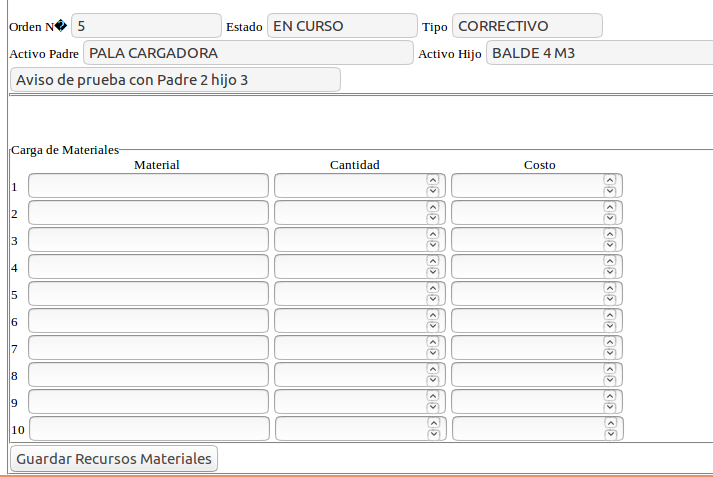
**CU03 – NOTIFICAR ORDEN DE MANTENIMIENTO - GUI**

Notificar confirmación de costos de orden de trabajo/Presupuesto.





* 1. **Especificaciones de Casos de Uso**

**CU03 – NOTIFICAR ORDENES DE TRABAJO**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Notificar Orden de trabajo** | | **Versión: 1.0** |
| ID de caso de uso: | CU-03 – Notificar Orden de trabajo | | |
| Creado por: | Martin Alice | | |
| Fecha de creación: | 24/10/2018 | | |
| Actor primario: | Supervisor de Mantenimiento | | |
| Actor secundario: | Analista Planificador | | |
| Descripción: | Permite notificar y confirmar la ejecución de las órdenes de trabajo detallando si las tareas, recursos materiales, recursos humanos fueron realizados de acuerdo a lo planificado, o no. En el formulario o pantalla de notificar orden de trabajo se deben setear los datos de equipos (padre y/o hijo), tareas realizadas, tipo de mantenimiento (correctivo, preventivo o predictivo), fecha programada de inicio y final tomados de una orden existente. A continuación se deben registrar los datos confirmados como ser, cantidad de horas hombre, categorias de personal utilzado para la tareas y en que proporcion, fechas reales de inicio y finalizacion de las tareas. Tiene que existir la posibilidad de notificar un valor parcial de la orden de trabajo. En este ultimo caso, la orden va a tomar un estado parcial el cual indica que esa orden aun no finalizo, pero parte de las tareas fueron realizadas y confirmadas por el Supervisor. En el caso de notificar la totalidad de una orden, la misma pasa a un estado cerrado donde se van a computar los costos, experiencias de equipo, horas de personal por categoria y otros posibles valores. Este caso de uso es fundamental para alimentar el calculo de algunos indicadores y metricas de gestion. | | |
|
|
|
|
|
| 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 Planificador 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.  Para notificar una orden, la misma debe existir y con un estado no cerrado. | | |
| Post – condiciones: | La notificación de una orden deja por sentado que el accionar del usuario es totalmente transparente y refleja la realidad de las cosas. | | |
| **Flujo Normal de Caso de Uso (Actor Principal)**  **Notificacion de Orden de Trabajo Total** | | | |
| **Usuario** | | **Sistema** | |
| Desde la pantalla de Notificar Orden de Trabajo: se debe seleccionar la orden afectada, confirmacion de tareas, recursos materiales, recursos humanos, y fecha de inicio y fin reales. | | El sistema guarda los datos ingresados y muestra un mensaje que los datos se guardaron correctamente. | |
| **Flujo Alternativo 1 de Caso de Uso**  **Notificacion de Orden Parcial (Actor Principal)** | | | |
| Desde la pantalla de Notificar Orden de Trabajo: se debe seleccionar la orden afectada, confirmacion de tareas, recursos materiales, recursos humanos, y fecha de inicio y fin reales. Indicar además que es un cierre de orden parcial. Si el usuario no lo especifica, el sistema entiende que la notificacion es total. | | El sistema guarda los datos ingresados y muestra un mensaje 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.

