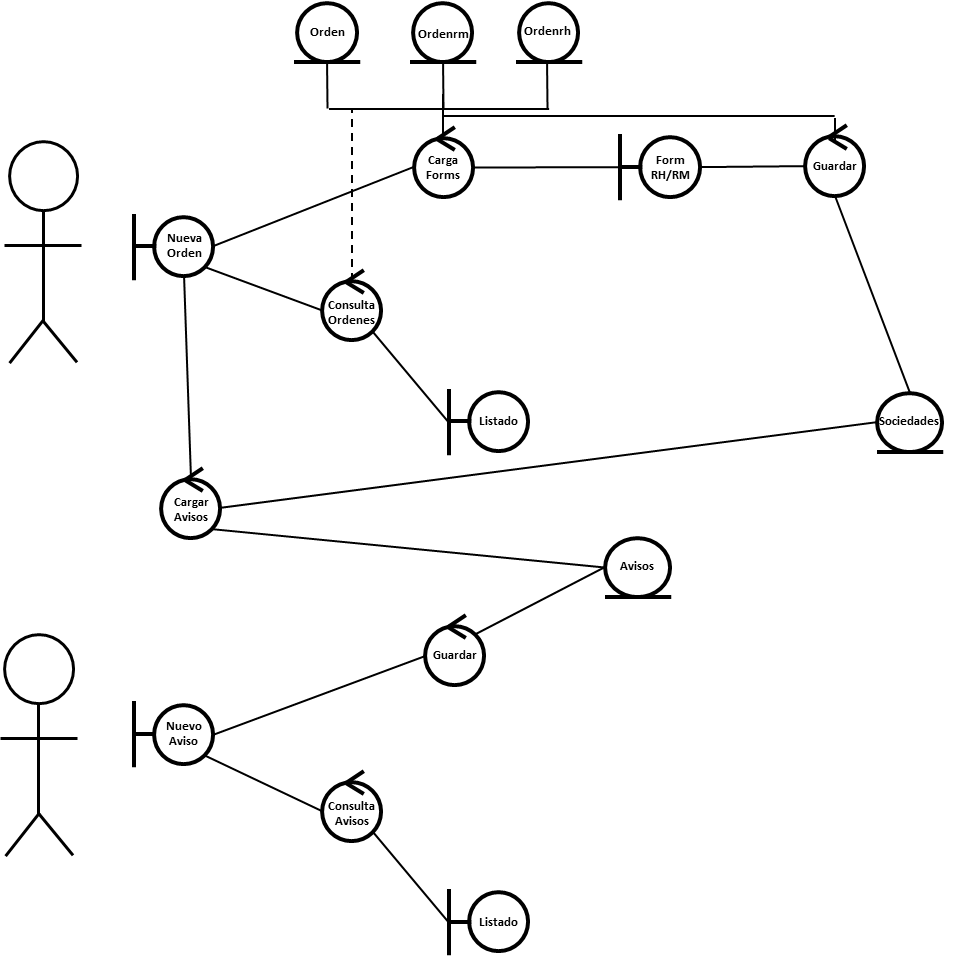
6. Use el modelo de dominio como un glosario del proyecto.  
5. Haga su modelo de dominio inicial antes de escribir sus casos de uso, para evitar ambigüedad en los nombres.  
4. No espere que su diagrama de clases final coincida precisamente con el modelo de dominio, pero debería haber alguna semejanza.  
3. Organice sus clases alrededor de las abstracciones de claves en el dominio del problema.  
2. Recuerde que lo que es buena práctica en un modelo de datos no es probable que sea buena práctica en un diagrama de clases (y viceversa).  
1. No ponga pantallas y otras clases de GUI específicos en su modelo de dominio.

* 1. Diagrama de Robustez

CU01 – PLANIFICAR MANTENIMIENTO



CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO



Autoevaluación del diagrama de robustez

10. Pegue el texto del caso de uso directamente en su diagrama de robustez.

9. Tome sus clases de entidad desde el modelo de dominio, y agregue cualquiera que esté faltando.

8. Cuente con que deberá reescribir (desambiguar) su caso de uso mientras dibuja el diagrama de robustez.

7. Haga un objeto límite para cada pantalla, y nombre sus pantallas sin ambigüedad.

6. Recuerde que los controladores son solo ocasionalmente objetos de control reales, y que son normalmente funciones de software lógicas.

5. No se preocupe por la dirección de las flechas en un diagrama de robustez.

4. Está bien arrastrar un caso de uso en un diagrama de robustez si este se llama desde un caso de uso padre.

3. El diagrama de robustez representa un diseño preliminar y conceptual de un caso de uso, no un diseño detallado y literal.

2. Las clases de límite y entidad en un diagrama de robustez generalmente se convertirán en instancias de objeto en un diagrama de secuencias, mientras que los controladores serán mensajes.

1. Recuerde que un diagrama de robustez es una "imagen de objeto" de un caso de uso, cuyo propósito es forzar el refinamiento del texto del caso de uso y del modelo de objetos.

Revisión Preliminar

10. Para cada caso de uso, verificar que el texto del caso de uso coincida con el diseño de robustez.

9. Verificar que todas las entidades de todos los diagramas de robustez aparezcan dentro del modelo de dominio actualizado.

8. Verificar que puedas rastrear flujos de datos entre clases de entidad y pantallas.

7. No te olvides de los cursos alternativos, y no te olvides de escribir comportamientos para cada uno de ellos cuando los encuentres.

6. Verificar que cada caso de uso cubra ambos lados del diálogo entre usuario y sistema.

5. Verificar que no hayas violado las reglas de sintaxis en el análisis de robustez.

4. Verificar que ésta revisión incluya gente técnica (programadores) y no técnica (cliente, equipo de marketing, etc.).

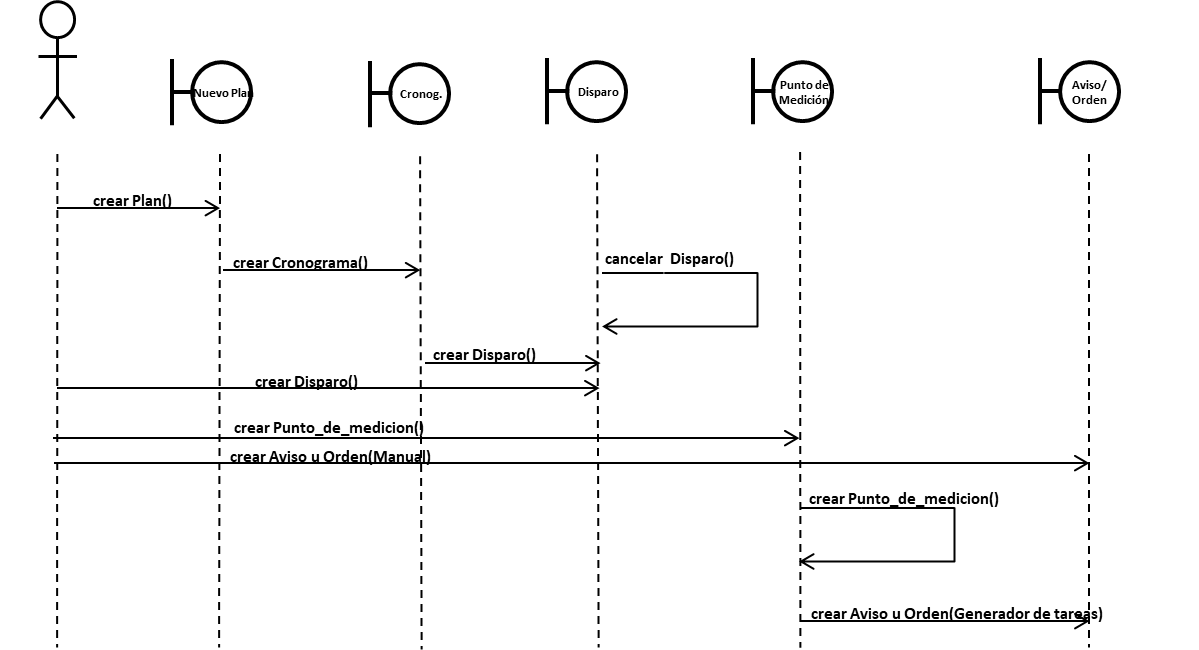
3. Verificar que los casos de uso estén en el contexto del modelo de objetos y en el contexto de la GUI.

2. Verificar que tus diagramas de robustez (y el correspondiente texto de caso de uso) no muestren el mismo nivel de detalle que próximamente será mostrado en los diagramas de secuencia (no trates de hacer diseño detallado todavía).

