Universidad del Bío – Bío
Concepción

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Proyecto de Título para optar al grado de

Ingeniero de Ejecución en Computación e Informática

Desarrollo de Sistema De Apoyo a la Gestión de Inspección de Viviendas

"Home Inspector"

Jorge Del Rio Richter

Julio de 2009

Profesor guía: Sr. Manuel Crisosto Muñoz

Índice

Introducción

En el mercado de las inmobiliarias uno de los aspectos más relevantes es el tema de la calidad, como por ejemplo, saber el estado de lo que se compra, en que condiciones se encuentra la construcción, entre otros factores importantes. Este tema está muy en boga el día de hoy, debido a la mala calidad de las construcciones habitacionales que se está presentando en el mercado inmobiliario, a su vez, en el mercado de vivienda u oficinas usadas el cliente promedio desea saber con precisión y con causa de conocimiento el estado del producto que va a adquirir y descubrir si realmente es una inversión que agregue valor a su haber y no se convierta en sinónimo de gastos excesivos.

Pensando en estos problemas de calidad y el descontento del público, la empresa "Home Inspector" ha introducido en el país un novedoso sistema de inspección de propiedades, el cuál es realizado por especialistas en calidad de construcción. Esta idea es pionera en el país, sin embargo se basa en experiencias ya realizadas exitosamente en el extranjero, principalmente en Estados Unidos. El propósito de este servicio es entregar información confiable del real estado de una propiedad.

Para apoyar a la gestión de esta iniciativa es que "Home Inspector" ha decidido implementar un sistema para la automatización de herramientas de información para la inspección, con la finalidad de mejorar el servicio al cliente y la calidad del producto entregado, el cual para este caso es un completo informe estructurado en base a las cualidades o características de cada tipo de construcción.

Para llevar a cabo este nuevo sistema "Home Inspector" ha contratado los servicios informáticos de una nueva y creciente empresa de consultoría en Ingeniería de Software, "Agrega Ltda.", empresa con sede en Santiago de Chile. Esta empresa se hará cargo de la totalidad del proceso de desarrollo del nuevo sistema y estará a cargo de Gonzalo Tirapegui, jefe de Desarrollo e Innovación de la empresa.

En este informe se presenta paso a paso el proceso de desarrollo del sistema, considerando todos los aspectos que fueron revisados para la construcción del mismo.

Capitulo 1

Descripción de la Institución

Introducción al Capitulo

Este sistema tiene como objetivo mejorar los servicios de la empresa "Home Inspector", sin embargo su diseño, construcción e Implementación está a cargo de la empresa especialista en sistemas de Información "Agrega Ltda", es por esto que se realiza un presentación de ambas empresas, ya que el papel de cada una es indispensable para lograr el desarrollo del sistema y llevarlo a buen final.

1.1 El Desarrollador: "Agrega Ltda"

1.1.1 Antecedentes Generales

Agrega Ltda., es una empresa dedicada al desarrollo y gestión de sistemas de información, nace en el año 2008 a partir de una sociedad formada por Christian Power y Miguel Widoycovich, quien antes de integrar esta sociedad poseía una empresa que dirigía sólo, esta era "SH Systems Ltda.". Esta empresa consta de dos departamentos principales, Comercial y Operaciones. La principal función es el desarrollo de nuevos sistemas para empresas del país, así como también entregar soporte informático a empresas que lo requieran.

1.1.2 Estructura Organizacional y Funciones

En la figura 1.1 muestra el Organigrama Informal de la empresa Agrega Ltda.

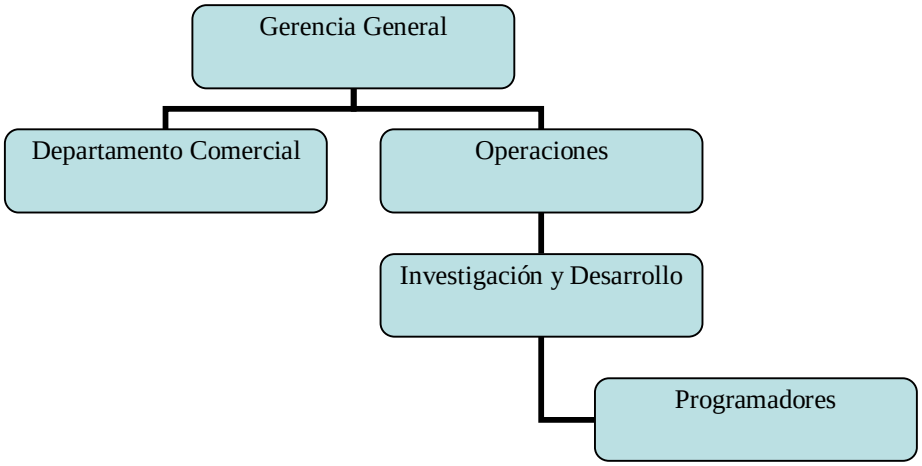


Figura 1.1 Organigrama Agrega Ltda.
Fuente: Agrega Ltda. (1)

(1) *Toda la información capturada de Agrega Ltda. fue dada por Gonzalo Tirapegui, socio de la empresa.*

Las Funciones de los departamentos que conforman el organigrama de la empresa son las siguientes:

Gerencia General: Este departamento es dirigido por Miguel Widoycovich, gerente general y fundador de Agrega Ltda.

Este departamento se encarga de la dirección comercial y operativa de la empresa.

Departamento Comercial: Departamento encargado de llevar toda la gestión contable y comercial de la empresa. Esta oficina es dirigida por Miguel Widoycovich. Todo el control contable, así como la búsqueda de nuevas oportunidades de negocio y la mantención de los ya existente están a cargo de esta oficina.

Departamento de Operaciones: Este departamento centra su tarea en la gestión de los proyectos adjudicados, estando a cargo de la evaluación y el desarrollo de cada proyecto. El departamento de operaciones está dirigido por el Ingeniero en Informática Christian Power, quién además es socio fundador de la empresa.

Bajo el alero del departamento de Operaciones se encuentra el Departamento de Investigación y Desarrollo, el cual está a cargo de Gonzalo Tirapegui, Ingeniero en Computación e Informática, quien además es socio de la empresa. Este sub-departamento tiene entre sus funciones el desarrollo de nuevos sistemas y el de investigación de tecnologías de Información, tanto innovadoras como ya existentes. Bajo este sub-departamento se encuentra el equipo de programadores y diseñadores de sistemas.

Misión

Entregar a sus clientes las mejores soluciones informáticas para solucionar sus problemas, con seriedad y respaldo, para así agregar valor apreciable al negocio de cada cliente.

Visión

Llegar a ser a mediano plazo, una de las empresas informáticas destacadas de nuestro país, por integrar en sus herramientas ideas innovadoras que tengan como resultado el agregar valor considerable a cada negocio atendido, así como también ser reconocidos como una empresa pionera y vanguardista en la investigación y el desarrollo de nuevas tecnologías de información.

Fuente: Agrega Ltda.

1.2. El Cliente "Home Inspector"

1.2.1.- Antecedentes Generales

Home Inspector es una empresa dedicada a la inspección de viviendas de todo tipo (Departamento, Casas, Oficinas), en la cual se determina la calidad de éstas, basadas en estándares de inspección determinados según el tipo de vivienda. Esta está ubicada en Av. Einstein 1043, Recoleta, Santiago.

Esta empresa nace a partir de la experiencia Norteamericana en la inspección inmobiliaria, quienes fueron pioneros en el tema de la inspección de calidad.

La empresa es dirigida por don Andrés Aninat, y está formada por un grupo de Arquitectos y constructores. Además posee un equipo de 10 inspectores, quienes poseen alguna de las profesiones mencionadas y además son expertos en calidad inmobiliaria y construcción. Estos inspectores realizan inspecciones en terreno, en donde detallan diversos aspectos de las viviendas, tomando notas del estado de cada uno de ellos.

Fuente: Agrega Ltda.

La empresa Home Inspector actualmente entrega los siguientes servicios a sus clientes:

- ITP: Inspección técnica de propiedades:
  Es un informe técnico para propiedades usadas, dirigido a vendedores y compradores, que mediante un riguroso protocolo de inspección permitirá conocer el estado real de la propiedad, además de conocer los costos de reparación involucrados en las fallas observadas.

- IRP: Inspección para recepción de propiedades:
  Es un informe técnico para propiedades nuevas, dirigido a compradores, el cuál permitirá conocer que elementos de la vivienda deberán ser reparados por la empresa constructora.

Para entregar un buen servicio a sus clientes, Home Inspector tiene la siguiente estructura para inspección, la cual atiende al protocolo de Home Inspector, esto se ve en la Tabla Nº 1:

Estructura y Terminaciones Exteriores	Cimientos Pilares Vigas Losas Entrepisos Tabiques Revestimientos exteriores Pavimentos Cierros perimetrales	**Techumbre**	Cerchas Estructuras Sellos Aislamiento Canaletas Bajadas de agua Otros
Interior	Puertas Ventanas Muros Revestimientos Pavimentos Cielos Detalles ornamentales Closet Muebles de cocina Otros	**Instalaciones Sanitarias**	Estado general Presión de agua Filtraciones Evacuaciones
Electricidad	Medidor Tablero general Circuitos Enchufes Interruptores Centros Otros	**Gasfitería**	Medidor Arranques Llaves de paso Posibles fugas Medición de Co2 Estado de calefont Otros
Calefacción	Operatividad de sistemas Eficiencia Otros	**Obras Complementarias**	Estacionamientos Bodegas Equipamiento de seguridad Espacios comunes Otros

Tabla Nº: Estructura de Inspección

Además de de la inspección ocular basada en los conocimientos y experiencia de cada inspector, la empresa pone a disposición un conjunto de herramientas especializadas, estas son:

- Detector de fuga de gases: Herramienta que tiene la finalidad de percibir emanaciones de gas, tales como gas natural, gas butano, etano, gasolina, entre otros.

- Medidor de monóxido de carbono: Esta herramienta tiene la finalidad de captar cualquier emanación nociva de gas monóxido de carbono.

- Medidor y Tester de Instalaciones Eléctricas: Con esta herramienta el inspector verifica la puesta a tierra, tensión de la línea, capacidad de los conductores, carga a los que son sometidos y la existencia de dispositivos de seguridad.

- Medidor de Presión y Termómetro: A través de esto es posible medir la presión del agua potable y la oscilación de temperaturas entre agua fría y caliente.

Para la realización de su producto, "Home Inspector" dispone de ciertas consideraciones generales, las cuales son puestas a disposición del cliente y son importantes para determinar el alcance de este sistema. A continuación se muestran estas consideraciones Generales:

1. El informe **Home Inspector (HI)** se basa en la inspección ocular de los elementos, componentes y sistemas integrados en un bien raíz. Serán sometidos a ésta, todos aquellos elementos a los que se pueda acceder libre y fácilmente, para poder determinar, en el momento de la visita, si cumplen con la función para la cual fueron diseñados.
2. El objetivo del inspector es identificar fallas o defectos que afecten la integridad o funcionalidad de los elementos y sistemas que componen la propiedad.
3. El objetivo del informe es declarar los problemas que puedan afectar la habitabilidad y seguridad de sus ocupantes, o eventuales habitantes, en relación a los elementos y sistemas que componen la propiedad.
4. Se excluye de la inspección, elementos ocultos o de difícil acceso. El inspector no tiene como obligación desplazar objetos, muebles, tabiques, cielos falsos, pisos u otros que dificulten la inspección visual de la vivienda, así como tampoco realizar pruebas que dañen o destruyan los elementos a evaluar.

5. Quedan excluidos de la revisión: artefactos, sistemas de seguridad (incluidas alarmas), sistemas de corrientes débiles, sistemas de riego automático, sistemas de radio o control remoto.
6. El inspector queda excluido de ingresar a propiedades que impliquen riesgo físico evidente, en general todas aquellas áreas que tengan riesgo de colapso (techumbres u otros) o de ocasionar daños a terceros.
7. El informe no incluye información referente a situaciones previas de emplazamientos, como análisis geológico, medio ambientales, residuos o desechos tóxicos, etc.
8. El informe no hace referencia a normativas o leyes tales como la Ordenanza General de Urbanismo y Construcción ni ordenanzas Municipales.
9. El informe no es una tasación, por lo que no hará comentarios relacionados con valores de mercado de la propiedad, plusvalías o consideraciones similares.
10. El inspector no hará referencia acerca de información topográfica, tales como establecer los límites de la propiedad, dimensiones del predio, cotas del mismo, expropiaciones existentes, ni superficies de edificación. Éstas últimas podrán ser tomadas sólo en caso de tratarse de una propiedad industrial, y únicamente cómo medio usado para determinar los honorarios correspondientes al informe.
11. El inspector no esta obligado a inspeccionar elementos o equipamiento común a un condominio o edificio.
12. El informe no debe ser interpretado como garantía o seguro.
13. El informe no garantiza el funcionamiento u operatividad futura de aquellos componentes en los que no se ha observado evidencia de desperfectos, fallas o malfuncionamientos.
14. El informe presentado es para uso exclusivo del cliente interesado y se considera como información confidencial.

Fuente: www.homeinspector.cl (2)

(2) Página Web comercial de la empresa Home Inspector.

1.2.2 Estructura Organizacional y Funciones

El Organigrama informal de la empresa "Home Inspector" es el que se muestra en la figura 1.2:

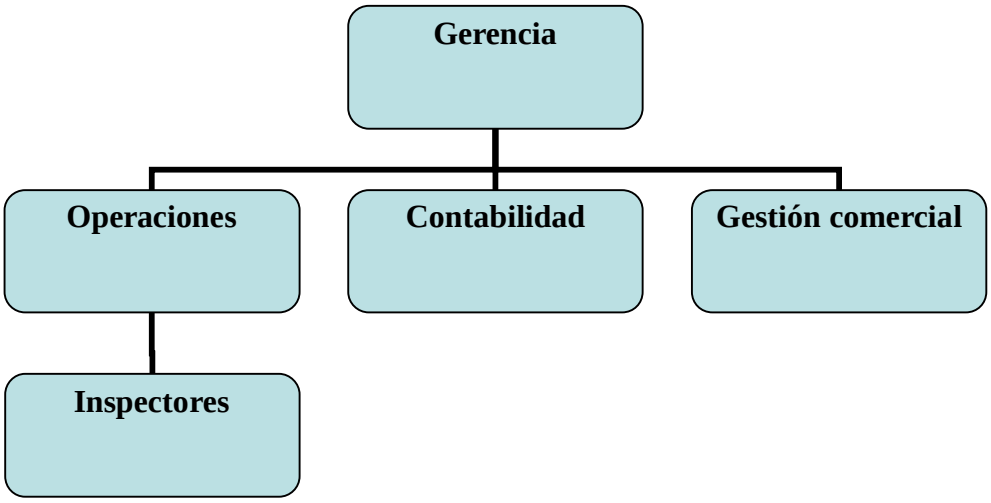


Figura 1.2: Organigrama Home Inspector.
Fuente: Agrega Ltda.