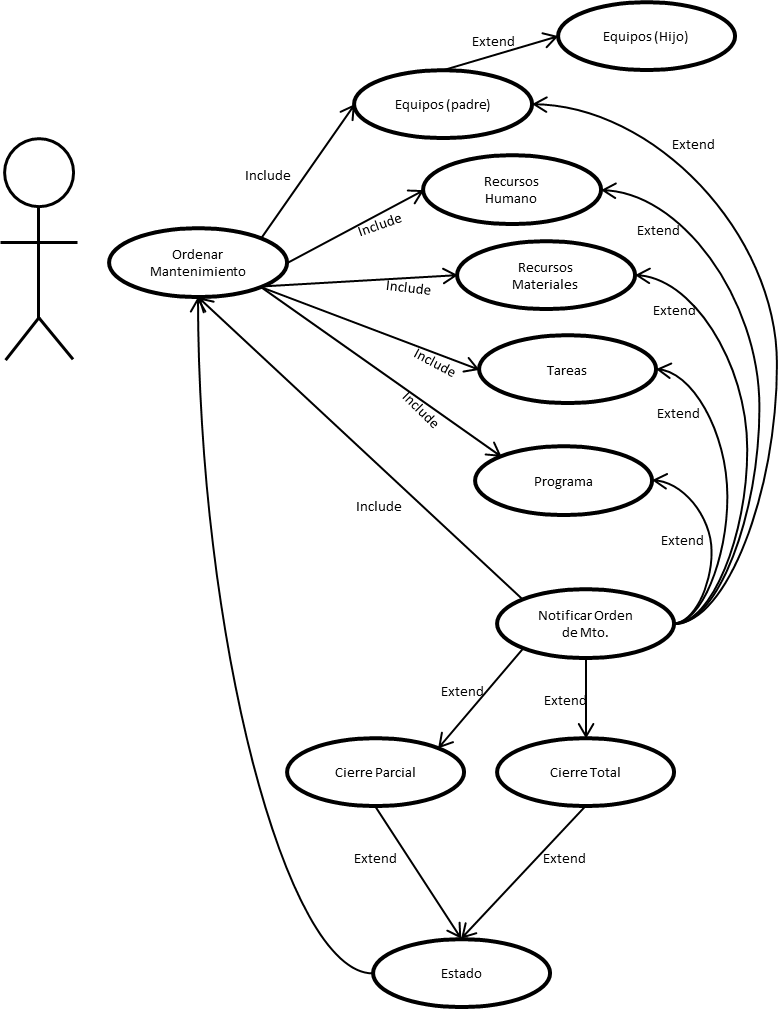
3. Crear estimaciones desde los casos de uso, NO desde los requerimientos funcionales.

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**

**CU03 – NOTIFICAR ORDEN DE TRABAJO**



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.