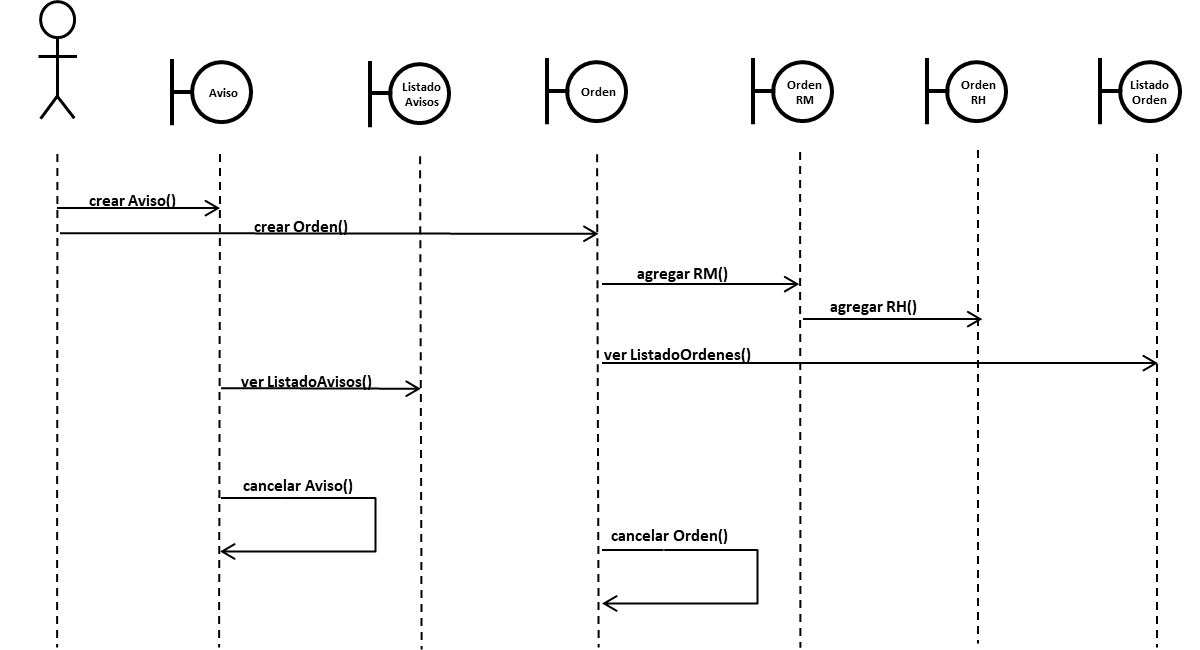
1. Seguir los “Pasos para un mejor PDR” anteriormente mencionado.

* 1. Diagrama de Secuencia

CU01 – PLANIFICAR MANTENIMIENTO



CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO



Autoevaluación Diagramas de Secuencia

10. Entienda por qué está dibujando un diagrama de secuencias, para obtener mejores resultados.

9. Realice un diagrama de secuencias por cada caso de uso.

8. Comience su diagrama de secuencias desde las clases límite, clases entidad, actores, y textos de casos de uso que resulten del análisis de robustez.

7. Use el diagrama de secuencias para mostrar cómo el comportamiento del caso de uso (ej. el controlador del diagrama de robustez) es cumplido por los objetos.

6. Asegúrese de que su texto de los casos de uso tienen relación con los mensajes que son pasados en los diagramas de secuencias. Intente alinear el texto y las flechas de mensajes.

5. No dedique mucho tiempo preocupándose acerca del foco de control.

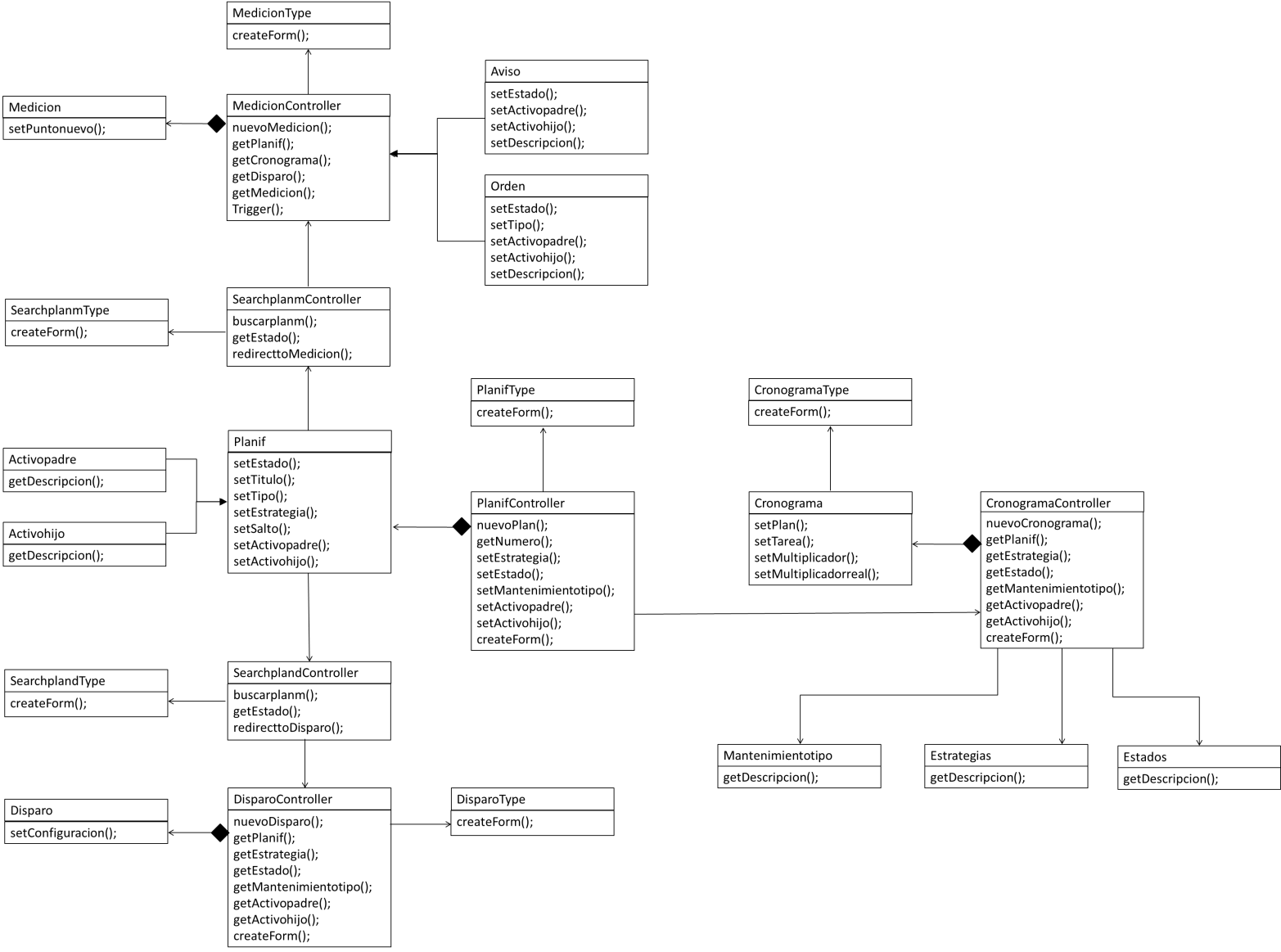
4. Asigne operaciones a las clases mientras dibuja los mensajes. Muchas herramientas de modelado soportan esta funcionalidad.

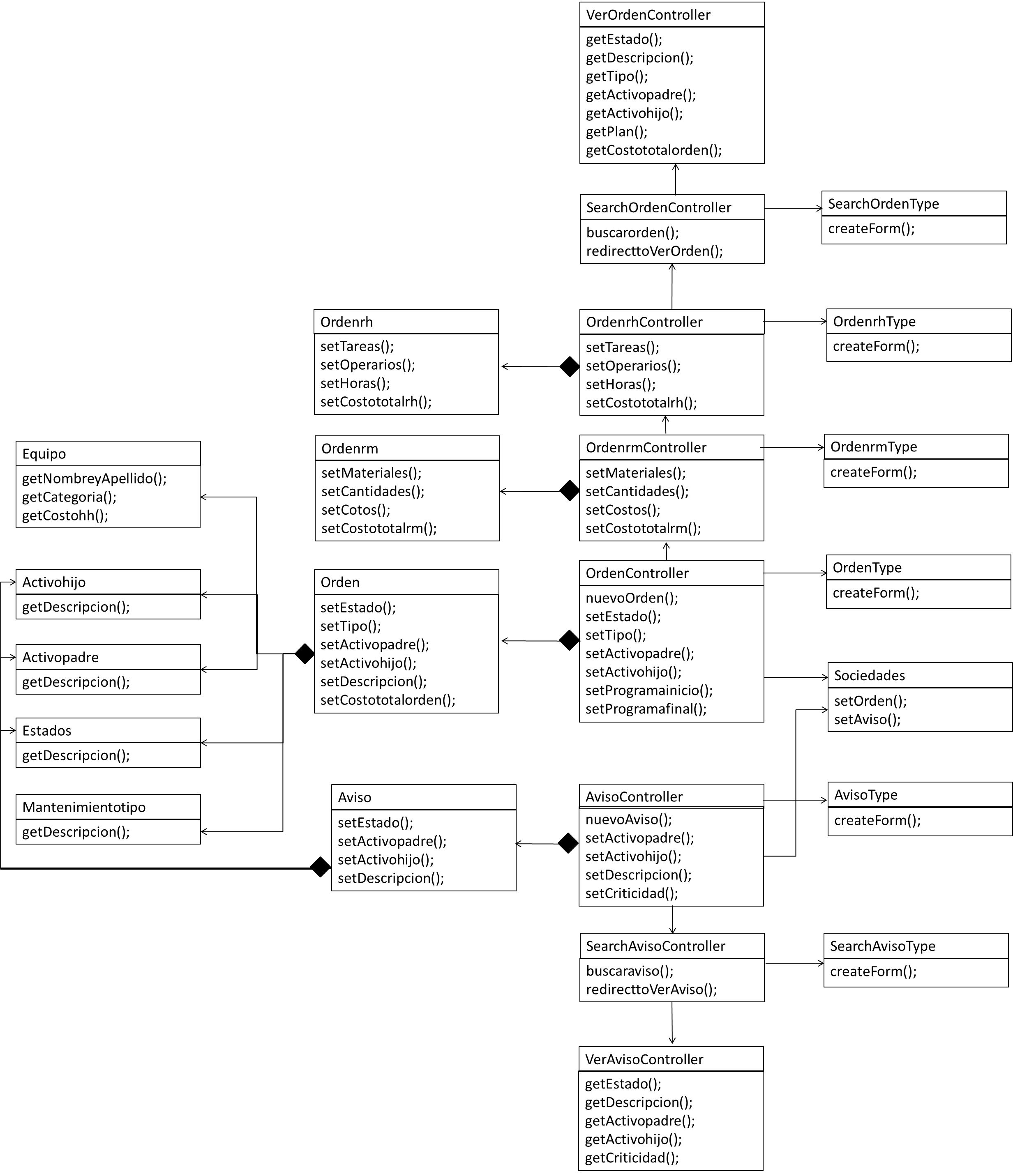
3. Revise su diagrama de clases frecuentemente mientras está asignando las operaciones a las clases, para asegurar que todas las operaciones están en la clase correspondiente.