Las funciones de los departamentos identificados en el organigrama son:

Gerencia: Su principal función es llevar toda la dirección de la empresa, es dirigida por Andrés Aninat.

Operaciones: Departamento encargado de la producción, controlar y cumplir el estado de las inspecciones y entrega de resultados.

Contabilidad: Departamento encarado de llevar a cabo el control financiero de la empresa.

Comercial: Departamento encargado de los negocios de la empresa, encontrar nuevos clientes y mantener las cuentas de los ya existentes.

Fuente: Agrega Ltda.

Capitulo 2

Descripción del proyecto

2.1 Descripción de la Situación Actual

Actualmente "Home Inspector" mantiene un equipo de profesionales altamente calificados en el área de construcción, especializados en control de calidad, quienes hacen el papel de "Inspectores", ya que son los encargados de concretar inspecciones a terreno con el fin de determinar el estado actual de la propiedad correspondiente. Para realizar estas inspecciones inmobiliarias, los inspectores van a la propiedad que debe ser inspeccionada y examinan el estado de los siguientes aspectos de la construcción:

- Estructura y terminaciones exteriores
- Techumbre
- Interior
- Instalaciones Sanitarias
- Electricidad
- Gasfitería
- Calefacción
- Obras complementarias.

Para apoyar su labor de inspección y realizar un trabajo completamente profesional, y por ende confiable, los inspectores, además de su conocimiento se poyan de las siguientes herramientas:

- Detector de fuga de gases combustibles
- Medidor de monóxido de carbono
- Medidor y Tester para instalaciones eléctricas
- Medidor de presión y termómetro.

No existe ningún tipo de software especializado en la empresa que apoye el proceso de inspección, por esto, los inspectores al ir chequeando uno a uno el estado de estos puntos poseen variadas formas de registrar lo observado, apoyándose principalmente en los siguientes elementos:

- Hojas de Calculo de Excel
- Grabadoras digitales
- Cámaras fotográficas
- Cuadernos de notas.

El inspector irá registrando las características o estado de lo observado según sea su disponibilidad y/o comodidad, algunos prefieren anotar en papel para luego traspasar a hojas de cálculo, o bien realizar todo en archivos de audio y luego realizar un informe en Excel, finalmente también está la posibilidad de hacerlo directamente en hojas de cálculo utilizando dispositivos móviles como Palm, netbook o similares.

Una vez recopilados los datos, se construye un informe en Word u otro procesador de textos, a partir de lo estampado en los ya mencionados libros de Excel. Este informe de Word muestra una lista los ítems inspeccionados, separándolos por categorías, lo que se llama sección, indicando el estado de cada ítem de dicha sección. Este informe se construye en base a un formato estándar implementado por la empresa.

2.2.- Descripción del Problema

El sistema actual de inspección presenta algunas deficiencias, las que son principalmente provocadas por la falta de automatización del sistema de inspección. Esto provoca problemas como los siguientes:

- Si bien hay un estándar para la creación del reporte final, la recopilación de datos se realiza de forma manual y desordenada, en donde no hay una forma única para realizar las anotaciones de observación, lo que se resume en inconvenientes para la recopilación final de los datos, por ejemplo para definir el estado de una viga, cada inspector puede catalogar un determinado nivel de diferentes formas queriendo expresar lo mismo, por ejemplo bueno, aceptable, muy bueno, excelente, perfecto, etc.
- Estos datos se traspasan a una hoja de cálculo de Excel, lo que no presenta seguridad alguna en la mantención de los mismos.
- La realización de los informes finales de inspección se hace en Word o procesador de textos similar, de forma manual, en donde se debe mantener la estructura del informe, además de esto y el tiempo de digitación hacen que la realización del informe sea demasiado lenta.
- Desde el momento de la inspección hasta la entrega final, los datos pasan por demasiadas manos, lo que finalmente puede llegar a causar inconsistencias en el informe final.
- El tiempo de entrega de resultado desde la inspección hasta el resultado final es demasiado largo, lo que provoca que la espera por parte del cliente sea demasiado larga, generalmente el cliente espera rapidez por parte de los servicios que contrata. El tiempo actual es de 5 a 7 días aproximadamente.
- Todos los datos que utiliza la empresa (clientes, viviendas, valores) son almacenados en hojas de cálculo Excel, por lo que la velocidad de acceso, la confianza y la seguridad de los datos es mínima.
- Además de lo anterior, no existe la centralización necesaria de los datos.
- No existe forma alguna de acceder a los informes y datos de forma remota, por lo que toda la información está delimitada a ser vista sólo en las dependencias físicas de la empresa.

Ejemplo de planilla Excel usada:

ESTRUCTURA y TERMINACIONES EXTERIORES
4. FACHADAS Y MUROS

A. FACHADA FRENTE
B. FACHADA LATERAL IZ.
FACHADA LATERAL
C. DER.
D. FACHADA TRASERA

Ubicación:	**norte**
	sur
	este
	oeste
ELEMENTOS	
Cimientos	
Sobre cimientos	
Pilares	
Fundaciones	
Vigas	
Rellenos	
Materialidad Estructura :	albañilería armada
	albañilería reforzada a la vista
	madera
	hormigón
	Metalcom
	madera
	bloques cemento
	Hormigón celular
	Covintec
	Estructura Metálica
	otro (especificar)
Revestimientos	estuco liso
	estuco texturado
	cerámica
	ladrillo
	piedra
	piedra reconstituida
	wetproof
	internit
	pvc
	zinc ondulado tipo Hunter Douglas
	Hunter Douglas
	madera
	pintado

	barnizado	
	otro (especificar)	
Estado general:	**BUENO**	Nota: completamente funcional
	REGULAR	Nota: medianamente funcional
	MALO	Nota: no cumple su función
Falla 1 (ídem Falla 2 - 3-4):	agrietado	
	trizado	
	partido	
	desprendido	
	desgastado	
	desnivelado	
	picada	
	podrido	
	suelto	
	quebrada	
	elemento faltante	
	material insuficiente	
	filtración	
	humedad	
	desaplomo	
	desprendimiento	
	descomposición	
	plagas	
	desasentado	
	craquelado	
	descascarado	
	oxidado	
	otro (especificar)	
Origen:	falta de mantención	
	ejecución de la obra deficiente	
	especificaciones técnicas insuficientes	
	uso incorrecto	
	desgaste natural	
	otro (especificar)	
Urgencia:	ALTA	
	MEDIA	
	BAJA	
Implicancia:	mayor deterioro	
	colapso de estructuras	
	perdida de funcionalidad	
Tamaño de la falla:	☐ m2 x	☐
Costo de reparación aproximado		☐ pesos

FOTOS

Contexto	☐
Detalle	☐

2.3.- Solución Propuesta

Este tipo de empresas abre un nuevo mercado en el país, y relativamente nueva en el extranjero, por lo que no existe referencia a sistemas de apoyo a la gestión para la inspección de viviendas, por lo que se descarta la compra de un software que satisfaga las necesidades de información de la empresa.

Por lo anterior la única solución factible encontrada es la que se presenta a continuación:

- Como solución al problema se recomienda la construcción de un sistema de información a medida, que automatice los procesos o apoye la gestión de producción de la empresa.

Esto implica la construcción de un sistema de información, basado en plataformas Web, utilizando herramientas de última generación, funcionando sobre una base de datos centralizada que contenga la totalidad de los datos de la empresa relacionados al proceso productivo, preparado además para ser tan flexible como la empresa lo requiera.

2.4.- Aporte o Beneficio

Este nuevo sistema mejorará considerablemente el proceso de producción de “Home Inspector”, agregando valor considerablemente al producto entregado por la empresa. Este sistema está pensado para trabajar sobre la base de un almacén de datos centralizado, esto mejorará indudablemente la confiabilidad de los datos ya que éstos al estar en una base de datos centralizada serán totalmente consistentes. Además de los datos para el cliente correspondientes al producto, están los datos utilizados por la empresa, ya sea registros de clientes, propiedades, etc. Logrando que las búsquedas de clientes y sus respectivas propiedades, sea mucho más efectiva y segura.

Otro punto en donde se hará diferencia es en los tiempos de entrega, ya que el informe será generado en tiempo real, a sólo un click después de finalizada la inspección, ya que el registro de observaciones realizadas por los inspectores se podrá hacer vía Web desde cualquier lugar, a través de un formulario dispuesto para esto y que podrá ser accesible desde cualquier navegador básico. Además de la información en el formulario vía Web, el desarrollo del informe final, actualmente hecho a mano en procesador de textos, se realizará de forma automática, reduciendo el tiempo del informe final al tiempo utilizado por el inspector encargado.

La característica anterior permitirá además controlar el trabajo en terreno de los inspectores, como por ejemplo los tiempos de traslado y de inspección del inmueble.

En resumen el sistema beneficiará en producción, seguridad, confiabilidad y desempeño al proceso productivo del principal servicio de "Home Inspector".

2.5 Descripción del sistema a Desarrollar

El sistema a desarrollar será una aplicación Web, la cual apoyará el proceso de inspección de inmuebles. Estará basado en mantenedores y generadores de reportes, construido en base a tecnologías de última generación y siguiendo estándares de diseño y programación que harán del software una herramienta robusta y completamente apto para el control de versiones, ya que una de sus principales características es que estará basado en el paradigma Orientado a Objetos (OO) y en el modelo de desarrollo Modelo-Vista-Controlador (MVC), el cual permite manejar la interacción con los datos, la vista del usuario y la lógica de negocios de forma completamente independiente cada una de la otra, esto hace que cualquier futura modificación o mejora sea fácil de lograr, además de permitir al desarrollador un mejor entendimiento del sistema.

Este sistema además presentará al usuario una interfaz sencilla y amigable, permitiendo que el usuario con un mínimo de conocimientos sea capaz de utilizar el software en su totalidad rápidamente, implicando un costo de tiempo de inducción mínimo.

Además de su interfaz sencilla, el sistema estará construido de tal forma que no sea impedimento acceder a sus módulos a través de dispositivos con navegadores básicos, como por ejemplo los dispositivos móviles de menor dimensión y capacidades, como por ejemplo Palm, Iphone y similares.

2.6.- Objetivos del sistema.

Objetivo General:

Presentar al usuario una plataforma Web que permita la automatización de los procesos de producción de "Home" Inspector, a través de un conjunto de módulos para el registro y edición de información utilizada por la empresa para la inspección inmobiliaria.

Objetivos específicos:

- Entregar acceso al sistema desde cualquier lugar a través de la red mundial Internet.
- Mantener los datos/información de "Home Inspector" en un solo lugar físico y no repetidos (base de datos centralizada)
- Gestionar Usuarios (Inspector, Administrador, Cliente).
- Gestionar Propiedades.
- Gestionar Órdenes de trabajo.
- Gestionar Plantillas para inspección y su respectiva estructura.
- A través de lo anterior imponer un estándar de revisión.
- Implementar módulo para la gestión dinámica de plantillas.
- Implementar formularios de inspección basados en plantillas.
- Entregar acceso a formularios de inspección en tiempo real.
- Emitir los informes finales de inspección.

2.7.- Alcances y Limitaciones.

- El sistema no ofrecerá efectos visuales complicados para la capa de usuario basados en JavaScript (JS), con el fin de ofrecer acceso a navegadores básicos.

- El sistema no tendrá interacción con ningún otro software existente sea ajeno o no al propósito del mismo.

- El sistema no contempla el uso de información para fines de tipo contable, por lo que no se consideran módulos de este tipo.

- El sistema tiene como único propósito el apoyo al proceso productivo, dejando de lado cualquier índole a Recursos Humanos y Financieros.

- El sistema no contempla la entrega de informes estadísticos de ninguna índole.

- El sistema permitirá el acceso sólo a usuarios identificados en él, ingresados previamente por el Administrador, quien a su vez será el único autorizado a hacer esto.

- El sistema no contempla la visualización ni modificación de la ubicación física de la propiedad (mapa).

- El sistema no contempla el manejo de lotes de viviendas.

- El sistema no contempla la deducción automática de los costos de reparación en base a tablas de evaluación de daños puesto que la estandarización de estos datos tomaría un tiempo mayor al estimado para el desarrollo, sin embargo el modelamiento de la información y de la implementación debe permitir el acoplamiento de esta funcionalidad en una segunda fase teórica.

2.8.- Aspectos Generales de la Metodología a Utilizar.

RUP

La Metodología de desarrollo a utilizar será RUP, la cual consiste en separar el desarrollo en 4 etapas, las cuales son:

- *Inicio*, El Objetivo en esta etapa es determinar la visión del proyecto.
- *Elaboración*, En esta etapa el objetivo es determinar la arquitectura óptima.
- *Construcción*, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- *Transmisión*, El objetivo es llegar a obtener el producto final.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.
Esto se logra a través de las siguientes tareas:

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

UML (Lenguaje de Modelado Unificado)

Para apoyar el diseño del sistema se utilizará UML, específicamente algunas herramientas que se describirán a continuación.

UML es una técnica de modelamiento creada con el fin de ser un lenguaje estándar, que reuniera las mejores características de los diversos sistemas existentes. Básicamente es una colección de diagramas destinados a una mejor comprensión y creación de modelo de un sistema, proceso de negocios o sistemas no-software. Para el desarrollo de este sistema se utilizan los Diagramas de Casos de Uso, Clase y Secuencia.