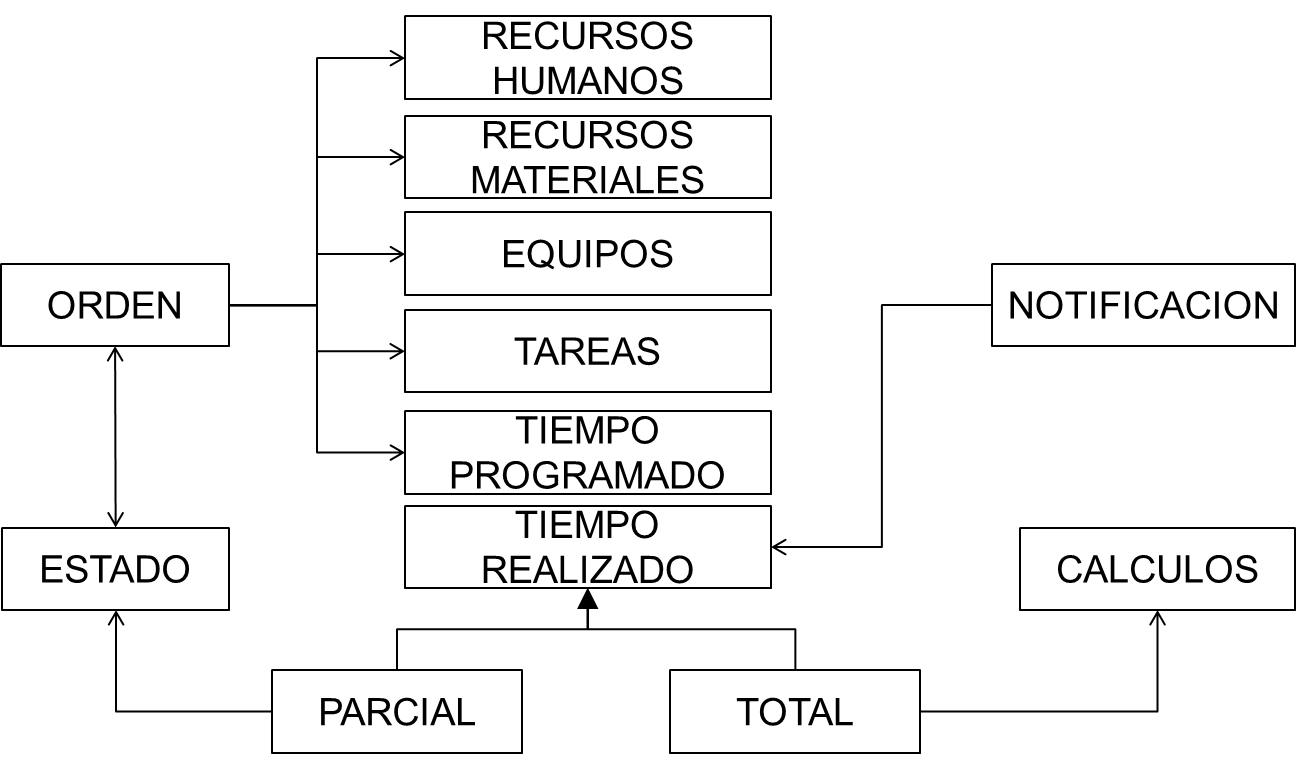
3. Escriba sus casos de uso usando una estructura de oración sustantivo-verbo-sustantivo.

2. Referencie las clases del dominio por nombre.

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

* 1. **Modelado del Dominio**

**CU03 – NOTIFICAR ORDEN DE TRABAJO**



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.

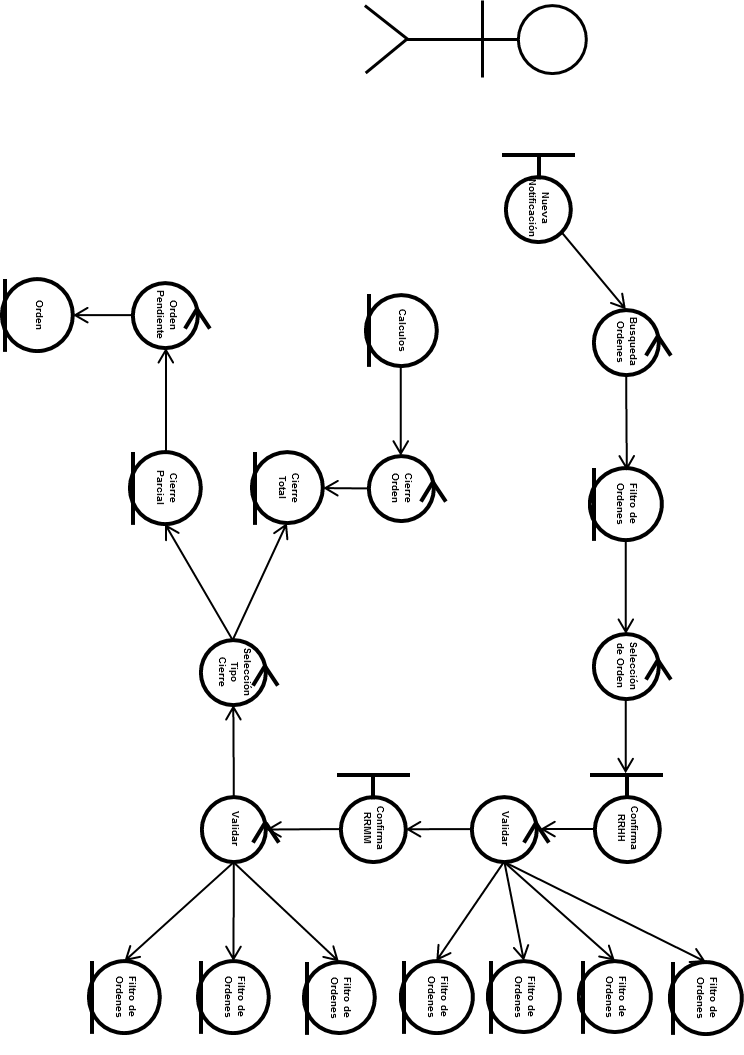
8. Limite los esfuerzos de modelado del dominio inicial a un par de horas.