2. Realice los diagramas de secuencias antes de codificar.

1. Intente realizar los cursos básicos y alternativos en el mismo diagrama de secuencias.

* 1. Diagrama de Clases

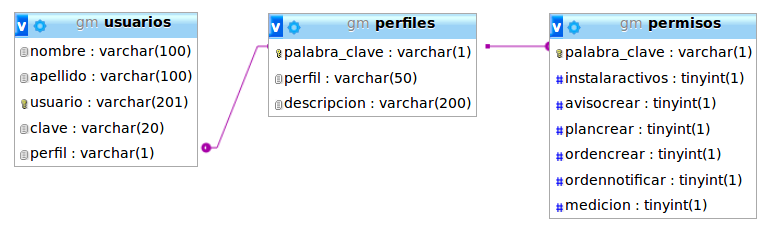
CU01 – PLANIFICAR MANTENIMIENTO

CU02 – ORDENAR/AVISAR TAREAS DE MANTENIMIENTO

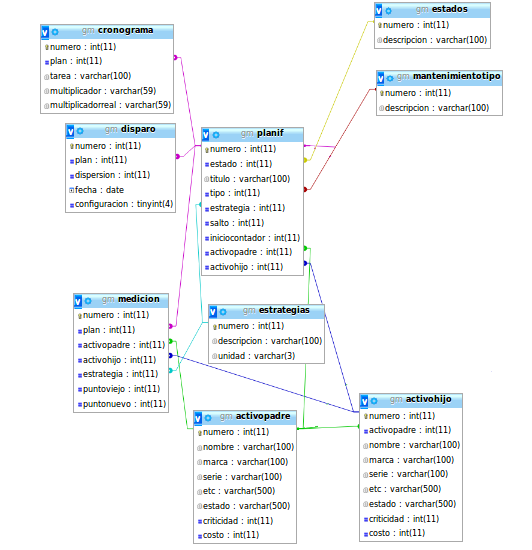
* 1. Modelo de Datos

A continuación se detalla el Modelo de Datos con las tablas correspondientes a las Entidades de todo el sistema.

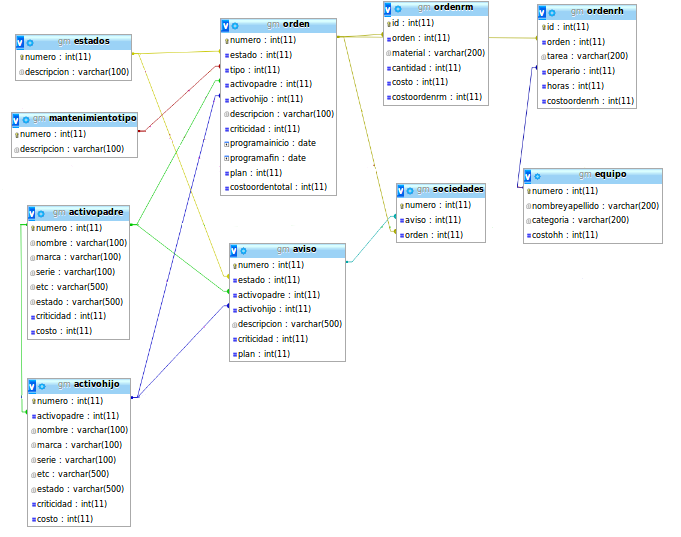
En primer lugar se muestra la normalización de las tablas que conforman el modulo de seguridad:



En segundo lugar se muestra el modelo de datos relacionado con las entidades que forman parte del caso de uso CU01 – Planificar Mantenimiento.



En tercer lugar se muestra la representación del modelo de datos del caso de uso CU02 – Ordenar/Avisar tareas de Mantenimiento:



* 1. Seguridad y Auditoría

|  |  |
| --- | --- |
| **Elementos del Sistema** | **Función principal** |
| Acceso al Sistema | Los accesos al sistema deberán ser controlados mediante la autenticación de usuarios con un nombre y contraseña preestablecidos y entregados de una forma segura al cliente una vez que es adquirido el producto.  El proyecto no contemplara la registración de usuarios a través de la aplicación web. Los nombres de usuarios con sus respectivos roles y contraseñas son suministrados al cliente en forma personal y en mano. Esta información sera volcada a la base de datos de Aligaro en donde las contraseñas se almacenaran de forma cifrada para evitar que sea conocida por agentes externos que puedan acceder a la información que envía el usuario a través de la aplicación web. |
| Perfiles y Roles de Usuarios | Cada cliente podrá obtener, como máximo, 3 perfiles de usuario, donde cada uno de estos tendrá la posibilidad de utilizar determinadas funciones y recursos del sistema de acuerdo a sus respectivas funciones en el proceso real de mantenimiento de equipos.  Todo proceso de gestión suministra a sus participantes una serie de roles y responsabilidades que por políticas de ética y transparencia propias de cada empresa, dichas responsabilidades no se deben superponer o solapar y deben responden a un método de trabajo coherente con la realidad de los procesos de la gestión del mantenimiento. Por ejemplo, un usuario cuya función es ejecutar tareas de mantenimiento, no debe poder planificar, programar ni autorizar dichas tareas. Estas funciones corresponden a un usuarios con un perfil del tipo supervisor o planificador.  Ante esta situación, el sistema deberá ofrecer al cliente 3 perfiles fundamentales:  Supervisor de Mantenimiento.  Planificador de Mantenimiento.  Ejecutante de Mantenimiento. |
| Permisos de uso y privilegios | Cada perfil antes descripto podrá realizar determinadas funciones según los privilegios dados por el administrador del sistema y atendiendo a los procesos reales de la gestión de mantenimiento.  En la siguiente tabla se reflejan los posibles privilegios y permisos de uso del sistema que deberá tener cada perfil de usuario.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **PERFIL** | **GESTION DE ACTIVOS** | **GESTION DE AVISOS** | **GESTION DE PLANES** | **GESTION DE ORDENES** | **GESTION DE NOTIFICACIONES** | **GESTION DE MEDICIONES** | | Ejecutante | NO | SI | NO | NO | NO | NO | | Planificador | NO | NO | SI | SI | NO | SI | | Supervisor | SI | NO | NO | NO | SI | NO | |
| Implementación ORM | La persistencia de datos deberá ser manejada a través del conocido ORM Doctrine utilizado para representar las relaciones entre tablas de la base de datos logrando de esta forma una gran abstracción de código al sistema y la reducción de probabilidad de cambios en la codificación del sistema a futuro. |

* 1. Pruebas Unitarias

Para las pruebas unitarias se realizarán dos enfoques de pruebas, Caja Blanca y Caja Negra.

En las pruebas de Caja Negra se centralizan en los requisitos funcionales. Se verifica la interfaz del Sistema, verificando que las entradas y salidas sean las esperadas.

En las pruebas de Caja Blanca se verifica el funcionamiento de un módulo seleccionado. Es un enfoque de pruebas estructurado. Son las funciones internas de dicho procedimiento las que se validan con esta prueba.