- Diagramas de Casos de Uso

Se usan específicamente en la especificación de requerimientos. Muestran las instancias en las cuales un Actor (usuario del sistema) hace uso del sistema.

- Diagramas de Actividad

Muestran la interacción del actor (usuario) y sistema, identificando las actividades realizadas por cada uno.

- Diagramas de Secuencia

Son para graficar la forma lógica u orden en que un Caso de Uso es ejecutado, mostrando los mensajes y datos que se traspasan entre distintas clases del sistema.

MER

Para el diseño del modelo de datos se utilizará MER (Modelo Entidad Relación) .

MER es una representación gráfica de los datos que necesitan ser usados por el sistema. Se usaran los de tipo lógico y físico.

El primero es una representación lógica de cómo se distribuyen las entidades que interactúan en el sistema y sus datos. El segundo, confeccionado a partir del anterior, contiene las tablas finales de la base de datos, incluyendo las que son agregadas tomando en cuenta el modelo lógico.

Patrón de Diseño

Para desarrollar este sistema se utiliza uno de los patrones de diseño mas valorados en el actual escenario del desarrollo de software, es el Modelo Vista Controlador, debido a que separa completamente la vista del sistema (la capa de usuario) con el modelo de negocios y los controladores del mismo. Esto permite que tanto el desarrollo como una posterior mantención del sistema se haga sin cambiar nada más que lo que se necesita.

Los elementos de este patrón son:

1. El **modelo** es el responsable de:
   - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
   - Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
   - Lleva un registro de las vistas y controladores del sistema.
   - Si está ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc.).

2. El **controlador**
   - Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
   - Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_orden_de_venta)".

3. Las **vistas** son responsables de:
   - Recibir datos del modelo y los muestra al usuario.
   - Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
   - Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Nota: Todos estos tópicos se encuentran detallados en su totalidad en el Anexo Nº 1 añadido al final de este informe. Las fuentes de información se encuentran en el respectivo anexo.

2.9 Tecnología a Utilizar.

La tecnología utilizada para la construcción de este software será en base a un conjunto de herramientas de libres distribución, las cuales poseen características de última generación y obedecen a estándares orientados al desarrollo de aplicaciones empresariales bajo el paradigma orientado a objetos. Estas herramientas son:

- NetBeans: IDE de programación, este IDE fue seleccionado debido a la amplia gama de posibilidades y/o características que posee, además es mantenido por la misma empresa creadora del lenguaje JAVA y del estándar J2EE (Sun Microsystems).

- Lenguaje de Programación JAVA: Lenguaje desarrollado por Sun Microsystems. Se eligió este lenguaje por ser Orientado a Objetos, robusto, con variadas características, estándares e implementaciones, de libre distribución y completamente portable, esta última característica es gracias que funciona sobre un motor de interpretación llamado "maquina Virtual de JAVA", el cual está disponible para todos los sistemas operativos. Los estándares utilizados para este sistema serán J2EE y J2SE.

- Framework Struts: Framework de JAVA para el desarrollo bajo MVC. Se eligió este framework ya que es considerado uno de los mejores y mas claros al momento de utilizar MVC, además su metodología permite utilizar características, como el manejo de objetos complejos anidados, de forma más simple y segura. Además su implementación en la Vista, basada en JSP, permite independencia de JAvascript.

- Framework Toplink: Esta herramienta perteneciente a la familia de los ORS (Object Relational mapping) es implementada por Oracle, de libre distribución y es la encargada de realizar el mapeo Objeto-Relacional, implementando la capa de persistencia. Característica permitida en java por el JPA (Java Persistente API). Esto permite abstraerse de la base de datos relacional y manejar cada tabla como un objeto (clase).

- Servlets: API de java, los servlets son componentes que se ejecutan en el servidor y son utilizados para incrementar la funcionalidad de la Web, estos servlets son programas Web, pero carentes de interfaz gráfica.

- Xstream: Librería implementada en JAVA que se utilizará para el parseo XML a objeto y Viceversa. Esto se utilizará para la creación y recuperación de plantillas.

- XSL FOP: Librería implementada en JAVA que se utilizará para la transformación del documento XML a formato RTF.

Nota: Para más detalles de la tecnología utilizada en la construcción de este sistema ver anexo Nº 2. Las fuentes de información se encuentran en el respectivo anexo.

2.10.- Estudio de Factibilidad

A continuación se presenta el estudio que determinó si este proyecto era factible de lograr. Se consideraron tres áreas de estudio de factibilidad: Operacional, Técnica y Económica.

Factibilidad Técnica:

Este apartado hace referencias a todas las herramientas, tanto de software como hardware que se necesitan para el desarrollo y posterior utilización del sistema.

Requerimientos para el Desarrollo del Proyecto:

Software:

- Windows XP o Linux
- JAVA SDK 6
- Netbeans IDE 6.5
- MySQL 5,0
- MySQL WorkBench
- Navegador Web (Mozilla Firefox u Opera)
- Procesador de Textos y Hoja de Calculo (MS Office u Open Office).

Todos estos elementos son, o tienen, alguna versión de libre distribución. La responsabilidad de tener estas herramientas cae en la empresa "Agrega Ltda.", por lo que "Home Inspector" no incurrirá en ningún tipo de gastos en herramientas CASE para el desarrollo del sistema. "Agrega Ltda." posee todo el software necesario para el desarrollo del sistema.

Hardware:

- PC de escritorio o Notebook con procesador de 2.0 Ghz o superior, mínimo 512 mb. RAM y 40 Gb Disco Duro
- Impresora Láser o Inyección de Tinta.

La empresa Agrega Ltda. posee todo el equipamiento de Hardware Necesario.

Requerimientos para la utilización del sistema:

Software:

- Sistema Operativo Windows (cualquier Versión) o Linux.
- Navegador Web Mozilla Firefox, Opera o I Explorer.

La empresa "Home Inspector" posee el software necesario. No se considera servidor ya que éste sistema será montado sobre un servidor externo de pago.

Hardware:

- PC de escritorio, procesador 1.5 Ghz o superior.
- Impresora Láser o Inyección de Tinta.
- Dispositivos móviles de comunicación con acceso a Internet vía Web (no wap)

"Home Inspector", posee el equipamiento necesario. Sin embargo es posible la renovación del equipamiento de comunicación móvil del equipo de Inspectores, fijando un monto tope de $3000000, el cuál ya está incluido en el presupuesto por lo que no implica un obstáculo para el desarrollo, lo mismo sucede con el servidor para la aplicación.

Otros requisitos

- Conexión a Internet de Banda Ancha superior a los 600 Kb/s.

La empresa ya posee la conexión.

La Conclusión de Factibilidad Técnica es la siguiente:

A través del estudio de factibilidad técnica ha quedado demostrado que tanto la empresa desarrolladora, como el cliente final poseen todo el equipamiento y herramientas necesarias para el desarrollo y utilización del sistema.

Factibilidad Operativa

Los futuros usuarios del sistema de inspección de viviendas Home Inspector son profesionales del área de la construcción, especialistas en calidad de viviendas y capacitados en el manejo computacional a nivel de usuario.

Este nuevo y pionero sistema tendrá una baja complejidad para su uso, siendo bastante intuitivo y fácil de usar, ya que estará basado principalmente en campos de texto para llenado y checkbox lo que harán de él una herramienta sencilla de seguir.

Por otro lado el sistema no pretende reemplazar un trabajo que es netamente factible de hacer por personas, si no que estará orientado a apoyar y/o automatizar los procesos realizados por ellos de forma completamente manual. Además el sistema no generará cambios en la forma de llevar a cabo el proceso, por lo que se espera que la adaptación a él sea rápidamente aceptada.

La Conclusión Factibilidad Operativa es la siguiente:

Según este análisis operativo se espera que el software sea bien recibido, ya que debido a la utilidad presentada y la facilidad de uso se verá como una herramienta útil y necesaria para el mejor desempeño y comodidad de los usuarios.

Factibilidad Económica

La empresa Agrega Ltda. para el desarrollo de este sistema necesita ciertas herramientas de software y Hardware, a continuación se presenta un cuadro de herramientas y valores:

Inversión en Software:

Nombre	Valor
MS Windows Xp OS	$ 107.000
Linux OS	$ 0
Netbeans IDE	$ 0
Mysql 5.x	$ 0
Mysql Gui Tools	$ 0
Java JDK	$ 0
Java JRE	$ 0
Umbrello	$ 0
MS Office Basic	$ 52.200
Total	$ 159.200

Inversión en Hardware:

Nombre	Valor
PC 2.1 Ghz +	$ 263.000
Impresora Láser	$ 47.400
Total	$ 310.400
Total HW + SW	**$ 469.600**

Sin embargo, los elementos antes mencionados ya están en existencias en la empresa, por lo que no se incurrirán a nuevos gastos en software o hardware.

Inversión en Recursos Humanos:

Se estima un total de 200 H/H utilizadas para la construcción y diseño de este sistema. El valor de hora hombre se calcula al 0,3 UF, lo que equivale a $6000 considerando la UF a $20900

Nombre	Valor
Diseño y Construcción	$ 1.200.000

Por otra parte la empresa "Home Inspector" requiere para el uso los siguientes elementos:

Nombre	Valor
PC 2.0 Ghz +	$ 263.000
MS Windows Xp	$ 107.000
MS Office	$ 52.000
Navegador Web Mozilla Firefox	$ 0
Host Web	$25000 mensual
Pocket PC (10 Unidades)	$ 1.930.000
Desarrollo	$ 1.500.000
Total	$ 3.852.000

Las Conclusiones de este estudio de Factibilidad Económica son las siguientes:

El valor total anterior corresponde a la máxima inversión a ser hecha por la empresa, sin embargo todos los elementos necesarios ya están en existencias de la empresa, por lo que la inversión se resume al valor a cancelar por el desarrollo, más el host mensual, y eventualmente la compra de nuevos dispositivos móviles (poquet PC). Lo anterior nos indica que el valor total está por debajo del presupuesto máximo de 5 millones de pesos.

A partir de este completo análisis de factibilidad se ha concluido lo siguiente:

Con los antecedentes señalados en los puntos anteriores se determina que es completamente factible el desarrollo de este sistema, ya que se poseen la mayoría de los elementos y el valor total a invertir está por debajo de lo presupuestado.

2.11 Algunos Alcances Generales

El problema de generar este tipo de reportes es que su estructura es de un tipo de dato complejo y necesita de más mecanismos lógicos de los que ofrecen las BDR, por lo tanto se ha recurrido a los operadores lógicos del lenguaje JAVA para alcanzar el nivel de funcionalidad que requiere la implementación y que ninguna funcionalidad cambiada involucre un elevado costo de programación.

Considerando lo anterior se utilizarán librerías y herramientas auxiliares para apoyar este proceso de la manera mas clara posible, utilizando XML como puente de mapeo entre objetos de persistencia e instanciación de clases en tiempo de ejecución.

El diseño del Software parte de la base que la lógica del sistema debe plasmarse en múltiples aplicaciones y tecnologías, siendo los objetos (y los mecanismos que ofrece el paradigma OO y el lenguaje, como herencia, polimorfismo, tipos complejos de datos, etc.) el puente de comunicación entre estas aplicaciones.

Esto significa que un problema complejo como este, se basa en la división de tareas asignadas a pequeños proyectos, en donde cada uno se hará cargo de una parte y toda la comunicación se hará a través de objetos complejos, por ejemplo un proyecto reunirá las clases de persistencia, las cuales se encargarán de hacer persistir los objetos en el tiempo, comunicando la capa controladora y de lógica de negocios con la base de dato real (mapeo Objeto-Relacional).

Otro de los factores críticos presentes en este proyecto es el de la carga del cliente, toda aplicación Web lleva implícito un retardo natural debido al traspaso de datos que existe entre el cliente y el servidor. Es por esto que en este proyecto queda descartada la posibilidad de utilizar Javascript en grandes cantidades de código, ya que el costo de renderización en navegadores livianos es demasiado alto.

Por otro lado está el costo de programación del sistema, principalmente en los que se refiere a la modificación. Este sistema contempla la posibilidad de extensibilidad, basados en esto se usarán los métodos de mapeo Objeto Relacional anteriormente mencionados, ya que abstrae el traspaso de datos de paradigma relacional a objetos.

Una de las carencias de las bases de datos libres de pago es el pobre manejo de archivos XML, afortunadamente en este proyecto no es requerimiento el llevar módulos estadísticos específicos de fallas estructurales, sin embargo el costo de cambiar el actual motor de base de datos por uno de pago que sí maneje XML a través de XQuery es mínimo, otra de las ventajas que da el mapeo de objetos relacional.

Por otro lado, la necesidad de cambiar cualquier cosa ya sea en el área de negocios, de la vista o de la lógica no significará un gran esfuerzo de programación. Esta es una de las ventajas que nos permite el modelo MVC, ya que cada capa está separada una de otra y la comunicación se realiza a través de los mensajes entre objetos, por ejemplo, si se quiere cambiar la posición de los elementos en la vista el programador no necesita tocar el código que dice el "como", cosa que pasa con otros lenguajes de libre acceso, como PHP, en donde la vista, la lógica de negocios y los controladores están en una misma sección provocando un escenario bastante complejo.