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**

**CU03 – NOTIFICAR ORDEN DE TRABAJO**



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.).

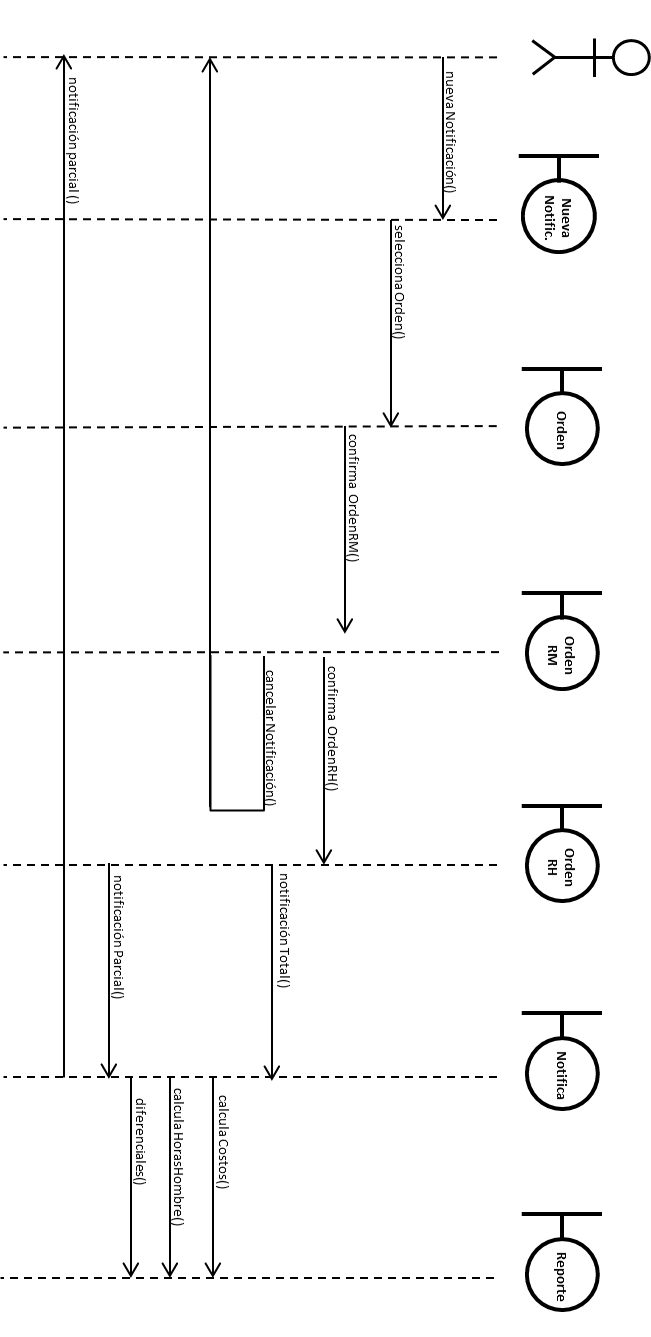
3. Verificar que los casos de uso estén en el contexto del modelo de objetos y en el contexto de la GUI.

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**

**CU03 – NOTIFICAR ORDEN DE TRABAJO**



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.