* + 1. Prueba Caja Negra

Se definen distintos escenarios de pruebas para validar las entradas y salidas esperadas del sistema. Los escenarios se focalizan en las funciones del CU01 – Planificar Mantenimiento:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Condición | Descripción de la condición | Dato de entrada | Resultado esperado | Estado |
| Crear nuevo Plan | Validar que el número de plan sea el número mayor siguiente al último plan guardado | Numero de plan (tomado directamente desde la base de datos) | Visualización del numero en pantalla | Aprobado |
| Crear nuevo Plan | Validación del estado del nuevo plan el cual | Estado: INACTIVO | Visualizado en pantalla | Aprobado |
| Crear nuevo Plan | Validar el tipo de plan | Deber ser un campo de opciones tomado con información de la base de datos con los distintos tipos de planes de mantenimiento. Menos el tipo “correctivo”. | Visualizado en pantalla | Aprobado |
| Crear nuevo Plan | Validar Estrategias definidas. | Debe ser un campo de opciones tomado de la base de datos con las distintas estrategias. | Visualizado en pantalla | Aprobado |
| Crear nuevo Plan | Validación de salto | El salto tiene que ser un valor numérico mayor a 1 y menor que 999.999 | Mostrar el mensaje de error en caso de registrar otro valor que esta por fuera del rango. | Aprobado |
| Crear nuevo Plan | Selección del activo padre | Debe ser un campo seleccionable con todos los activos padre traídos de la base de datos | Visualizado en pantalla. | Aprobado |
| Crear nuevo Plan | Selección del activo hijo | Debe ser un campo seleccionable con todos los activos hijos que son parte del activo padre anteriormente seleccionado. Esta información de debe captar de la base de datos. | Visualizado en pantalla | Aprobado |
| Crear nuevo Plan | Click al botón de guardar | Procesar el controlador | Mostrar mensaje de éxito al guardar | Aprobado |
| Crear nuevo Cronograma | Ingresar tareas | Verificar que al menos la primera línea de tareas tenga un valor | Mostrar error en caso de intentar grabar sin haber registrado el dato | Aprobado |
| Crear nuevo Cronograma | Tildar frecuencias | Verificar que al menos una frecuencia de la primera línea de tareas este activada | Mostrar mensaje de error en caso de intentar grabar sin seleccionar una frecuencia | Aprobado |
| Crear nuevo Cronograma | Click al botón de guardar | Procesar el controlador | Mostrar mensaje de éxito al guardar | Aprobado |
| Crear nuevo Lanzamiento | Ingreso del campo fecha | Validar que sea un formato valido (yyyy-MM-dd) y posterior o igual a la fecha actual. | Mostrar el mensaje de error | Aprobado |
| Crear nuevo Lanzamiento | Click al botón de guardar | Procesar el controlador | Mostrar mensaje de éxito al guardar | Aprobado |
| Crear nuevo Punto de Medición | Ingreso del punto viejo | Campo bloqueado. Debe traerlo de la base de datos según el plan seleccionado. | Visualizado en pantalla | Aprobado |
| Crear nuevo Punto de Medición | Ingreso del punto nuevo | Validar que el campo debe ser superior al punto viejo y no mayor a 999.999 | Mostrar el mensaje de error | Aprobado |
| Crear nuevo Punto de Medición | Click al botón de guardar | Procesar el controlador | Mostrar mensaje de éxito al guardar | Aprobado |

* + 1. Prueba Caja Blanca

Las pruebas de caja blanca se realizan mediante el complemento PHPUNIT que provee el FrameWork Symfony. Se selecciona una funcionalidad interna del caso de uso CU01 Plan de Mantenimiento y sobre la misma se verifica las respuestas en el siguiente código:

Crear nuevo Plan/Cronograma/Disparo y Punto de Medición

Una vez cliqueado el botón de guardar plan, se invoca al controlador que registra en la base de datos la información completada por el usuario.

<?php

namespace App\Controller;

//invoca las entidades que necesita

use Symfony\Component\HttpFoundation\Response;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Doctrine\ORM\EntityManager;

use App\Entity\Planif;

use App\Entity\Medicion;

use Symfony\Component\HttpFoundation\Request;

use App\Form\PlanifType;

class PlanifController extends Controller

{

public function nuevoPlan(Request $request)

{

// conecta con la base de datos y obtiene el último número de plan

$em = $this->getDoctrine()->getManager();

$db = $em->getConnection();

$query = "SELECT \* FROM planif ORDER BY numero DESC LIMIT 1;";

$stmt = $db->prepare($query);

$stmt->execute();

$po=$stmt->fetchColumn();

$plannum = $this->getDoctrine()->getManager()->getRepository('App:Planif')->find($po);

// crea una nueva instancia a la entidad Plan

$plan = new Planif();

// crea el formulario manejando el método Request

$form = $this->createForm(PlanifType::class, $plan);

$form->handleRequest($request);

// si el formulario es valido, guarda los datos en la base de datos

if($form->isSubmitted() && $form->isValid())

{

$plan = $form->getData();

$em->persist($plan);

$em->flush();

// genera un registro nuevo en la entidad Medicion con un valor inicial 0

$medicion = new Medicion();

$medicion->setPlan($plannum->getNumero() + 1);

$medicion->setActivopadre($form->get('activopadre')->getData());

$medicion->setActivohijo($form->get('activohijo')->getData());

$medicion->setEstrategia($form->get('estrategia')->getData());

$medicion->setPuntoviejo('0');

$medicion->setPuntonuevo('0');

$em->persist($medicion);

$em->flush();

// redirecciona a la ruta Cronograma

return $this->redirectToRoute('cronograma');

}

// renderiza el formulario Plan

return $this->render('plan.html.twig', array('form' => $form->createView(), 'plannum' => $plannum));

}

}

A continuación se registra el cronograma del plan.

<?php

namespace App\Controller;

// invoca las entidades que necesita

use Symfony\Component\HttpFoundation\Response;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Doctrine\ORM\EntityManager;

use App\Entity\Cronograma;

use App\Entity\Medicion;

use App\Entity\Planif;

use App\Entity\Estados;

use App\Entity\Mantenimientotipo;

use App\Entity\Estrategias;

use App\Entity\Activohijo;

use App\Entity\Activopadre;

use Symfony\Component\HttpFoundation\Request;

use App\Form\CronogramaType;

class CronogramaController extends Controller

{

public function nuevoCronograma(Request $request)

{

// obtiene el número de plan al cual corresponde el cronograma

$em = $this->getDoctrine()->getManager();

$db = $em->getConnection();

$query = "SELECT \* FROM planif ORDER BY numero DESC LIMIT 1;";

$stmt = $db->prepare($query);

$stmt->execute();

$po=$stmt->fetchColumn();

$plannum = $this->getDoctrine()->getManager()->getRepository('App:Planif')->find($po);

// inserta en variables los datos del encabezado

$e = $plannum->getEstado();

$est = $plannum->getEstrategia();

$t = $plannum->getTipo();

$ap = $plannum->getActivopadre();

$ah = $plannum->getActivohijo();

$queryestado = "SELECT \* FROM estados WHERE numero = ".$e.";";

$stmtestado = $db->prepare($queryestado);

$stmtestado->execute();

$poestado=$stmtestado->fetchColumn();

$estado = $this->getDoctrine()->getManager()->getRepository('App:Estados')->find($poestado);

$queryestrategia = "SELECT \* FROM estrategias WHERE numero = ".$est.";";

$stmtestrategia = $db->prepare($queryestrategia);

$stmtestrategia->execute();

$poestrategia=$stmtestrategia->fetchColumn();

$estrategia = $this->getDoctrine()->getManager()->getRepository('App:Estrategias')->find($poestrategia);

$querytipo = "SELECT \* FROM mantenimientotipo WHERE numero = ".$t.";";

$stmttipo = $db->prepare($querytipo);

$stmttipo->execute();

$potipo=$stmttipo->fetchColumn();

$tipo = $this->getDoctrine()->getManager()->getRepository('App:Mantenimientotipo')->find($potipo);

$queryap = "SELECT \* FROM activopadre WHERE numero = ".$ap.";";

$stmtap = $db->prepare($queryap);

$stmtap->execute();

$poap=$stmtap->fetchColumn();

$actp = $this->getDoctrine()->getManager()->getRepository('App:Activopadre')->find($poap);

$queryah = "SELECT \* FROM activohijo WHERE numero = ".$ah.";";

$stmtah = $db->prepare($queryah);

$stmtah->execute();

$poah=$stmtah->fetchColumn();

$acth = $this->getDoctrine()->getManager()->getRepository('App:Activohijo')->find($poah);

// crea una nueva instancia del cronograma

$cronograma = new Cronograma();

// crea el formulario manejando el método Request

$form = $this->createForm(CronogramaType::class, $cronograma);

$form->handleRequest($request);

// si el formulario es válido, guarda la información en la base de datos

if($form->isSubmitted() && $form->isValid())

{

for($x=1;$x<=10;$x++)

{

$tarea = $form->get('tarea'.$x)->getData();

if($tarea!='')

{

for($i=1;$i<=30;$i++)

{

$crono[$i] = $form->get($x.$i)->getData();

if($crono[$i]==false)

{

$crono[$i] = '0';

}

else

{

$crono[$i] = '1';

}

}

$saltito = implode(',',$crono);

$cronograma = new Cronograma();

$cronograma->setPlan($plannum->getNumero());

$cronograma->setTarea($tarea);

$cronograma->setMultiplicador($saltito);

$cronograma->setMultiplicadorreal($saltito);

$em->persist($cronograma);

$em->flush();

}

}

// mensaje de exito

$this->addFlash('notice', 'Tus cambios se han guardado!');

}

// renderiza la vista Cronograma

return $this->render('cronograma.html.twig', array('form' => $form->createView(), 'plannum' => $plannum, 'estado' => $estado, 'estrategia' => $estrategia, 'tipo' => $tipo, 'actp' => $actp, 'acth' => $acth));

}

}

A continuación se dispara el plan.