A continuación se presenta un diagrama explicativo de la operabilidad del sistema en la figura 2.1:

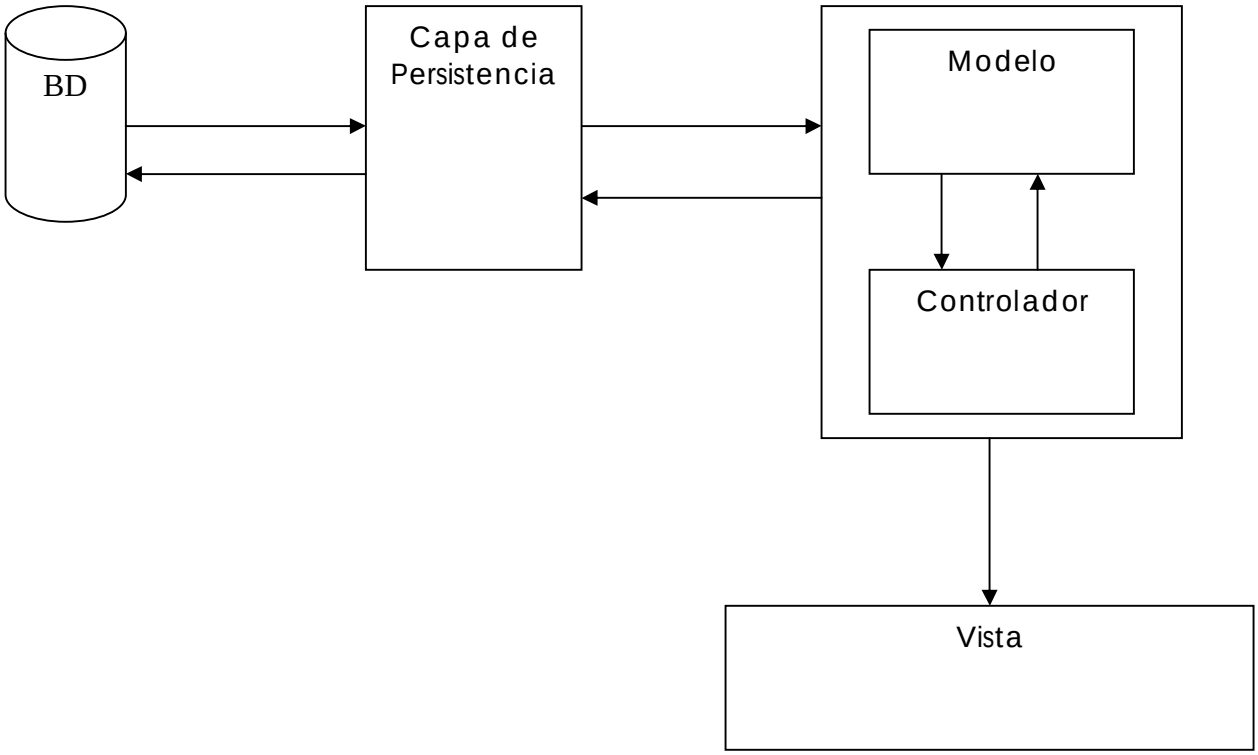


Figura 2.1: Operación global del sistema

En la figura 2.2 se muestra Diagrama de la operación del sistema para las funciones de plantillas y reportes:

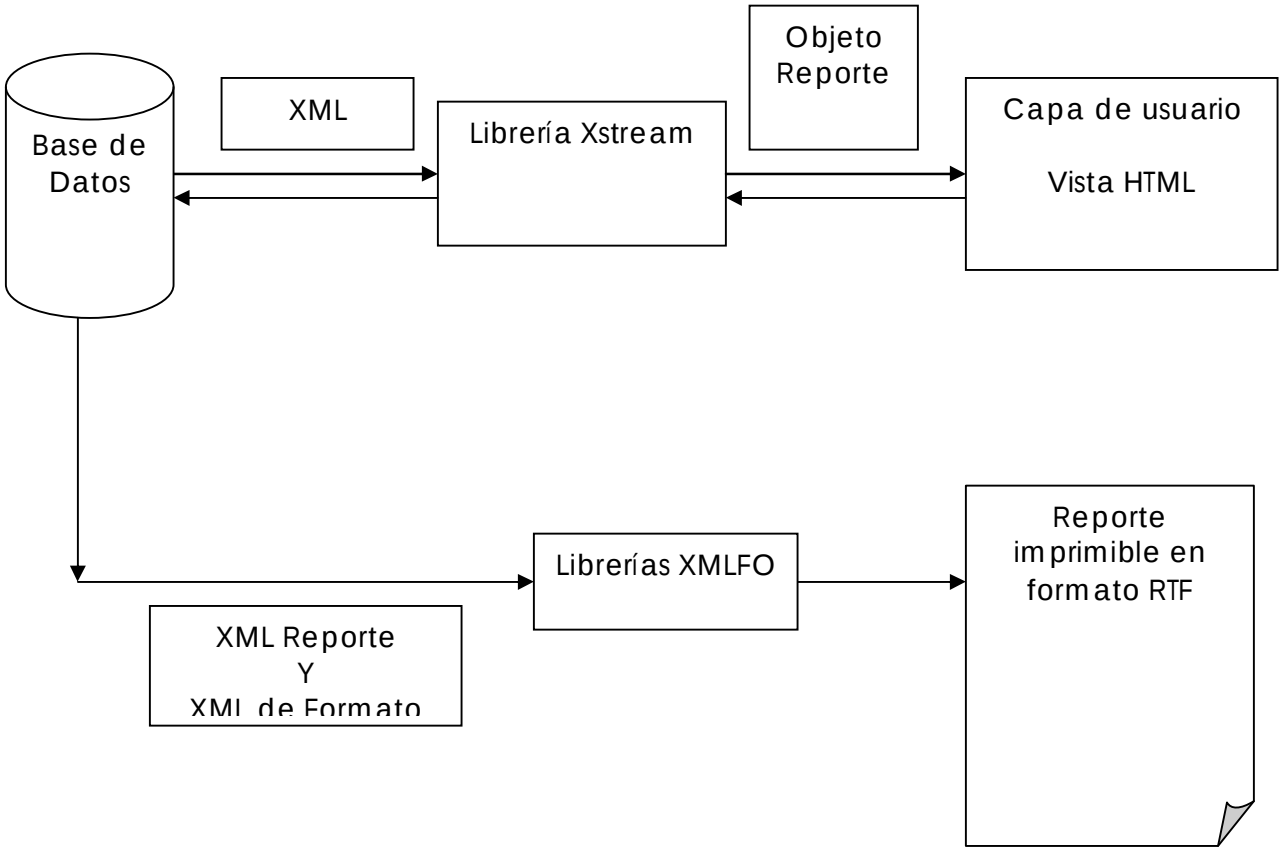


Figura 2.2: Operación del sistema en plantillas e informes.

Capitulo 3

Especificación de Requerimientos

Introducción al Capítulo

En este capitulo se establecen todos los requerimientos del sistema, tanto respecto a su funcionalidad, como a la tecnología necesaria para llevarse a cabo, se describe cada uno según su clasificación.

3.1.- Requerimientos Funcionales

Del Acceso al sistema:

1.- El sistema deberá poseer un módulo de ingreso al sistema.
- 1.1.- El sistema deberá restringir el acceso si el usuario no se ha identificado.
- 1.2.- El sistema deberá solicitar un nombre de usuario y una contraseña para poder acceder a él.
- 1.3.- El nombre de usuario deberá ser alfanumérico de máximo 10 caracteres.
- 1.4.- La contraseña deberá ser alfanumérica de máximo 10 caracteres.

De la mantención de usuarios

2.- El sistema deberá permitir la gestión de usuarios
- 2.2.- Los usuarios deberán ser de tres tipos:
  - Administrador: usuario global, tendrá acceso a todas las opciones del sistema.
  - Cliente: deberá tener acceso limitado, sólo podrá revisar los trabajos asociados e él.
  - Inspector: deberá tener acceso limitado, orientado al manejo de plantillas para inspección.
- 2.1.- El sistema deberá permitir el ingreso de nuevos usuarios.
- 2.2.- Para cada usuario (Administrador, Cliente, Inspector) el sistema deberá guardar la siguiente información.
  - Nombre
  - Dirección
  - Correo Electrónico
  - Fono
  - Fax
  - Nombre de usuario del sistema
- 2.3.- El sistema deberá proveer de un módulo de mantención de los usuarios que permita modificar sus datos y eliminar al usuario.

De las Propiedades

3.- El sistema deberá almacenar las propiedades asociadas a cada cliente.
- 3.1.- Para cada propiedad el sistema deberá almacenar:
  - Dirección
  - Rol

- Avalúo Fiscal
- Tipo

3.2.- El sistema deberá permitir asociar a las propiedades una respectiva orden de trabajo.

3.3.- El sistema deberá proveer de un módulo para la mantención de las propiedades, el cuál debe permitir eliminar y modificar dicha propiedad.

3.4.- El sistema deberá permitir listar las propiedades.

De las Órdenes de Trabajo:

4.- El sistema deberá permitir la emisión y/o almacenamiento de una orden de trabajo.

4.1.- Cada orden de trabajo deberá estar asociado a una propiedad.

4.2.- La numeración de la orden de trabajo deberá ser otorgada en forma automática y correlativa por el sistema.

4.3.- Cada Orden de trabajo deberá estar asociada a una plantilla.

4.3.- Cada orden de trabajo se deberá crear almacenando los siguientes datos:

- Honorarios (bruto, iva, total)
- Identificador de Plantilla asociada.
- Inspector encargado
- Propiedad asociada.

4.4.- El sistema deberá permitir listar las órdenes de trabajo almacenadas.

De las Plantillas:

5.- El sistema deberá permitir la creación de Plantillas en forma dinámica, pudiéndose crear plantillas de acuerdo a la necesidad de cada vivienda.

5.1.- Cada plantilla deberá ser creada con la siguiente Información

- Identificador (automático)
- Nombre
- Descripción

5.2.- El sistema deberá listar las plantillas existentes.

6.- El sistema deberá permitir la modificación de los datos de la plantilla.

7.- El sistema deberá permitir Eliminar una plantilla.

8.- El sistema deberá tener un módulo que permita la estructuración de las plantillas, el cual deberá contener los siguientes módulos:

- Capitulo
- Sección
- Ítem
- Sub-Ítem
- Valor (no monetario, valor de estado)

De los módulos de Plantillas:

9.- El sistema deberá almacenar los ítems asociados a cada módulo mencionado en el punto anterior.

9.1.- cada ítem podrá ser seleccionado a su respectivo módulo en la creación de la estructura de la plantilla.

10.- El sistema deberá permitir la mantención de los módulos de la estructura de plantillas.

10.1.- El sistema deberá permitir el ingreso de nuevos ítems a los módulos.

10.2.- El sistema deberá permitir la eliminación de un ítem específico de cada módulo.

De los reportes:

11.- El sistema deberá ser capaz de emitir el informe de inspección final.

11.1.- El informe debe estar asociado a una orden de trabajo.

11.2.- El sistema deberá formatear este informe al modelo estándar a partir de la respectiva plantilla seleccionada.

11.3.- El informe deberá ser emitido en formato PDF, DOC o RTF.

3.2.-Requerimientos No Funcionales.

1.- El sistema deberá ser intuitivo y fácil de utilizar.
2.- El sistema deberá ser estable y confiable.
3.- El sistema deberá poder ser accedido en forma remota desde cualquier navegador Web.
4.- El sistema deberá estar diseñado de tal forma que el llenado de plantillas se pueda realizar desde dispositivos móviles vía Web.

3.3.- Requerimientos de Entrada/salida

En la tabla nº2 expuesta a continuación se detallan las entradas y salidas principales del sistema.

Item	Detalle	Medio de Entrada	Rango Válido	Unidad de Medida	Formato de Datos
Usuarios	Id	No Aplicable	No Aplicable	No Aplicable	Entero(10)
	Rut	Teclado	RUT válido	No Aplicable	Varchar (10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar (100)
	Dirección	Teclado	Alfanumérico	No Aplicable	Varchar (100)
	Comuna	Teclado	Alfabético	No Aplicable	Varchar(45)
	Provincia	Teclado	Alfabético	No Aplicable	Varchar(45)
	Región	Combo-Box	No Aplicable	No Aplicable	Varchar(45)
	E-mail	Teclado	Formato Mail	No Aplicable	Varchar (45)
	Fono	Teclado	Números	No Aplicable	Varchar (20)
	Fax	Teclado	Números	No Aplicable	Varchar (20)
	Nombre para acceso	Teclado	Alfanumérico	No Aplicable	Varchar (20)
	Clave de acceso	Teclado	Alfanumérico	No Aplicable	Varchar(10)
	Rol	Combo-Box	No Aplicable	No Aplicable	Varchar(20)
Propiedad	Id	No Aplicable	No Aplicable	No Aplicable	Entero(10)
	Rol Propiedad	Teclado	Alfanumérico	No Aplicable	Varchar(20)
	Dirección	Teclado	Alfabético	No Aplicable	Varchar(100)
	Comuna	Teclado	Alfabético	No Aplicable	Varchar(45)
	Provincia	Teclado	Alfabético	No Aplicable	Varchar(45)
	Región	Combo-Box	No Aplicable	No Aplicable	Varchar(45)
	Tipo	Combo-Box	No Aplicable	No Aplicable	Varchar(20)
	Avalúo	Teclado	Número > 0	Pesos	Long Int
Plantilla	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	No Aplicable	No Aplicable	Varchar(45)
	Descripción	Teclado	No Aplicable	No Aplicable	Long Text
	Fecha Creación	No Aplicable	No Aplicable	No Aplicable	Date
Capitulo	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar
	Descripción	Teclado	Alfabético	No Aplicable	Long Text

Sección	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar
	Descripción	Teclado	Alfabético	No Aplicable	Long Text
Valor	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar
	Descripción	Teclado	Alfabético	No Aplicable	Long Text
Ítem	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar
	Descripción	Teclado	Alfabético	No Aplicable	Long Text
Sub-Ítem	Id	No Aplicable	No Aplicable	No Aplicable	Int(10)
	Nombre	Teclado	Alfabético	No Aplicable	Varchar
	Descripción	Teclado	Alfabético	No Aplicable	Long Text
Reporte	Id	No Aplicable	No Aplicable	No Aplicable	int(10)
	Fecha Creación	No Aplicable	No Aplicable	dd/mm/aaaa	date
	factor	Teclado	Numero >0	Porcentaje	float
	Honorarios	Teclado	Numero >0	Peso	int(10)
	XML	No Aplicable	Texto	No Aplicable	Long Text
	Fecha Inicio	No Aplicable	Fecha	dd/mm/aaaa	date
	Fecha Fin	No Aplicable	Fecha	dd/mm/aaaa	date
	IVA	Teclado	Numero >0	Peso	int(10)
	Total	No Aplicable	Numero >0	Peso	int(10)
	Plantilla	Check	No Aplicable	No Aplicable	int
	Administrador	No Aplicable	No Aplicable	No Aplicable	int
	Inspector	Check	No Aplicable	No Aplicable	int
	propiedad	Check	No Aplicable	No Aplicable	int

Tabla Nº2: Entradas y salidas del sistema.