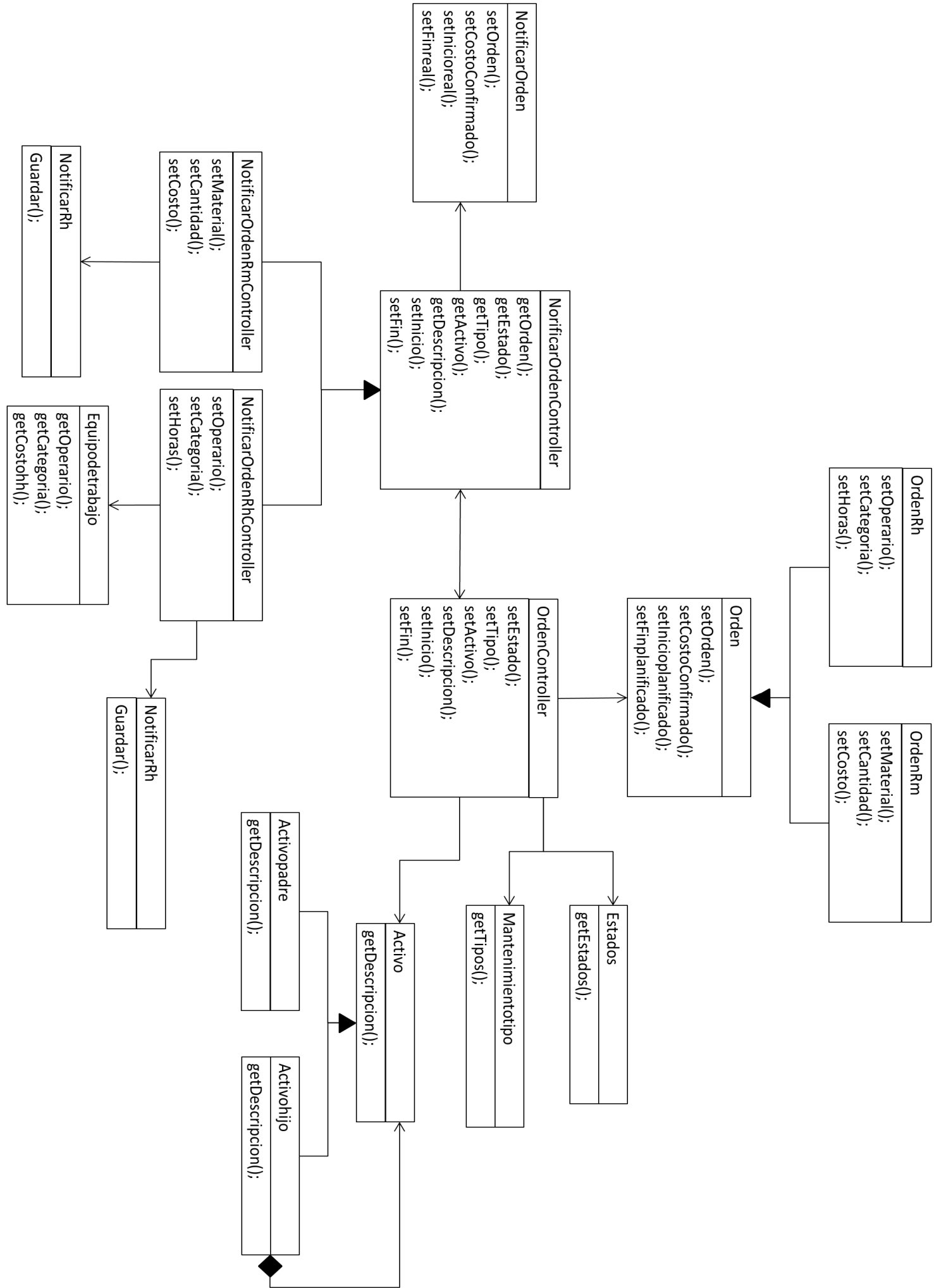
3. Revise su diagrama de clases frecuentemente mientras está asignando las operaciones a las clases, para asegurar que todas las operaciones están en la clase correspondiente.

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**

**CU03 – NOTIFICAR ORDEN DE TRABAJO**



* 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 detallan algunos de los patrones de diseño más utilizados en el proyecto:

|  |  |
| --- | --- |
| **Pattern** | **Ordenes** |
| **based on** | **Strategy** |
| **because** | **Funcionalidad:** Permite la herencia de interfaz y funciones de una orden de trabajo hacia los tipos de recursos que forman parte de la misma. Tanto los recursos materiales como los recursos humanos utilizan las mismas funciones que la orden y otras funciones más propias de cada clase.  **Cambios Previstos:** La existencia de nuevos recursos o funciones dentro de la orden de trabajo no debe afectar la creación de un plan, una orden y/o avisos de mantenimiento y sus derivados.  **Ventaja:** la ventaja de este patrón es que antes posibles cambios futuros en el negocio de la empresa, el sistema puede ser modificado sin necesidad de alterar el resto las funciones. |
| **Where** | Context **is** OrdenController  Strategy **is** Activo  ConcreteStrategyA **is** Activopadre  ConcreteStrategyB **is** Activohijo  Execute() **is** getOrden() |