<?php

namespace App\Controller;

// invoca las entidades que necesita

use Symfony\Component\HttpFoundation\Response;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Doctrine\ORM\EntityManager;

use App\Entity\Disparo;

use App\Entity\Transito;

use Symfony\Component\HttpFoundation\Request;

use App\Form\DisparoType;

use App\Form\SearchPlantoDisparoType;

use App\Controller\SearchPlanDisparo;

class DisparoController extends Controller

{

public function nuevoDisparo(Request $request)

{

$em = $this->getDoctrine()->getManager();

$db = $em->getConnection();

$queryn = "SELECT \* FROM transito ORDER BY id DESC LIMIT 1;";

$stmtn = $db->prepare($queryn);

$stmtn->execute();

$pon=$stmtn->fetchColumn();

$plannumn = $this->getDoctrine()->getManager()->getRepository('App:Transito')->find($pon);

$num = $plannumn->getNumero();

// obtiene de la base de datos la información del plan que se va a disparar

$query = "SELECT \* FROM planif WHERE numero =".$num.";";

$stmt = $db->prepare($query);

$stmt->execute();

$po=$stmt->fetchColumn();

$plannum = $this->getDoctrine()->getManager()->getRepository('App:Planif')->find($po);

$e = $plannum->getEstado();

$est = $plannum->getEstrategia();

$t = $plannum->getTipo();

$ap = $plannum->getActivopadre();

$ah = $plannum->getActivohijo();

$queryestado = "SELECT \* FROM estados WHERE numero = ".$e.";";

$stmtestado = $db->prepare($queryestado);

$stmtestado->execute();

$poestado=$stmtestado->fetchColumn();

$estado = $this->getDoctrine()->getManager()->getRepository('App:Estados')->find($poestado);

$queryestrategia = "SELECT \* FROM estrategias WHERE numero = ".$est.";";

$stmtestrategia = $db->prepare($queryestrategia);

$stmtestrategia->execute();

$poestrategia=$stmtestrategia->fetchColumn();

$estrategia = $this->getDoctrine()->getManager()->getRepository('App:Estrategias')->find($poestrategia);

$querytipo = "SELECT \* FROM mantenimientotipo WHERE numero = ".$t.";";

$stmttipo = $db->prepare($querytipo);

$stmttipo->execute();

$potipo=$stmttipo->fetchColumn();

$tipo = $this->getDoctrine()->getManager()->getRepository('App:Mantenimientotipo')->find($potipo);

$queryap = "SELECT \* FROM activopadre WHERE numero = ".$ap.";";

$stmtap = $db->prepare($queryap);

$stmtap->execute();

$poap=$stmtap->fetchColumn();

$actp = $this->getDoctrine()->getManager()->getRepository('App:Activopadre')->find($poap);

$queryah = "SELECT \* FROM activohijo WHERE numero = ".$ah.";";

$stmtah = $db->prepare($queryah);

$stmtah->execute();

$poah=$stmtah->fetchColumn();

$acth = $this->getDoctrine()->getManager()->getRepository('App:Activohijo')->find($poah);

// crea una nueva instancia del Disparo

$disparo = new Disparo();

$disparo->setPlan($plannum->getNumero());

// crea el formulario manejando el método Request

$form = $this->createForm(DisparoType::class, $disparo);

$form->handleRequest($request);

if($form->isSubmitted() && $form->isValid())

{

// cambia el estado del Plan

$plannum->setEstado('1');

$em->flush();

$disparo = $form->getData();

$em->persist($disparo);

$em->flush();

// muestra mensaje de exito

$this->addFlash('notice', 'Tus cambios se han guardado!');

}

// renderiza la vista Disparo

return $this->render('disparo.html.twig', array('form' => $form->createView(), 'plannum' => $plannum, 'estado' => $estado, 'estrategia' => $estrategia, 'tipo' => $tipo, 'actp' => $actp, 'acth' => $acth));

}

}

Por ultimo en la prueba se verifica el funcionamiento del Punto de Medición.

<?php

namespace App\Controller;

// invoca las entidades que necesita

use Symfony\Component\HttpFoundation\Response;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Doctrine\ORM\EntityManager;

use App\Entity\Medicion;

use App\Entity\Notificacion;

use App\Entity\Cronograma;

use App\Entity\Planif;

use App\Entity\Disparo;

use App\Entity\Aviso;

use App\Entity\Orden;

use App\Entity\Transito;

use Symfony\Component\HttpFoundation\Request;

use App\Form\MedicionType;

use App\Controller\SearchPlanMedicion;

use App\Form\SearchPlantoMedicionType;

class MedicionController extends Controller

{

public function nuevoMedicion(Request $request)

{

// conecta con la base de datos y trae los datos del plan seleccionado

$em = $this->getDoctrine()->getManager();

$db = $em->getConnection();

$queryn = "SELECT \* FROM transito ORDER BY id DESC LIMIT 1;";

$stmtn = $db->prepare($queryn);

$stmtn->execute();

$pon=$stmtn->fetchColumn();

$plannumn = $this->getDoctrine()->getManager()->getRepository('App:Transito')->find($pon);

$num = $plannumn->getNumero();

$query = "SELECT \* FROM planif WHERE numero =".$num.";";

$stmt = $db->prepare($query);

$stmt->execute();

$po=$stmt->fetchColumn();

$plannum = $this->getDoctrine()->getManager()->getRepository('App:Planif')->find($po);

$n = $plannum->getNumero();

// obtiene de la base de datos la información que necesita del plan (cronograma, disparo, medición inicial)

$query1 = "SELECT \* FROM cronograma WHERE plan = ".$n.";";

$query3 = "SELECT \* FROM disparo WHERE plan = ".$n.";";

$query4 = "SELECT \* FROM medicion WHERE plan = ".$n.";";

$stmt1 = $db->prepare($query1);

$stmt3 = $db->prepare($query3);

$stmt4 = $db->prepare($query4);

$stmt1->execute();

$stmt3->execute();

$stmt4->execute();

$po1=$stmt1->fetchColumn();

$po3=$stmt3->fetchColumn();

$po4=$stmt4->fetchColumn();

$cronograma = $this->getDoctrine()->getManager()->getRepository('App:Cronograma')->find($po1);

$e = $plannum->getEstado();

$est = $plannum->getEstrategia();

$t = $plannum->getTipo();

$ap = $plannum->getActivopadre();

$ah = $plannum->getActivohijo();

$queryestado = "SELECT \* FROM estados WHERE numero = ".$e.";";

$stmtestado = $db->prepare($queryestado);

$stmtestado->execute();

$poestado=$stmtestado->fetchColumn();

$estado = $this->getDoctrine()->getManager()->getRepository('App:Estados')->find($poestado);

$queryestrategia = "SELECT \* FROM estrategias WHERE numero = ".$est.";";

$stmtestrategia = $db->prepare($queryestrategia);

$stmtestrategia->execute();

$poestrategia=$stmtestrategia->fetchColumn();

$estrategia = $this->getDoctrine()->getManager()->getRepository('App:Estrategias')->find($poestrategia);

$querytipo = "SELECT \* FROM mantenimientotipo WHERE numero = ".$t.";";

$stmttipo = $db->prepare($querytipo);

$stmttipo->execute();

$potipo=$stmttipo->fetchColumn();

$tipo = $this->getDoctrine()->getManager()->getRepository('App:Mantenimientotipo')->find($potipo);

$queryap = "SELECT \* FROM activopadre WHERE numero = ".$ap.";";

$stmtap = $db->prepare($queryap);

$stmtap->execute();

$poap=$stmtap->fetchColumn();

$actp = $this->getDoctrine()->getManager()->getRepository('App:Activopadre')->find($poap);

$queryah = "SELECT \* FROM activohijo WHERE numero = ".$ah.";";

$stmtah = $db->prepare($queryah);

$stmtah->execute();

$poah=$stmtah->fetchColumn();

$acth = $this->getDoctrine()->getManager()->getRepository('App:Activohijo')->find($poah);

$allreg = $this->getDoctrine()->getManager()->getRepository('App:Cronograma')->findBy(array('plan' => $cronograma->getPlan()));

$plannum1 = $this->getDoctrine()->getManager()->getRepository('App:Cronograma')->find($po1);

$plannum3 = $this->getDoctrine()->getManager()->getRepository('App:Disparo')->find($po3);

$plannum4 = $this->getDoctrine()->getManager()->getRepository('App:Medicion')->find($po4);

$salto = $plannum->getSalto();

$contador = $plannum->getIniciocontador();

$disparador = $plannum3->getConfiguracion();

// crea una nueva instancia de Medicion y crea el formulario

$medicion = new Medicion();

$form = $this->createForm(MedicionType::class, $medicion);

$form->handleRequest($request);

// establece en variales los puntos de medición (viejo y nuevo)

$puntonuevo = $form->get('puntonuevo')->getData();

$puntoviejo = $form->get('puntoviejo')->getData();

// si el formulario es valido, guarda los datos y procesa el trigger

if($form->isSubmitted() && $form->isValid())

{

foreach($allreg as $all)

{

$tarea = $all->getTarea();

$multreal = $all->getMultiplicadorreal();

$matriz = explode(',',$multreal);

for($p=0;$p<=29;$p++)

{

if($matriz[$p]==1)

{

$saltear = ($salto\*($p+1));

{

if($puntonuevo>=$saltear)