3.4.- Requerimientos de Seguridad

- Permitir el acceso al sistema sólo a usuarios autorizados.
- Cuentas de Usuarios con nivel de acceso dependiendo de su rol de usuario (Administrador, Cliente, Inspector).

3.5.- Requerimientos de Diseño e Implementación

- El sistema debe estar construido para soportar una interfaz multi usuario, quienes trabajarán bajo los mismos datos (Base de Datos Centralizada)
- El sistema debe estar diseñado de tal forma que permita ser accedido en tiempo mínimo a través de dispositivos de navegación Web básica, como Palm o Iphone.

3.6.- Requerimientos para la construcción

Para hacer posible la construcción del sistema se requieren los siguientes elementos:

- Sistema Motor de Base de Datos (MySql)
- Lenguaje de Programación Java
- Software Utilitario: Windows Xp o Linux 2.xx, NetBeans IDE 6 o superior, Open Office o MS Office, Umbrello o similar, MySql Gui Tools, JAVA SDK, JAVA JRE, Navegador Web.
- PC de escritorio o Notebook, Athlon 2.1 o superior.

Capitulo 4

Diseño Global del Software

Introducción al Capitulo

En este capítulo se espera identificar todos lo usuarios que tendrá el sistema, que mas adelante se denominan actores, además de establecer la totalidad de las funciones del sistema y la relación que tiene con cada tipo de usuario. Lo anterior se muestra en los Diagramas de Casos de Uso, los cuales muestran al actor y sus funciones, además está el Caso de Uso en donde se muestra el proceso de comunicación entre el sistema y el actor y como se lleva a cabo cada función del sistema.

4.1 Definición de Actores y Usuarios

En esta sección se identifican y caracterizan los diferentes usuarios que interactúan con el sistema

- Administrador:

Este usuario tiene acceso global al sistema, entre sus funciones está la de mantener los módulos de usuarios y propiedades, así como también tiene acceso al gestor de plantillas. Sólo este usuario está autorizado a acceder a todas las funciones del sistema.

- Inspector

Este usuario tiene acceso a las plantillas creadas por el usuario administrador encargado del diseño de plantillas, puede acceder también a las ordenes de trabajo, las que puede utilizar para cargar el formulario de inspección adecuado. Su principal función es el llenado de los formularios, ya que es él quién hará el trabajo en terreno. También puede generar el informe de inspección.

- Cliente

Este es el único usuario externo a la empresa que tiene acceso al sistema, y su participación solo tiene relevancia a la visualización de un informe asociado a un propiedad asociada a él como cliente. No puede visualizar ningún otro módulo del sistema.

4.2 Definición y Especificación de Casos de Uso

Caso de uso Registrar Usuario

Nombre	Registrar usuario
Req. Asociados	1 - 2
Descripción: En este caso de uso el usuario podrá registrar en la base de datos un nuevo usuario, el cual tendrá acceso al sistema una vez registrado.	
Actores: Administrador	
Precondiciones: 1.- El usuario Administrador debe haber ingresado al sistema previamente. 2.- El nuevo registro no debe existir en la base de datos.	
Flujo Normal: 1.- El actor pulsa en el menú la pestaña "usuarios" 2.- El sistema despliega el formulario para la inserción del nuevo usuario. 3.- El usuario llena los campos desplegados que corresponden a rut, nombre, dirección, fono, mail, fax, nombre de acceso, clave y rol (Administrador, Inspector, Cliente). 4.- Una vez completados los datos el actor presiona el botón registrar usuario. 5.- El sistema comprueba la validez de los datos y los almacena.	
Flujo Alternativo: 5.- El sistema comprueba la validez de los datos, si éstos no son correctos, el sistema emite un mensaje de error y da la posibilidad al usuario de corregirlos.	
Post Condiciones 1.- El nuevo registro debe haber sido ingresado en la base de datos.	

Caso de Uso Modificar Usuario

Nombre:	Modificar Usuario
Req. Asociados:	1 - 2
Descripción: En este caso de uso el Administrador podrá modificar los datos de un usuario existente en la base de datos.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- El registro a modificar debe existir en el sistema.	
Flujo Normal: 1.- El actor pulsa en el menú la pestaña "usuarios". 2.- El sistema despliega el formulario de modificación/inserción y la lista con los usuarios existentes. 3.- El actor busca y selecciona el usuario a modificar en la lista de registros. 4.- El sistema muestra en el formulario los datos del usuario seleccionado. 5.- El actor modifica los datos correspondientes y presiona el botón de modificación. 6.- El sistema comprueba que los datos estén correctos y graba en la base de datos.	
Flujo Alternativo: 6.- El sistema comprueba la validez de los datos, si éstos no son correctos, el sistema emite un mensaje de error y da la posibilidad al usuario de corregirlos.	
Post Condiciones: 1.- El sistema debe haber registrado las modificaciones en la BD.	

Caso de Uso Eliminar Usuario

Nombre:	Eliminar Usuario
Req. Asociados:	1-2
Descripción: En este módulo el actor puede desactivar a un usuario para impedir su acceso al sistema.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- El registro a modificar debe existir en el sistema.	
Flujo Normal: 1.- El actor pulsa en el menú la pestaña "usuarios". 2.- El sistema despliega el formulario de modificación/inserción y la lista con los usuarios existentes. 3.- El actor selecciona el usuario a eliminar de la lista y presiona el botón Eliminar. 4.- El sistema solicita confirmación del borrado del usuario. 5.- El actor confirma la eliminación. 6.- El sistema desactiva el registro en la base de datos (modifica campo estado).	
Flujo Alternativo: 5.- El actor cancela la eliminación. 6.- El sistema vuelve al formulario.	

Post Condiciones: 1.- El sistema debe sacar de la lista de usuario activos al usuario eliminado.

Caso de uso Listar usuario

Nombre:	Listar Usuario
Req. Asociados:	1-2
Descripción: El actor podrá ver en pantalla una lista con los usuarios activos.	
Actores: Administrador	
Precondiciones: 1.- El usuario Administrador debe haber ingresado al sistema previamente**.**	
Flujo Normal: 1.- El actor pulsa en el menú la pestaña "usuarios". 2.- El sistema despliega el formulario de modificación/inserción y la lista con los usuarios existentes.	
Flujo Alternativo: No aplica	
Post Condiciones: No Aplica	

Caso de uso Modificar Cliente

Nombre:	Modificar Cliente
Req. Asociados:	1-2
Descripción: En este caso de uso el Administrador podrá modificar los datos de un cliente existente en la base de datos.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- El registro del cliente a modificar debe existir en el sistema.	
Flujo Normal: 1.- El actor pulsa en el menú la pestaña "Clientes". 2.- El sistema despliega el formulario de modificación de clientes junto a una lista de los clientes activos, además despliega las propiedades asociadas a dicho cliente. 3.- El actor busca y selecciona el cliente a modificar en la lista de registros. 4.- El sistema muestra en el formulario los datos del cliente seleccionado. 5.- El actor modifica los datos correspondientes y presiona el botón de modificación.	
Flujo Alternativo: 5.- El sistema comprueba la validez de los datos, si éstos no son correctos, el sistema emite un mensaje de error y da la posibilidad al actor de corregirlos.	
Post Condiciones: 1.- El sistema debe haber registrado las modificaciones en la BD.	

Caso de uso Ingresar Propiedad

Nombre:	Ingresar propiedad
Req. Asociados:	1-4
Descripción: En este caso de uso el actor puede ingresar una nueva propiedad y asociarla a un cliente previamente ingresado en el sistema.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haberse identificado previamente en el sistema. 2.- Debe existir un cliente al cual asociar la propiedad a ingresar. 3.- La propiedad no debe existir en el sistema.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Clientes" en el menú de usuarios. 2.- El sistema debe desplegar el formulario de Clientes/Propiedades. 3.- El actor debe seleccionar desde la lista de clientes aquel al cual le vaya a asignar la propiedad. 4.- El sistema deberá desplegar los datos del cliente y las propiedades existentes en caso de existir. 5.- El actor deberá rellenar con los datos el formulario desplegado correspondiente al ingreso de la nueva propiedad, estos datos son dirección, rol, avalúo fiscal y tipo, una vez ingresado los datos el actor debe presionar el botón "Guardar". 6.- El sistema comprobará la validez de los datos y los guardará en la base de datos.	
Flujo Alternativo: 6.- El sistema comprobará la validez de los datos, de encontrar errores emitirá un mensaje y dará la opción al actor de modificar los datos.	
Post Condiciones: 1.- La nueva propiedad debe quedar ingresada en el sistema y asociada a un cliente. 2.- Cuando se listen los datos del cliente el sistema deberá listar además la propiedad ingresada.	

Caso de uso Modificar propiedad

Nombre:	Modificar propiedad
Req. Asociados:	1-4
Descripción: En este caso de uso el actor podrá modificar una propiedad asociada a un cliente.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema. 2.- La propiedad debe existir. 3.- El cliente debe estar activo.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Clientes" en el menú de usuarios. 2.- El sistema debe desplegar el formulario de Clientes/Propiedades. 3.- El actor debe seleccionar desde la lista de clientes aquel al cual corresponde la propiedad a modificar. 4.- El sistema deberá desplegar los datos del cliente y las propiedades existentes. 5.- El actor deberá seleccionar la propiedad que desea modificar desde la lista de propiedades asociadas al cliente. 6.- El actor deberá modificar la información correspondiente la propiedad, seleccionada y una vez hecho presionar el botón "Guardar" 6.- El sistema comprobará la validez de los datos y los guardará en la base de datos con las respectivas modificaciones.	
Flujo Alternativo: 4.- no existen propiedades asociadas al cliente, el actor deberá ingresar una nueva propiedad de ser necesario. 6.- El sistema comprueba la validez de los datos, si estos no son válidos el sistema emite un mensaje de error y permite al actor ingresar los datos nuevamente.	
Post Condiciones: 1.- La propiedad debe quedar registrada en la base de datos con la correspondiente modificación.	

Caso de uso Eliminar Propiedad

Nombre:	Eliminar propiedad
Req. Asociados:	1-4
Descripción: En este caso de uso el actor podrá eliminar una propiedad asociada a un cliente.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema. 2.- La propiedad debe existir. 3.- El cliente debe estar activo.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Clientes" en el menú de usuarios. 2.- El sistema debe desplegar el formulario de Clientes/Propiedades. 3.- El actor debe seleccionar desde la lista de clientes aquel al cual corresponde la propiedad a modificar. 4.- El sistema deberá desplegar los datos del cliente y las propiedades existentes. 5.- El actor deberá seleccionar la propiedad que desea eliminar desde la lista de propiedades asociadas al cliente. 6.- El actor deberá presionar el botón "Eliminar" ubicado en la propiedad correspondiente en la lista de propiedades. 7.- El sistema solicita al actor confirmación para eliminar la propiedad. 8.- El actor presiona el botón de confirmación. 9.- El sistema cambiará el estado de la propiedad a inactivo.	
Flujo Alternativo: 6.- El actor no desea eliminar la propiedad, presiona el botón cancelar.	
Post Condiciones: 1.- El sistema debe cambiar el estado de la propiedad a inactivo.	

Caso de uso Listar Propiedad

Nombre:	Listar Propiedad
Req. Asociados:	1-3
Descripción: El actor puede visualizar las propiedades asociadas a un cliente en específico.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema. 2.- La propiedad debe existir. 3.- El cliente debe estar activo.	
Flujo Normal: 1.- El cliente desea acceder a ingreso, modificación o eliminación de propiedades. 2.- El cliente presiona la pestaña "clientes" desde el menú. 3.- El sistema lista las propiedades.	
Flujo Alternativo: No Aplica	
Post Condiciones: No Aplica.	

Caso de uso Crear Orden de Trabajo

Nombre:	Crear Orden de Trabajo (OT)
Req. Asociados:	1-4-5-9
Descripción: El actor podrá generar una orden de trabajo para una propiedad ya existente.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- Debe existir la propiedad. 3.- Debe existir la plantilla a utilizar para la inspección.	
Flujo Normal: 1.- El actor debe seleccionar la pestaña "clientes" desde el menú de navegación. 2.- El sistema debe desplegar el formulario de Clientes/Propiedades. 3.- El actor debe seleccionar el cliente al cual le desea asignar una orden de trabajo. 4.- El sistema debe desplegar los datos del cliente y las propiedades asociadas a éste. 5.- El actor debe seleccionar la propiedad a la cual desea asignar una OT. 6.- El actor debe presionar el botón "Generar OT" que se encuentra junto a la propiedad deseada. 7.- El sistema debe desplegar el módulo para generar OT. 8.- El actor debe completar los campos dispuestos para la creación de la OT, los cuales son Factor Honorarios, Total Honorarios, IVA, Total, plantilla para el informe, inspector y la propiedad. 9.- Una vez ingresados los datos, el actor debe presionar el botón "Generar OT" 10.- El sistema debe ingresar la OT a la base de datos y llevar al actor a la pantalla "Lista de OT" y permitir al actor editar la OT y completar el informe de inspección.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.-La OT debe quedar ingresada en el sistema. 2.- Un inspector deberá realizar el informe asociado a esa OT.	

Caso de uso Modificar Orden de Trabajo

Nombre:	Modificar OT
Req. Asociados:	1-4
Descripción: En este caso de uso el actor podrá modificar los datos de una OT	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- Debe existir la orden de trabajo.	
Flujo Normal: 1.- El actor debe acceder al módulo de órdenes de trabajo a través del menú de navegación. 2.- El sistema debe desplegar en pantalla una lista con las órdenes de trabajo existentes. 3.- El actor debe seleccionar la OT que desea modificar y presionar el botón "Modificar" 4.- El sistema debe desplegar el formulario de Ingreso/Modificación de OT 5.- El actor debe modificar los campos deseados y presionar el botón "Guardar". 6.-El sistema guardará la información modificada y llevará al actor de vuelta a la pantalla de "Orden de Trabajo"	
Flujo Alternativo: 3.- El actor no desea modificar OT alguna, presiona el botón "Cancelar" y vuelve atrás.	
Post Condiciones: 1.- El sistema debe reflejar los cambios realizados en la OT.	