{

$matriz[$p]=0;

if($disparador==1)

{

$aviso = new Aviso();

$aviso->setEstado('5');

$aviso->setActivopadre($plannum->getActivopadre());

$aviso->setActivohijo($plannum->getActivohijo());

$aviso->setDescripcion($tarea);

$aviso->setCriticidad('1');

$aviso->setPlan($plannum->getNumero());

$em->persist($aviso);

$em->flush();

}

else

{

$orden = new Orden();

$orden->setEstado('5');

$orden->setTipo($plannum->getTipo());

$orden->setActivopadre($plannum->getActivopadre());

$orden->setActivohijo($plannum->getActivohijo());

$orden->setDescripcion($tarea);

$orden->setCriticidad('1');

$orden->setPlan($plannum->getNumero());

$orden->setCostoordentotal('0');

$em->persist($orden);

$em->flush();

$notificacion = new Notificacion();

$notificacion->setEstado('7');

$notificacion->setTipo($plannum->getTipo());

$notificacion->setActivopadre($plannum->getActivopadre());

$notificacion->setActivohijo($plannum->getActivohijo());

$notificacion->setDescripcion($tarea);

$notificacion->setCriticidad('1');

$notificacion->setPlan($plannum->getNumero());

$em->persist($notificacion);

$em->flush();

}

}

}

}

}

$var = implode(',',$matriz);

$all->setMultiplicadorreal($var);

$em->flush();

}

$plannum4->setPuntoviejo($puntoviejo);

$plannum4->setPuntonuevo($puntonuevo);

$em->flush();

// mensaje de exito

$this->addFlash('notice', 'Tus cambios se han guardado!');

}

// renderiza el formulario Medicion

return $this->render('medicion.html.twig', array('form' => $form->createView(), 'plannum' => $plannum, 'estado' => $estado, 'estrategia' => $estrategia, 'tipo' => $tipo, 'actp' => $actp, 'acth' => $acth));

}

}

Pruebas dirigidas por el diseño

10. Adoptar una “mentalidad de prueba” donde cada bug encontrado sea una victoria y no una derrota. Si se encuentra (y se arregla) el bug en la prueba, los usuarios no lo encontrarán en el producto lanzado.

9. Entender los diferentes tipos de prueba, y cuándo y por qué usaste cada una.

8. Cuando estés en la prueba de unidad, crear una o más pruebas de unidad para cada controlador en cada diagrama de robustez.

7. Para los sistemas de tiempo real, usar los elementos en los diagramas de estado como base para los casos de prueba.

6. Hacer una verificación a nivel de requerimiento (chequear que cada requerimiento que hayas identificado sea considerado).

5. Usar una matriz de trazabilidad para asistir en la verificación de requerimiento.

4. Hacer pruebas de aceptación para cada caso.Diseño de objetos dirigido por Casos de Uso Metodología ICONIX – v0.2

3. Expandir hilos en tus escenarios de prueba para cubrir un camino completo a través de la parte apropiada del camino básico más cada curso alternativo en tu prueba de escenarios.

2. Usar un framework de prueba (como JUnit) para almacenar y organizar tus pruebas de unidad.

1. Mantener tus pruebas de unidad bien detalladas.

* 1. Manual de Usuario

La documentación requerida se corresponde a Manual de Uso y Manual de Instalación.

* + 1. Manual de Uso

A continuación se describe el manual para el uso del sistema el cual también poder ser consultado con el servicio de Ayuda On Line.

**Operación: Menú de Gestiones.**

En el menú de gestiones, todos los usuarios pueden acceder a sus respectivas gestiones de acuerdo a los roles designados en el módulo de seguridad del sistema:

* Ejecutor: puede acceder a:
  + Gestión de Avisos de Mantenimiento.
  + Carga de Puntos de Medición.
* Planificador:
  + Gestión de Planes de Mantenimiento.
  + Lanzamientos.
* Supervisor:
  + Gestión de Ordenes de Trabajo.
  + Gestión de Equipos y Recursos Humanos.

**Operación: Gestión Manual de Avisos de Mantenimiento.**

Mediante esta operación el usuario puede registrar un aviso de mantenimiento donde debe indicar:

* Equipos (padre y/o hijo) que necesitan ser reparados o mantenidos.
* Descripción de la avería o falla.
* Criticidad del no atendimiento de la falla.

Además puede ver un listado de los avisos creados.

**Operación: Gestión Manual de Ordenes de Trabajo.**

El usuario de rol Supervisor puede ulitizar la función de creación de Ordenes de Trabajo la cual involucra costos de insumos y/o repuestos además de los recursos humanos necesarios para llevar adelantes las tareas. La orden de trabajo refleja que se va a hacer, como, donde, cuando y que costo tiene cada orden.

Durante la creación de la nueva orden se agina un equipo padre y/o hijo que serán mantenidos, fecha prevista, lista de recursos materiales, lista de recursos humanos y asociaciones a algún aviso de mantenimiento. Este último campo asocia una orden con un aviso gestionando el cierre automático del aviso cuando ocurra el cierre de la orden.

Además el usuario podrá ver el costo presupuestados de cada orden en el opción Ver Ordenes.

**Operación: Gestión de Planes de Mantenimiento.**

El usuario planificador puede:

Crear un nuevo plan de mantenimiento, indicando:

* Tipo de plan: preventivo o predictivo.
* Equipos padre y/o hijo.
* Estrategias de frecuencia: horas, distancia, volumen, tiempo calendario.
* Saltos de frecuencia.
* Configurar lista de tareas.
* Definir frecuencia de cada tarea según los saltos registrados.
* Disparador/Lanzador de plan indicando el tipo de configuración (autogeneración de avisos u órdenes de mantenimiento).

**Operación: Puntos de Medición.**

El usuario ejecutante debe registrar en la pantalla el valor actual del equipo padre/hijo según la unidad de medida establecida en la estrategia de frecuencias definida por el planificador en el plan de mantenimiento. Es obligatoriamente un valor numérico.

Dependiendo de lo registrado en el plan, el sistema dispara automáticamente ordenes o avisos según es configurado.

* + 1. Manual de instalación

No es necesario un proceso de instalación para usar GeMOL a través de la Web.

* 1. Patrones de Diseño

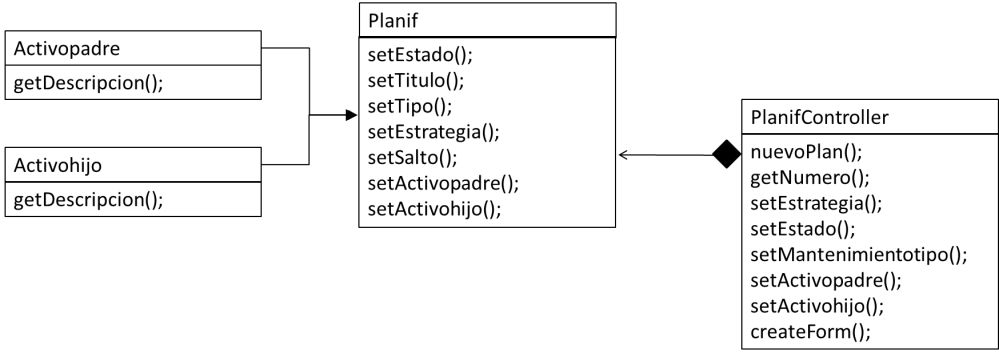
Un patrón de diseño debe cumplir al menos dos requisitos: debe ser efectivo, de modo que se haya podido comprobar su éxito resolviendo problemas anteriores; y debe ser reutilizable, es decir, se debe poder aplicar a problemas que se encuentren en circunstancias similares a las descritas por el patrón.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Se clasifican en tres tipos diferentes dependiendo del tipo de problema que resuelven. Estos pueden ser creacionales, estructurales y de comportamiento.

Se detalla un patrón de diseño utilizado en este proyecto:

|  |  |
| --- | --- |
| **Pattern** | **Activos** |
| **based on** | Strategy |
| **because** | **Funcionalidad:** Permite la selección de activos padre y/o hijo según su dependencia entre ellos. Tanto el activo padre como el hijo tienen más mismas características.  **Cambios Previstos:** La existencia de nuevos activos no debe afectar la creación del plan y sus derivados. |
| **Where** | Context **is** PlanifController  Strategy **is** Planif  ConcreteStrategyA **is** Activopadre  ConcreteStrategyB **is** Activohijo |



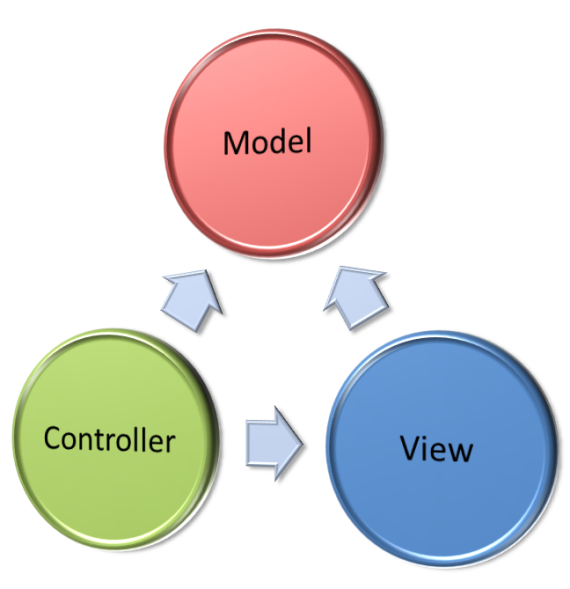


Además de eso, el FrameWork Symfony, embebe el patrón MCV dentro de su arquitectura.

El MVC (Model-View-Controller) es un patrón de diseño que en realidad ha existido desde hace algunas décadas, y se ha utilizado en muchas tecnologías diferentes.

El MVC separa la interfaz de usuario (UI) de una aplicación en las siguientes tres partes:

* **Model** - Un conjunto de clases que describe los datos que está trabajando, así como la lógica de negocio.
* **View** - Define cómo se mostrará la interfaz de usuario de la aplicación. Es un HTML puro que decide cómo la interfaz de usuario se va a parecer.
* **Controller** - Un conjunto de clases que maneja la comunicación del usuario, el flujo general de la aplicación, y la lógica específica de la aplicación.



ANEXOS

1. Contenido
   1. Normas de Calidad

Los estándares de calidad de software hacen parte de la ingeniería de software, utilización de estándares y metodologías para el diseño, programación, prueba y análisis del software desarrollado, con el objetivo de ofrecer una mayor confiabilidad, mantenibilidad en concordancia con los requisitos exigidos, con esto se eleva la productividad y el control en la calidad de software, parte de la gestión de la calidad se establecen a mejorar su eficacia y eficiencia.

En general, una vez validado que el sistema responde a los principales requisitos funcionales especificados, el usuario realizará las pruebas de aceptación, corrigiendo los errores encontrados y tas pasándose al fin del entorno de producción. Sin embargo, en muy pocas ocasiones se validan de manera rigurosa los requisitos funcionales y los no funcionales, o se ejecutan validaciones que aseguren que el sistema es lo suficientemente robusto y estable como para pasar a un entorno productivo con las garantías adecuadas.

**NORMAS ISO/IEC**

ISO 12207 – Modelos de Ciclos de Vida del Software.

Estándar para los procesos de ciclo de vida del software de la organización, Este estándar se concibió para aquellos interesados en adquisición de software, así como desarrolladores y proveedores. El estándar indica una serie de procesos desde la recopilación de requisitos hasta la culminación del software.

El estándar comprende 17 procesos lo cuales son agrupados en tres categorías:

* Principales
* De apoyo
* De organización

Este estándar agrupa las actividades que se pueden llevar a cabo durante el ciclo de vida del software en cinco procesos principales, ocho procesos de apoyo y cuatro procesos organizativos

Norma ISO/IEC 9126

La norma ISO/IEC 9126 de 1991, es la norma para evaluar los productos de software, esta norma nos indica las características de la calidad y los lineamientos para su uso, las características de calidad y sus métricas asociadas, pueden ser útiles tanto como para evaluar el producto como para definir los requerimientos de la calidad y otros usos.

Esta norma definida por un marco conceptual basado en los factores tales como Calidad del Proceso, Calidad del Producto del Software y Calidad en Uso; según el marco conceptual, la calidad del producto, a su vez, contribuye a mejorar la calidad en uso.

La norma ISO/IEC 9126 define la calidad en uso como la perspectiva del usuario de la calidad del producto software cuando éste es usado en un ambiente específico y un contexto de uso específico. Éste mide la extensión para la cual los usuarios pueden conseguir sus metas en un ambiente particular, en vez de medir las propiedades del software en sí mismo.

El modelo de la calidad en uso muestra un conjunto de 4 características: efectividad, productividad, integridad, y satisfacción.

Estándar ISO/IEC 14598

El estándar ISO/IEC 14598 es actualmente usado como base metodológica para la evaluación del producto software. En sus diferentes etapas, establece un marco de trabajo para evaluar la calidad de los productos de software proporcionando, además, métricas y requisitos para los procesos de evaluación de los mismos.

La norma define las principales características del proceso de evaluación

* Repetitividad.
* Reproducibilidad.
* Imparcialidad.
* Objetividad.

Para estas características se describen las medidas concretas que participan:

* Análisis de los requisitos de evaluación.
* Evaluación de las especificaciones.
* Evaluación del diseño y definición del plan de evaluación.
* Ejecución del plan de evaluación.
* Evaluación de la conclusión.

El estándar ISO/IEC 14598 define el proceso para evaluar un producto de [softwar](http://www.blogger.com/blogger.g?blogID=6627624266577841858)e, el mismo consta de seis partes:

* **ISO/IEC 14598-1** Visión General: provee una visión general de las otras cinco partes y explica la relación entre la evaluación del producto software y el modelo de calidad definido en la ISO/IEC 9126.
* **ISO/IEC 14598-2** Planeamiento y Gestión: contiene requisitos y guías para las funciones de soporte tales como la planificación y gestión de la evaluación del producto del software.
* **ISO/IEC 14598-3** Proceso para desenvolvedores: provee los requisitos y guías para la evaluación del producto software cuando la evaluación es llevada a cabo en paralelo con el desarrollo por parte del desarrollador.
* **ISO/IEC 14598-4** Proceso para adquirentes: provee los requisitos y guías para que la evaluación del producto software sea llevada a cabo en función a los compradores que planean adquirir o reutilizar un producto de software existente o pre-desarrollado.
* **ISO/IEC 14598-5** Proceso para avaladores: provee los requisitos y guías para la evaluación del producto software cuando la evaluación es llevada a cabo por evaluadores independientes.
* **ISO/IEC 14598-6** Documentación de Módulos: provee las guías para la documentación del módulo de evaluación.

Norma ISO/IEC 25000 (SquaRE)

ISO 25000:2005 (SQuaRE -Software Quality Requirements and Evaluation) es una nueva serie de normas que se basa en ISO 9126 y en ISO 14598 (Evaluación del software). Uno de los principales objetivos de la serie SQuaRE es la coordinación y armonización del contenido de ISO 9126 y de ISO 15939:2002 (Measurement Information Model).

ISO 15939 tiene un modelo de información que ayuda a determinar que se debe especificar durante la planificación, performance y evaluación de la medición. Para su aplicación, cuenta con los siguientes pasos: Recopilar los datos, Preparación de los datos y Análisis de los datos.

SQuaRE está formada por las divisiones siguientes:

* **ISO/IEC 2500n**. División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.
* **ISO/IEC 2501n**. División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.
* **ISO/IEC 2502n**. División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación.
* **ISO/IEC 2503n**. División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requisitos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requisitos de calidad para un producto software que va a ser desarrollado ó como entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288 (ISO, 2003).
* **ISO/IEC 2504n**. División de evaluación de la calidad. Estos estándares proporcionan requisitos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.
* **ISO/IEC 25050–25099**. Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software “Off-The-Self” y para el formato común de la industria (CIF) para informes de usabilidad.

**SPICE**

Es un estándar importante iniciativa internacional para apoyar el desarrollo de una Norma Internacional para la Evaluación de Procesos de Software. El proyecto tiene tres objetivos principales: Para desarrollar un proyecto de trabajo para un estándar para la evaluación de procesos de software. Para llevar a cabo los ensayos de la industria de la norma emergente. Para promover la transferencia de tecnología de la evaluación de procesos de software en la industria mundial del software a nivel mundial.

El estándar SPICE creciente en número de métodos de evaluación disponibles, y la creciente utilización de la técnica comercial en áreas sensibles, fueron los factores clave que impulsaron el desarrollo y la aceptación de una propuesta para desarrollar un estándar internacional para la evaluación de procesos de software.

Una Norma Internacional sobre Evaluación de Procesos de Software ofrecerá los siguientes beneficios a la industria y los usuarios del software: Beneficios para la Industria del Software Los proveedores de software se someterá a un solo esquema de proceso de evaluación.

Las organizaciones de desarrollo de software tendrán una herramienta para iniciar y sostener un proceso continuo de mejora. Los directores de programas tendrán un medio para garantizar que su desarrollo de software está en consonancia y apoya, las necesidades comerciales de la organización.

**CMMI**

Es un modelo de mejora de los procesos de construcción de software que provee los elementos necesarios para determinar su efectividad. Este modelo puede ser utilizado como guía para mejorar las actividades de un proyecto, área u organización, ya que proporciona un marco de referencia para evaluar la efectividad de los procesos actuales, facilitando con ello la definición de actividades, prioridades y metas para garantizar la mejora continua. Es el estándar más conocido para la mejora de procesos en mejora de procesos para el desarrollo de proyectos, gestión de proveedores y gestión de servicio.

El CMMI establece cinco niveles de madurez los cuales son:

Nivel 0: Incompleto El proceso no se realiza, o no se consiguen los objetivos.

* **Nivel 1 Inicial o ejecutando:** Este es el nivel en donde todas las empresas que no tienen procesos, es donde el proceso se ejecuta y se logra su objetivo, así sea fuera de presupuesto y de cronograma.
* **Nivel 2 Repetible:** Se da cuando el éxito de los resultados obtenidos se puede repetir.
* **Nivel 3 Definido:** Significa que la forma de desarrollar proyectos está definida, establecida, documentada y que existen métricas.
* **Nivel 4 Administrado:** Los proyectos usan objetivos medibles y cuantificables para alcanzar cubrir las necesidades de los clientes y la organización. Es decir, se usan métricas para gestionar la organización.
* **Nivel 5 Optimizado:** Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades, que mediante métricas son identificadas, evaluadas y puestas en práctica.