Caso de uso Listar Orden de Trabajo

Nombre:	Listar OT
Req. Asociados:	1-4
Descripción: En este caso de uso el actor se dirige a un módulo que lista todas las órdenes de trabajo existentes.	
Actores: Administrador, Inspector	
Precondiciones: 1.- El actor debe haber ingresado al sistema. 2.- Debe existir la orden de trabajo.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Orden de Trabajo" desde el menú de navegación. 2.- El sistema deberá desplegar en pantalla la lista de Órdenes de Trabajo.	
Flujo Alternativo: No Aplica	
Post Condiciones: No Aplica	

Caso de uso Editar Reporte

Nombre:	Editar Reporte
Req. Asociados:	1, 4-11
Descripción: En este caso de uso el actor podrá editar un reporte asociado a una orden de trabajo existente.	
Actores: Inspector, Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema. 2.- Debe existir la OT.	
Flujo Normal: 1.- El actor debe listar las ordenes de trabajo accediendo la pestaña "orden de Trabajo" del menú de navegación. 2.- El sistema deberá desplegar en pantalla la lista de Órdenes de Trabajo. 3.- El actor debe seleccionar la orden de trabajo y presionar el botón "Editar Reporte" que se encuentra junto a la orden de trabajo. 4.- El sistema debe desplegar en pantalla el módulo que contiene el formulario con la estructura de plantilla. 5.- El actor debe comenzar a completar el formulario desplegado en pantalla, el cual está basado en la plantilla asociada a la orden de trabajo. 6.- Una vez completo el formulario el actor debe presionar el botón "Guardar reporte". 7.- El sistema guardará el reporte en la base de datos.	
Flujo Alternativo: 3.- El actor debe seleccionar la orden de trabajo y presionar el botón "Editar Reporte" que se encuentra junto a la orden de trabajo, si la orden de trabajo no existe, el actor debe comunicarse con el administrador. 5.- No se desea modificar el reporte, el actor presiona el botón "cancelar"	
Post Condiciones: 1.- El sistema deberá desplegar el reporte en su estado final.	

Caso de uso Imprimir reporte

Nombre:	Generar/Imprimir reporte
Req. Asociados:	1, 11
Descripción: En este caso de uso, el actor solicitará al sistema formatear e imprimir en reporte de inspección.	
Actores: Cliente, Administrador, Inspector	
Precondiciones: 1.- El actor debe haber accedido al sistema. 2.- El reporte debe haber sido editado previamente.	
Flujo Normal: 1.- El actor debe listar las ordenes de trabajo accediendo la pestaña "orden de Trabajo" del menú de navegación. 2.- El sistema deberá desplegar en pantalla la lista de Órdenes de Trabajo. 3.- El actor debe seleccionar la orden de trabajo y presionar el botón "Imprimir Reporte" que se encuentra junto a la orden de trabajo. 4.- El sistema imprimirá un documento en formato RTF, el cual corresponde a informe de inspección.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- El informe debe quedar en la base de datos para un futura nueva impresión.	

Caso de uso Visualizar Reporte

Nombre:	Visualizar Reporte
Req. Asociados:	1, 11
Descripción: El actor podrá visualizar en línea el informe de inspección, en formato html.	
Actores: Inspector, Cliente, Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Orden de Compra" desde el menú de navegación. 2.- El sistema deberá desplegar la lista de órdenes de compra. 3.- El actor deberá seleccionar la orden de compra a la cual está asociada el informe que desea visualizar. 4.- El sistema deberá desplegar el formulario de informe basado en HTML.	
Flujo Alternativo: No Aplica	
Post Condiciones: No Aplica	

Caso de uso Crear plantilla

Nombre:	Crear Plantilla
Req. Asociados:	5-10
Descripción: En este caso de uso el actor podrá crear un identificador de plantilla, al cual posteriormente podrá darle una estructura.	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- La plantilla no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña "Plantillas" desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú "Plantillas" 4.- El sistema despliega el formulario de creación de plantillas y la lista con las plantillas existentes. 5.- El actor debe llenar los campos correspondientes para la creación de la plantilla, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón "guardar" 6.- El sistema guardará la plantilla en la base de datos y refrescará la lista de plantillas agregando la recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- La nueva Plantilla debe estar disponible en el sistema para su estructuración.	

Caso de Uso Modificar Plantilla

Nombre:	Modificar Plantilla
Req. Asociados:	6
Descripción: En este caso de uso el usuario podrá editar la información de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- La plantilla debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Plantillas” 4.- El sistema despliega el formulario de creación/edición de plantillas y la lista con las plantillas existentes. 5.- El actor debe modificar los campos correspondientes para la creación/edición de la plantilla, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará las modificaciones de la plantilla en la base de datos y refrescará la lista de plantillas.	
Flujo Alternativo: No Aplica	
Post Condiciones: La plantilla debe ser modificada en la base de datos.	

Caso de uso Eliminar Plantilla

Nombre:	Eliminar Plantilla
Req. Asociados:	7
Descripción: En este caso de uso el actor podrá eliminar una plantilla existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- La plantilla debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Plantillas” 4.- El sistema despliega el formulario de creación de plantillas y la lista con las plantillas existentes. 5.- El actor busca la plantilla requerida en la lista de plantillas y presiona el botón “Eliminar” 6.- El sistema cambia el estado de la plantilla a desactivada y la elimina de las listas de plantillas activas	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de plantillas, aquella que se eliminó no debe aparecer	

Caso de Uso Crear Capitulo

Nombre:	Crear Capitulo
Req. Asociados:	9-10
Descripción: En este caso de uso el actor podrá crear un ítem Capitulo para ser utilizado en la estructura de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- El Capitulo no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Capitulo” 4.- El sistema despliega el formulario de creación de capitulo y la lista con los capítulos existentes. 5.- El actor debe llenar los campos correspondientes para la creación del capitulo, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará el capitulo en la base de datos y refrescará la lista de capítulos agregando el recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- El nuevo capítulo debe estar disponible en el sistema para usarlo en la estructuración de plantillas.	

Caso de uso Modificar Capitulo

Nombre:	Modificar Capitulo
Req. Asociados:	9-10
Descripción: En este caso de uso el usuario podrá editar la información de un Capitulo.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- El Capitulo debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Capítulos” 4.- El sistema despliega el formulario de creación/edición de Capítulos y la lista con los Capítulos existentes. 5.- El actor debe buscar el Capitulo que desea modificar y seleccionarlo. 6.- El sistema debe desplegar los datos del Capitulo en el formulario Creación/Edición de Capitulo. 7.- El actor modifica los campos correspondientes para la creación/edición del Capitulo, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará las modificaciones del Capitulo en la base de datos y refrescará la lista de Capítulos.	
Flujo Alternativo: No Aplica	
Post Condiciones: El Capitulo debe ser modificada en la base de datos.	

Caso de Uso Eliminar Capitulo

Nombre:	Eliminar Capitulo
Req. Asociados:	9-10
Descripción: En este caso de uso el actor podrá eliminar Capítulo existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- El Capítulo debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Capítulos” 4.- El sistema despliega el formulario de creación de Capítulos y la lista con los Capítulos existentes. 5.- El actor busca el Capítulo requerida en la lista de Capítulo y presiona el botón “Eliminar” 6.- El sistema cambia el estado del Capítulo a desactivado y lo elimina de las listas de Capítulo activos.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de Capítulos, aquel que se eliminó no debe aparecer listado.	

Caso de uso Crear Sección

Nombre:	Crear Sección
Req. Asociados:	9-10
Descripción: En este caso de uso el actor podrá crear un ítem Sección para ser utilizado en la estructura de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- La Sección no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Secciones” 4.- El sistema despliega el formulario de creación de Sección y la lista con las Secciones existentes. 5.- El actor debe llenar los campos correspondientes para la creación de la Sección, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará la Sección en la base de datos y refrescará la lista de Secciones agregando la recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- La nueva Sección debe estar disponible en el sistema para usarlo en la estructuración de plantillas.	

Caso de uso Modificar Sección

Nombre:	Modificar Sección
Req. Asociados:	9- 10
Descripción: En este caso de uso el usuario podrá editar la información de una Sección.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- La Sección debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Secciones” 4.- El sistema despliega el formulario de creación/edición de Secciones y la lista con las Secciones existentes. 5.- El actor buscará la sección a modificar desde la lista de secciones y debe seleccionarlo. 6.- El sistema despliega los datos en el formulario de creación/edición de secciones. 7.- El actor debe modificar los campos correspondientes para la creación/edición de la Sección, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 8.- El sistema guardará las modificaciones de la Sección en la base de datos y refrescará la lista de Secciones.	
Flujo Alternativo: No Aplica	
Post Condiciones: La Sección debe ser modificada en la base de datos.	

Caso de uso Eliminar Sección

Nombre:	Eliminar Sección
Req. Asociados:	9- 10
Descripción: En este caso de uso el actor podrá eliminar una Sección existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- La Sección debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Secciones” 4.- El sistema despliega el formulario de creación de Secciones y la lista con las Secciones existentes. 5.- El actor busca la Sección requerida en la lista de Secciones y presiona el botón “Eliminar” 6.- El sistema cambia el estado de la Sección a desactivado y lo elimina de las listas de Secciones activas.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de Secciones, aquel que se eliminó no debe aparecer listado.	

Caso de uso Crear Ítem

Nombre:	Crear Ítem
Req. Asociados:	9- 10
Descripción: En este caso de uso el actor podrá crear un Ítem para ser utilizado en la estructura de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- El Ítem no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Ítem” 4.- El sistema despliega el formulario de creación de Ítem y la lista con los Ítem existentes. 5.- El actor debe llenar los campos correspondientes para la creación del Ítem, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará el Ítem en la base de datos y refrescará la lista de Ítems agregando el recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- El nuevo Ítem debe estar disponible en el sistema para usarlo en la estructuración de plantillas.	

Caso de uso Modificar Ítem

Nombre:	Modificar Ítem
Req. Asociados:	9-10
Descripción: En este caso de uso el usuario podrá editar la información de un Ítem.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- El Ítem debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Ítem” 4.- El sistema despliega el formulario de creación/edición de Ítem y la lista con los Ítem existentes. 5.- El actor debe seleccionar el ítem a modificar desde la lista de ítem. 6.- El sistema despliega en el formulario de creación/edición de ítem los datos de éste. 7.- El actor debe modificar los campos correspondientes para la creación/edición del Ítem, los cuales son nombre y descripción, una vez completos el actor debe presionar el botón “guardar” 8.- El sistema guardará las modificaciones del Ítem en la base de datos y refrescará la lista de Ítem.	
Flujo Alternativo: No Aplica	
Post Condiciones: El Ítem debe ser modificado en la base de datos.	

Caso de uso Eliminar Ítem

Nombre:	Eliminar Ítem
Req. Asociados:	9- 10
Descripción: En este caso de uso el actor podrá eliminar Ítem existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- El ítem debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Plantillas" desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú "Ítem" 4.- El sistema despliega el formulario de creación de Ítem y la lista con los Ítems existentes. 5.- El actor busca el Ítem requerida en la lista de Ítem y presiona el botón "Eliminar" 6.- El sistema cambia el estado del Ítem a desactivado y lo elimina de las listas de Ítem activos.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de Ítems, aquel que se eliminó no debe aparecer listado.	

Caso de uso Crear Sub-Ítem

Nombre:	Crear Sub-Ítem
Req. Asociados:	9- 10
Descripción: En este caso de uso el actor podrá crear un Sub-Ítem para ser utilizado en la estructura de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- El Ítem no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña "Plantillas" desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú "Sub-Ítem" 4.- El sistema despliega el formulario de creación de Sub-Ítem y la lista con los Sub-Ítem existentes. 5.- El actor debe llenar los campos correspondientes para la creación del Sub-Ítem, los cuales son nombre y descripción, y permitir al actor hacer del sub-ítem un elemento de selección Múltiple, una vez completos el actor debe presionar el botón "guardar" 6.- El sistema guardará el Sub-Ítem en la base de datos y refrescará la lista de Sub-Ítems agregando el recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- El nuevo SubÍtem debe estar disponible en el sistema para usarlo en la estructuración de plantillas.	

Caso de uso Modificar Sub-Ítem

Nombre:	Modificar Sub-Ítem
Req. Asociados:	9- 10
Descripción: En este caso de uso el usuario podrá editar la información de un Sub-Ítem.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- El Sub-Ítem debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Sub-Ítem” 4.- El sistema despliega el formulario de creación/edición de Sub-Ítem y la lista con los Sub-Ítem existentes. 5.- El actor debe seleccionar el Sub-ítem a modificar desde la lista de Sub-ítem. 6.- El sistema despliega en el formulario de creación/edición de Sub-ítem los datos de éste. 7.- El actor debe modificar los campos correspondientes para la creación/edición del Sub-Ítem, los cuales son nombre, descripción y opción “selección Múltiple”, una vez completos el actor debe presionar el botón “guardar” 8.- El sistema guardará las modificaciones del Sub-Ítem en la base de datos y refrescará la lista de Sub-Ítem.	
Flujo Alternativo: No Aplica	
Post Condiciones: El Sub-Ítem debe ser modificado en la base de datos.	

Caso de uso Eliminar Sub-Ítem

Nombre:	Eliminar Sub-Ítem
Req. Asociados:	9- 10
Descripción: En este caso de uso el actor podrá eliminar Sub-Ítem existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- El Sub-Ítem debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Plantillas" desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú "Sub-Ítem" 4.- El sistema despliega el formulario de creación de Sub-Ítem y la lista con los Sub-Ítems existentes. 5.- El actor busca el Sub-Ítem requerida en la lista de Sub-Ítem y presiona el botón "Eliminar" 6.- El sistema cambia el estado del Sub-Ítem a desactivado y lo elimina de las listas de Sub-Ítem activos.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de Sub-Ítems, aquel que se eliminó no debe aparecer listado.	