**IEEE (Institute of Electrical and Electronics Engineers)**

Es un método de establecimiento y mejora del trabajo en equipo para procesos software, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Su creación se remonta al año 1884, contando entre sus fundadores a personalidades de la talla de Thomas Alva Edison, Alexander GrahamBell y Franklin Leonard Pope. En 1963 adoptó el nombre de IEEE al fusionarse asociaciones como el AIEE (American Institute of ElectricalEngineers) y el IRE (Institute of Radio Engineers).

Según el mismo IEEE, su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales. Algunos de sus estándares son: VHDL, POSIX, IEEE 1394, IEEE 488, IEEE 802, IEEE 802.11, IEEE 754.

Mediante sus actividades de publicación técnica, conferencias y estándares basados en consenso, el IEEE produce más del 30% de la literatura publicada en el mundo sobre ingeniería eléctrica, en computación, telecomunicaciones y tecnología de control, organiza más de 1000 conferencias al año en todo el mundo, y posee cerca de 900 estándares activos, con otros 700 más bajo desarrollo.

**PSP**

El proceso personal del software es un método de autoconocimiento, que permite estimar cuánto se tarda un individuo en realizar una aplicación de software, para así calcular el presupuesto y asegurar la operatividad de los desarrollos. PSP se concentra en las prácticas de trabajo de los ingenieros en una forma individual.

El PSP se caracteriza porque es de uso personal y se aplica a programas pequeños de menos de 10.000 líneas de código. El PSP sirve para producir software de calidad, donde cada ingeniero debe trabajar en la necesidad de realizar trabajo de calidad.

**TSP**

Team Software Process es un método de establecimiento y mejora del trabajo en equipo para procesos de software. Es un proceso para equipos de software, a través del cual se contribuye equipos de alto rendimiento, capaces de comprometerse con el plan y administración del desarrollo de software, así como de producir productos de calidad y a bajo costo, logrando el mejor desempeño posible.

**Moprosoft**

Es una norma mexicana, basada en procesos para las industrias de software, la cual sirve para estandarizar operaciones y prácticas en gestión de ingeniería de software, para así elevar la capacidad de las organizaciones de ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad. Está enfocado a las Pymes de la Industria de Software en México. Está dirigido a las empresas o áreas internas dedicadas al desarrollo y/o mantenimiento de software.

A continuación se realiza un comparativo

|  |  |  |
| --- | --- | --- |
| Estándares y Normas | Organismo que regula | Aplicable a |
| CMMI | (SEI) Software Engineering Institute | Mejora de procesos de construcción de software y proyectos de TI. |
| PSP | ISO | Permite estimar cuánto se tarda un individuo en realizar una aplicación de software |
| PSP-TSP | ISO | Predice el tiempo y tamaño del software Administración de calidad |
| ISO 25000 | ISO | Establecen un modelo de calidad para el producto del software, además de definir la evaluación de la calidad del producto. |
| IEEE | IEEE | Serie de documentación para el desarrollo de software y proyectos de TI |
| TSP | Team Software Process | Es un método de establecimiento y mejora del trabajo en equipo para procesos de software |
| SPICE | Programa de simulación con énfasis en circuitos integrados | Es una importante iniciativa internacional para apoyar el desarrollo de una Norma Internacional para la Evaluación de procesos del software |
| Moprosoft | ISO | Norma mexicana, basada en procesos para las industrias de software, la cual sirve para estandarizar operaciones y prácticas en gestión de ingeniería de software |

Factibilidades

Factibilidad Técnica

Para que la implementación de GeMOL sea factible técnicamente los clientes deben contar con los siguientes recursos:

Hardware

* 2 estaciones de trabajo (mínimo), desktop o laptop de cualquier marca y modelo con 2 GHz de procesamiento o superior, 4GB Memoria RAM o más.
* 500 GB espacio en disco, recomendado SSD.

Software y Servicios

* Windows 10 Profesional o cualquier otro sistema operativo con entorno gráfico.
* Navegador Chrome, IE, o Firefox actualizado con sus complementos Java.
* Servicio de conexión a Internet 10MB como mínimo.

La empresa utiliza las siguientes tecnologías de software para el desarrollo de GeMOL.

* Framework Symfony versión 4.1.
* Lenguajes de programación PHP v7.0
* Servidor Web Apache y MySQL para acceso y manejo de datos.

* 1. Comercial

La factibilidad comercial de GeMOL se basa en los análisis de contexto y mercado realizados anteriormente. Teniendo en cuenta las fortalezas y amenazas del sector, se reconoce la oportunidad en el desarrollo de un negocio donde si bien existen soluciones similares, se detectan ventajas competitivas a explotar enfocadas hacia un nicho determinado.

GeMOL cuenta con recursos humanos capacitados y profesionales necesarios como para asegurar un producto y servicio final de primer nivel, que apunta a un mercado específico.

Administrativa

La empresa se definió en base a las necesidades primarias para el desarrollo, implementación y comercialización del producto.

Se mantiene una estructura de trabajo equilibrada que permite desarrollar nuevas funcionalidades al producto, estar preparados para el cambio, sostener una estrategia de marketing activa y otorgar soporte a los clientes que adquieran la solución.

En base a los requisitos planteados, se organiza la estructura en las siguientes gerencias:

* Ingeniería y Desarrollo e Investigación. Encargado de analizar el mercado y proponer mejoras e innovaciones al producto para mantenerse siempre en ventaja competitiva. Responsable de llevar adelante el análisis, desarrollo, pruebas e implementación de las distintas funcionalidades de GeMOL.
* Marketing. Encargado de establecer una relación a largo plazo con los clientes y comunicar el producto.
* Recursos Humanos. Responsable de las tareas administrativas, liquidación de sueldos y cargas sociales.

Los beneficios de esta estructura son:

* Definición de autoridad. En una estructura jerárquica, la escalera sube a una figura de autoridad superior. Dentro de la escala están los departamentos con funciones claras y sus gerentes. Con este tipo de estructura vertical, los empleados saben a quién informar los problemas.
* Los gerentes están capacitados en áreas específicas. Los gerentes en cada nivel son expertos en la función específica del departamento. Los gerentes con conocimientos en un área especializada guían a los empleados a niveles avanzados. Los empleados que sienten confianza en la competencia de sus directivos tiene un desempeño superior. De esta manera, los gerentes construyen un departamento fuerte que puede mejorar el éxito de la organización.
* Camino promocional transparente. Los empleados están en conocimiento del plan de carrera.
* Lealtad departamental. Los departamentos que trabajan para desarrollar un rol en común forman un sentido de camaradería. A medida que los empleados trabajan para desempeñar un papel específico en la organización, forman enlaces con los que contribuyen hacia el mismo resultado.

GeMOL fomenta la comunicación y trabajo en equipo como pilares fundamentales en su estructura organizativa.

Legal

GeMOL firma con sus clientes un Acuerdo de Confidencialidad, confeccionado para proteger la información que el cliente considere sensible.

Para cumplir con las disposiciones legales establecidas en el país, se pone en conocimiento al cliente sobre las implicaciones que lleva este desarrollo, a saber:

* LEY 24.766 Confidencialidad de la información: Se pone en conocimiento del cliente la vigencia de la mencionada ley, que encuadra la propiedad de los datos personales que se pudieran volcar eventualmente en el banco de datos que se desarrolla para la solución en una etapa posterior.
* LEY 25.326 Protección de los Datos Personales: Se pone en conocimiento del cliente la vigencia de la mencionada ley. Queda bajo responsabilidad de los representantes del organismo el registrar el banco de datos que se conforma en la solución.
* LEY 11.723 Régimen de la Propiedad Intelectual: Se pone en conocimiento del cliente la vigencia de la mencionada ley, que encuadra la propiedad intelectual del desarrollo.

En cuanto al registro del dominio WEB para GeMOL, se sigue la normativa vigente: Resolución 110/2016. Reglamento para la Administración de Dominios de Internet en Argentina. Aprobación.

GeMOL aparece en el mercado bajo una plataforma web sin la necesidad de establecer licencias de software en lo que la Ley respecta.

* 1. Bibliografía

1. Saporosi, Gerardo: “Clínica Empresaria” (Grupo Macchi, 1999).
2. Kothler, Philip; Armstrong, Gary: “Fundamentos de Marketing” (Pearson, 2008).
3. Laudon, Kenneth; Traver, Carol: “Comercio Electrónico: la revolución acaba de empezar” (Prentice Hall, 2008).
4. Gaitan, Juan J.; Pruvost, Andrés: “El comercio electrónico al alcance de su empresa” (UNL, 2001).
5. Hax, Arnoldo; Wilde II, Dean: “El proyecto Delta” (Norma, 2003).
6. David, Fred: “Conceptos de la administración estratégica” (Pearson, 2003).
7. Heerkens, Gary: “Administración de proyectos” (McGraw-Hill, 2002).
8. Moore, James: “Software Engineering Standards” (IEEE, 19989.
9. Ingenieria de Software - Ian Sommerville - 7ma Edicion - Otra versión
10. Patrones de Diseno - GOF [Gamma,Helm,Johnson,Vlissides]