Caso de uso Crear Valores

Nombre:	Crear Valores
Req. Asociados:	9-10
Descripción: En este caso de uso el actor podrá crear un Valor para ser utilizado en la estructura de una plantilla	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente ingresado al sistema. 2.- El Valor no debe existir	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Valores” 4.- El sistema despliega el formulario de creación/edición de Valores y la lista con los Valores existentes. 5.- El actor debe llenar los campos correspondientes para la creación del Valor, los cuales son nombre y descripción, y tipo de valor, una vez completos el actor debe presionar el botón “guardar” 6.- El sistema guardará el Valor en la base de datos y refrescará la lista de Valores agregando el recientemente creada.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- El nuevo Valores debe estar disponible en el sistema para usarlo en la estructuración de plantillas.	

Caso de uso Modificar Valor

Nombre:	Modificar Valor
Req. Asociados:	9-10
Descripción: En este caso de uso el usuario podrá editar la información de un Valor.	
Actores: Administrador	
Precondiciones: 1.- El actor debe haber accedido al sistema previamente 2.- El Valor debe existir en el sistema	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Valores” 4.- El sistema despliega el formulario de creación/edición de Valores y la lista con los Valores existentes. 5.- El actor debe seleccionar el Valor a modificar desde la lista de Valor. 6.- El sistema despliega en el formulario de creación/edición de Valores los datos de éste. 7.- El actor debe modificar los campos correspondientes para la creación/edición del Valor, los cuales son nombre, descripción y tipo, una vez completos el actor debe presionar el botón “guardar” 8.- El sistema guardará las modificaciones del Valor en la base de datos y refrescará la lista de Valor.	
Flujo Alternativo: No Aplica	
Post Condiciones: El Valor debe ser modificado en la base de datos.	

Caso de uso Eliminar Valor

Nombre:	Eliminar Valor
Req. Asociados:	9-10
Descripción: En este caso de uso el actor podrá eliminar un Valor existente en el sistema	
Actores: Administrador	
Precondiciones: 1.- El actor debe estar previamente identificado en el sistema. 2.- El Valor debe existir.	
Flujo Normal: 1.- El actor debe presionar la pestaña “Plantillas” desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú “Valores” 4.- El sistema despliega el formulario de creación de Valor y la lista con los Valores existentes. 5.- El actor busca el Valor requerida en la lista de Valor y presiona el botón “Eliminar” 6.- El sistema cambia el estado del Valor a desactivado y lo elimina de las listas de Valores activos.	
Flujo Alternativo: No Aplica	
Post Condiciones: 1.- En un posterior listado de Valores, aquel que se eliminó no debe aparecer listado.	

Caso de uso Estructurar Plantilla

Nombre:	Estructurar Plantilla / Modificar Estructura de Plantilla
Req. Asociados:	8
Descripción: En este caso de uso el actor podrá estructurar una plantilla de acuerdo a la necesidad de la inspección en la que será utilizada la plantilla.	
Actores: Administrador	
Precondiciones: 1.- El usuario debe estar identificado en el sistema. 2.- La plantilla a estructurar debe existir en el sistema. 3.- Todos los elementos de estructuración a utilizar deben existir en el sistema.	
Flujo Normal: 1.- El actor debe presionar la pestaña "Plantillas" desde el menú de Navegación. 2.- El sistema debe desplegar la pantalla correspondiente a la administración de plantillas. 3.- El actor debe presionar el sub menú "Estructura" 4.- El sistema descargará en el PC cliente la aplicación de escritorio de estructuración y presentar el formulario de estructura y la lista de plantillas. 5.- El actor deberá buscar la plantilla que desea estructurar o modificar su estructura y cargarla al sistema. 6.- El sistema deberá desplegar la actual estructura de plantilla (vacía si no tiene). 7.- El actor deberá incluir o sacar ítems de la estructura de plantilla. 8.- Una vez decidida la estructura final el actor presiona el botón "guardar"	
Flujo Alternativo: 7.- El actor no desea modificar plantilla alguna, cierra la aplicación de escritorio temporal.	
Post Condiciones: La nueva estructura debe verse reflejada en el formulario de edición de plantilla (llenado de informe)	

En esta sección se ha mostrado el detalle de las funciones del sistema y su interacción con el usuario una vez montado, para dejar mas claras estas relaciones se han generado los diagramas de casos de uso, en donde podemos ver gráficamente cada función del sistema y la relación con el actor.

4.3 Diagramas de Casos de Uso

Casos de Uso General (todo el sistema):

El caso de uso de la figura 4.1muestra todos los módulos del sistema y los diferentes actores que intervienen en el funcionamiento del módulo. A continuación se detallan uno a uno estos casos de uso.

Figura 4.1: Caso de Uso general.

En la figura 4.2 se visualiza el Caso de Uso Gestionar Usuario:

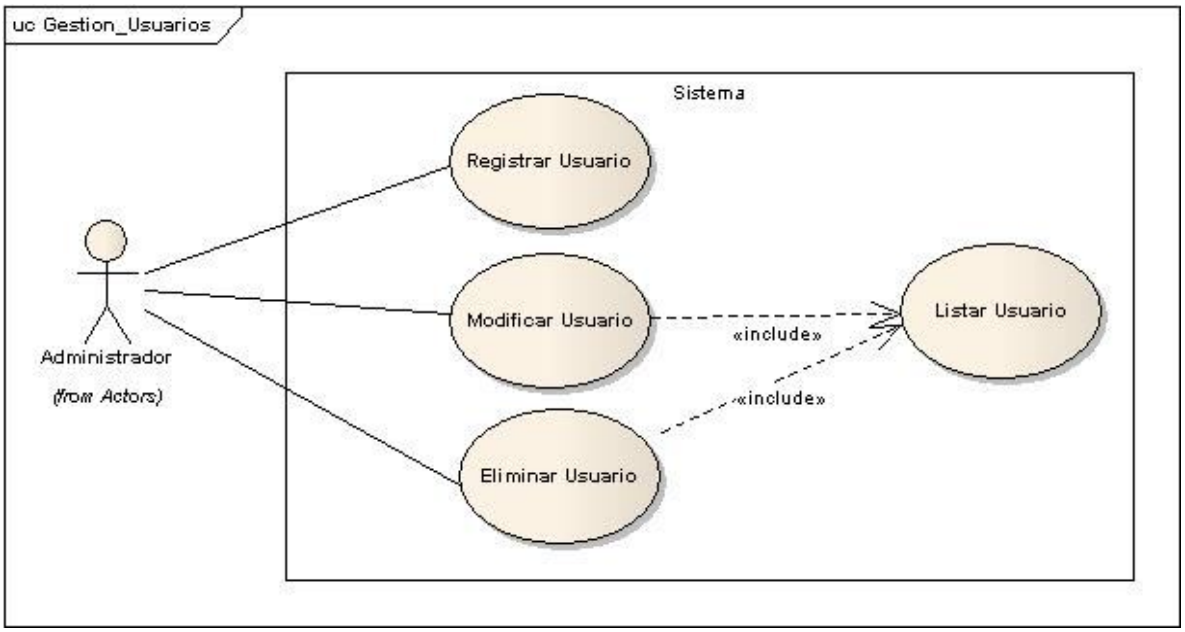


Figura 4.2: Caso de Uso Gestionar Usuario

En la Figura 4.3 se visualiza el Caso de Uso Gestionar Propiedad:

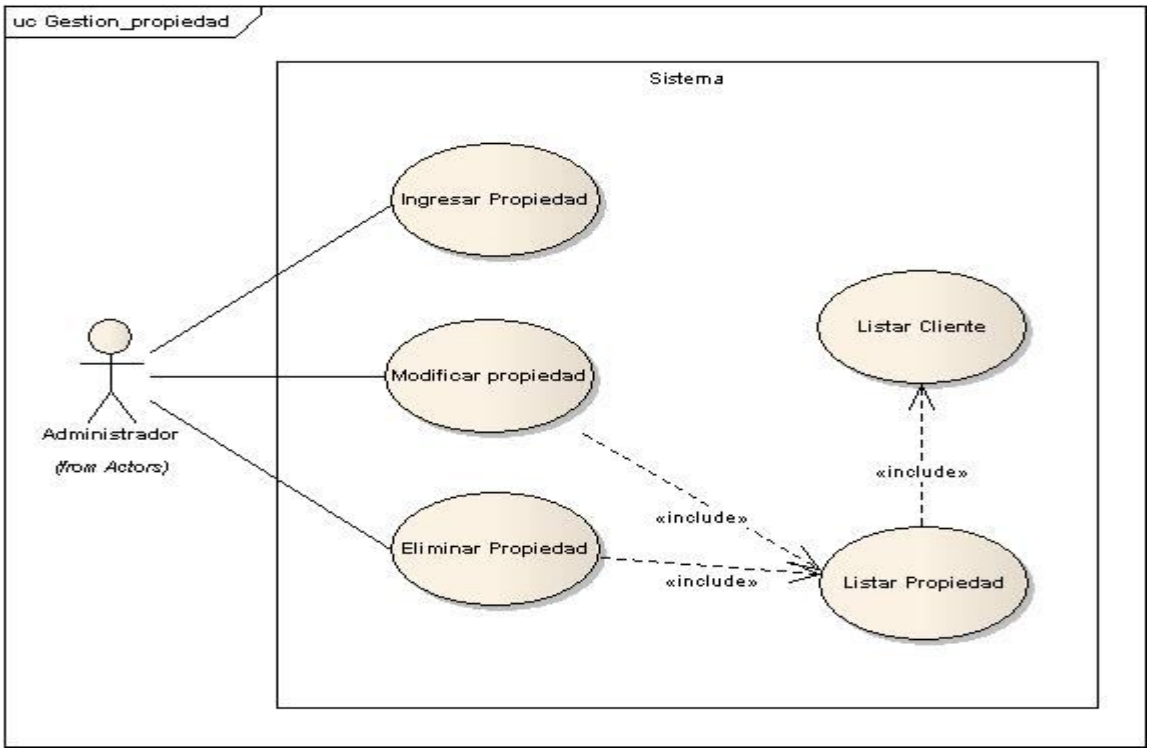


Figura 4.3: Caso de Uso Gestionar Propiedad

En la figura 4.4 se ve el Caso de Uso Gestionar OT:

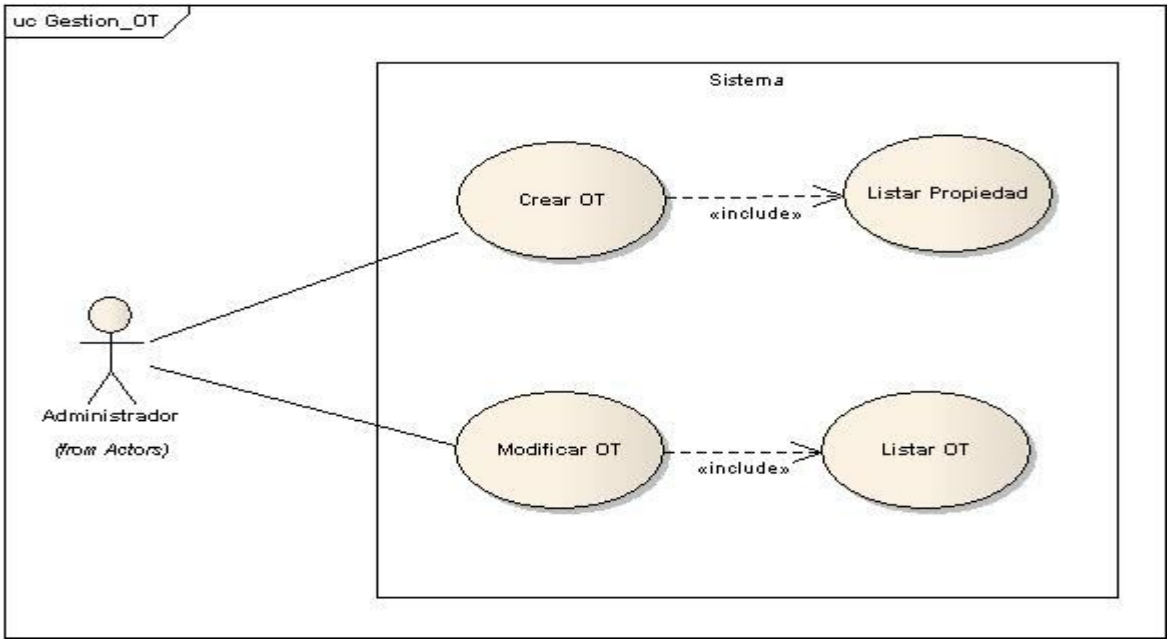


Figura 4.4: Caso de Uso Gestionar Orden de Trabajo

En la Figura 4.5 se observa el Caso de Uso gestionar Reporte:

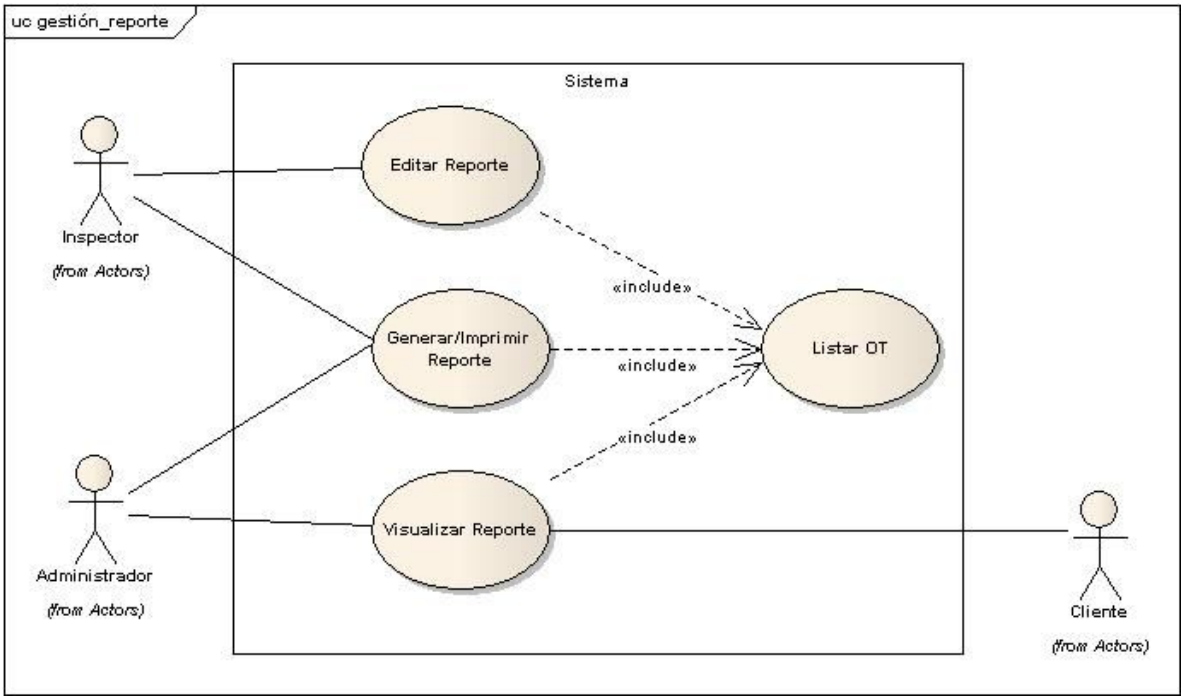


Figura 4.5: Caso de uso Gestionar Reporte

En la figura 4.6 se observa el Caso de Uso Gestionar Plantilla:

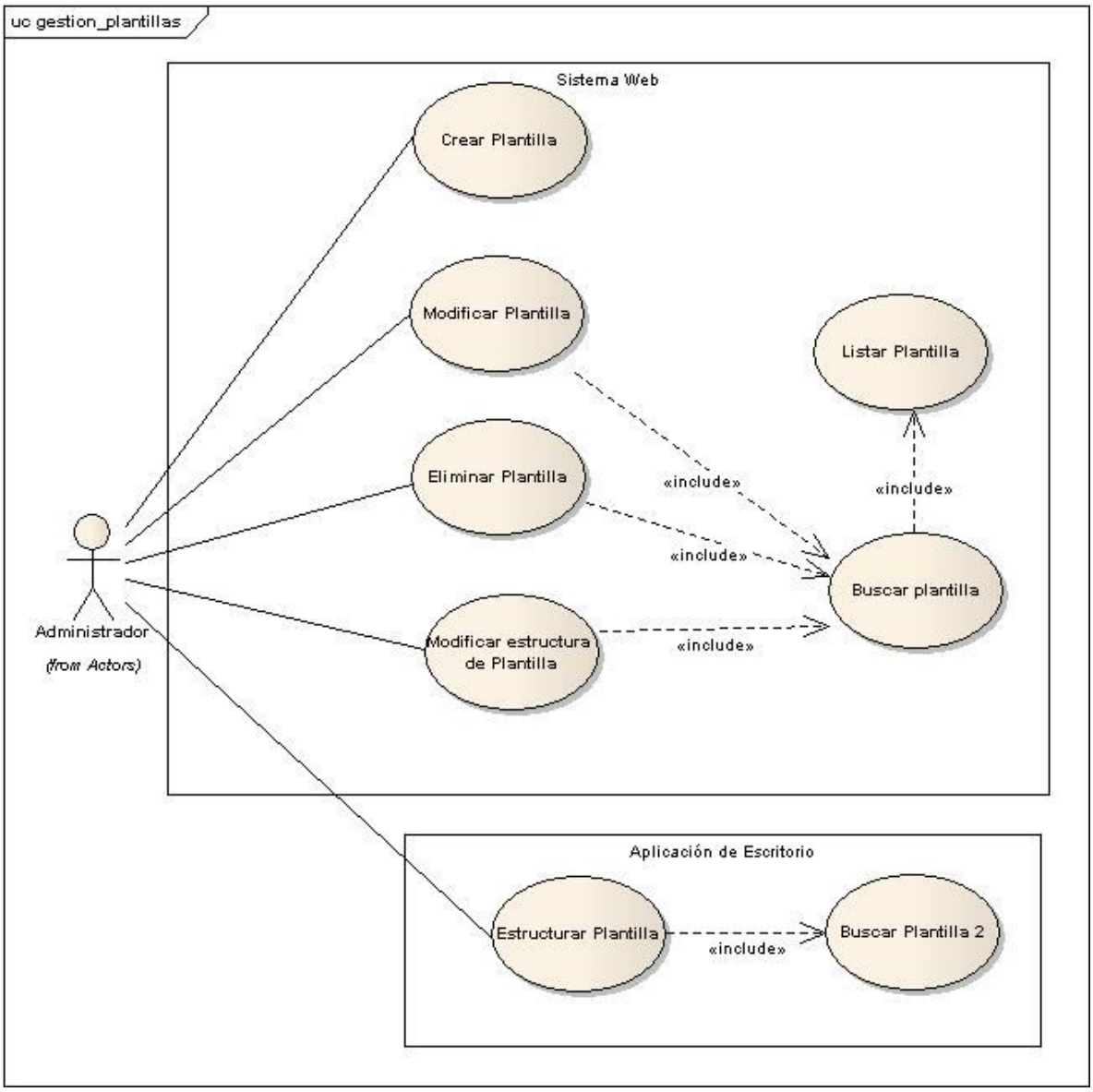


Figura 4.6: Caso de Uso Gestionar Plantilla

En la figura 4.7 se visualiza el Caso de Uso Gestionar Capítulos

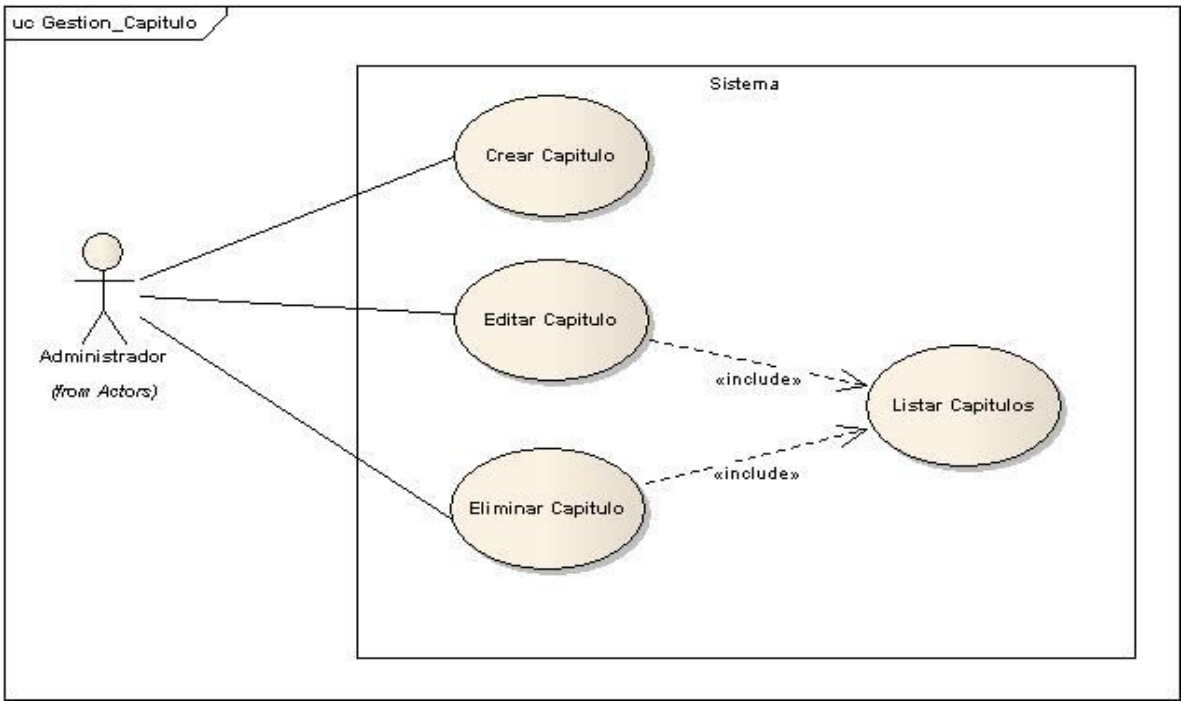


Figura 4.7: Caso de Uso Gestionar Capitulo

En la Figura 4.8 se observa el Caso de uso Gestionar Sección

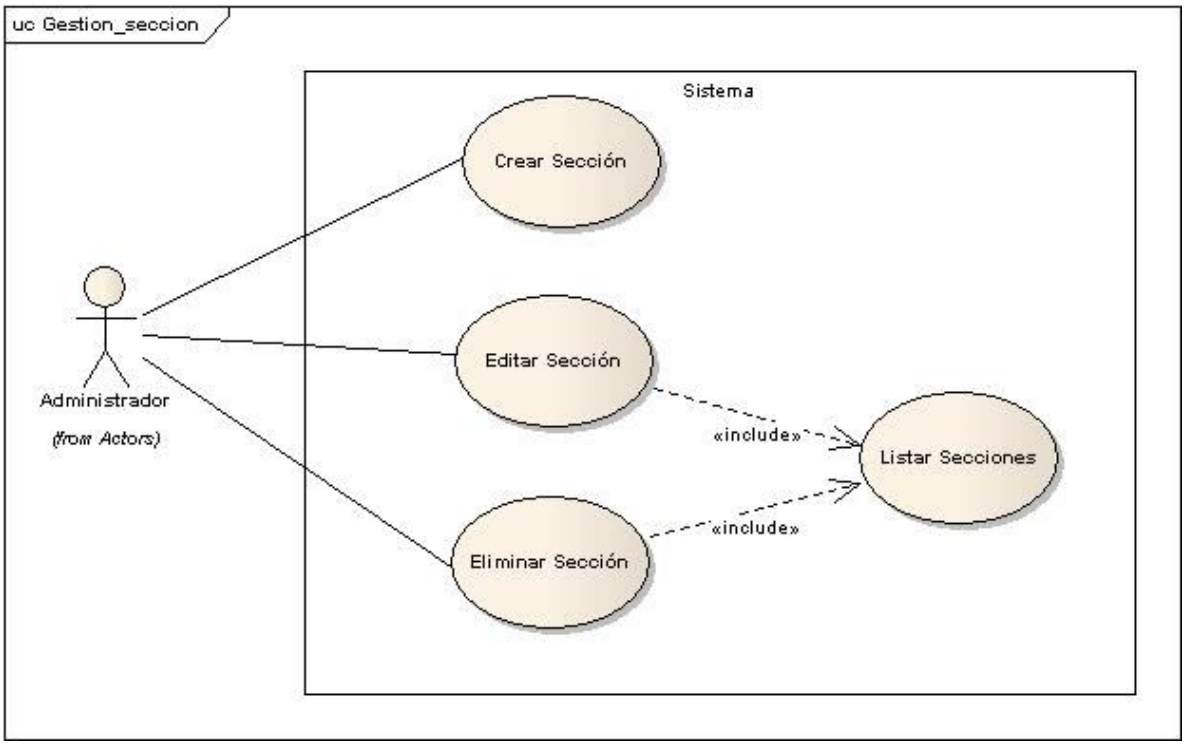


Figura 4.8: Caso de Uso Gestionar Sección

En la Figura 4.9 se observa el Caso de Uso Gestionar Ítem

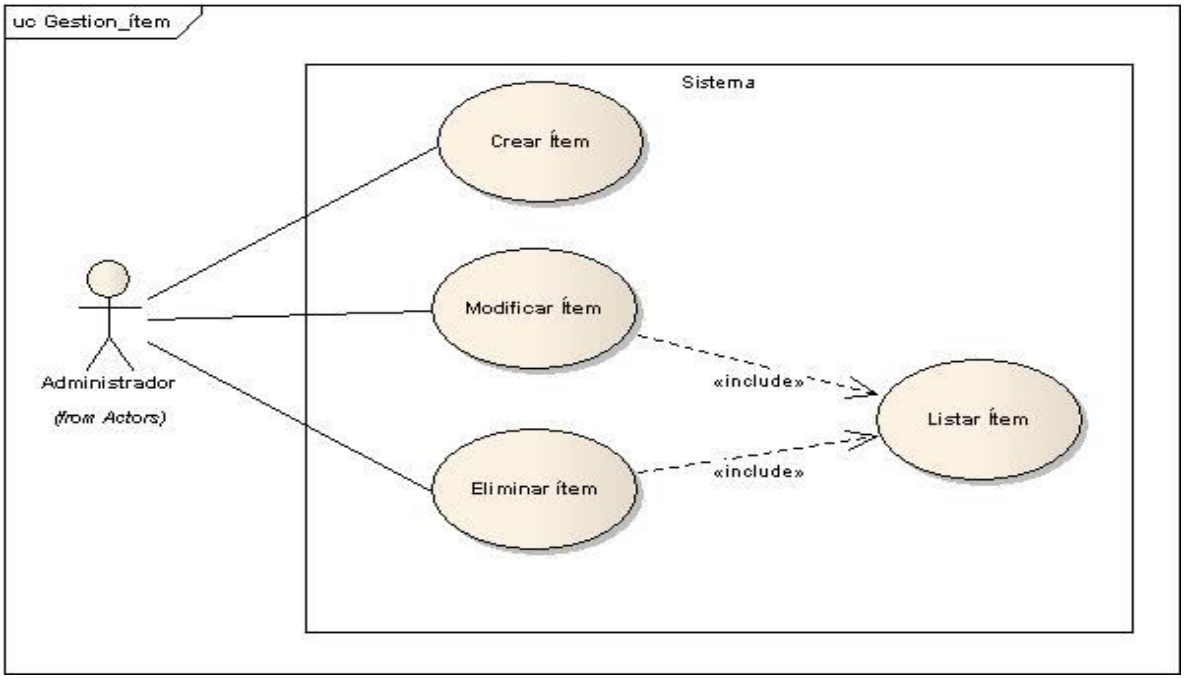


Figura 4.9: Caso de Uso Gestionar Ítem.

En la figura 4.10 se observa el Caso de Uso Gestionar Sub-Ítem

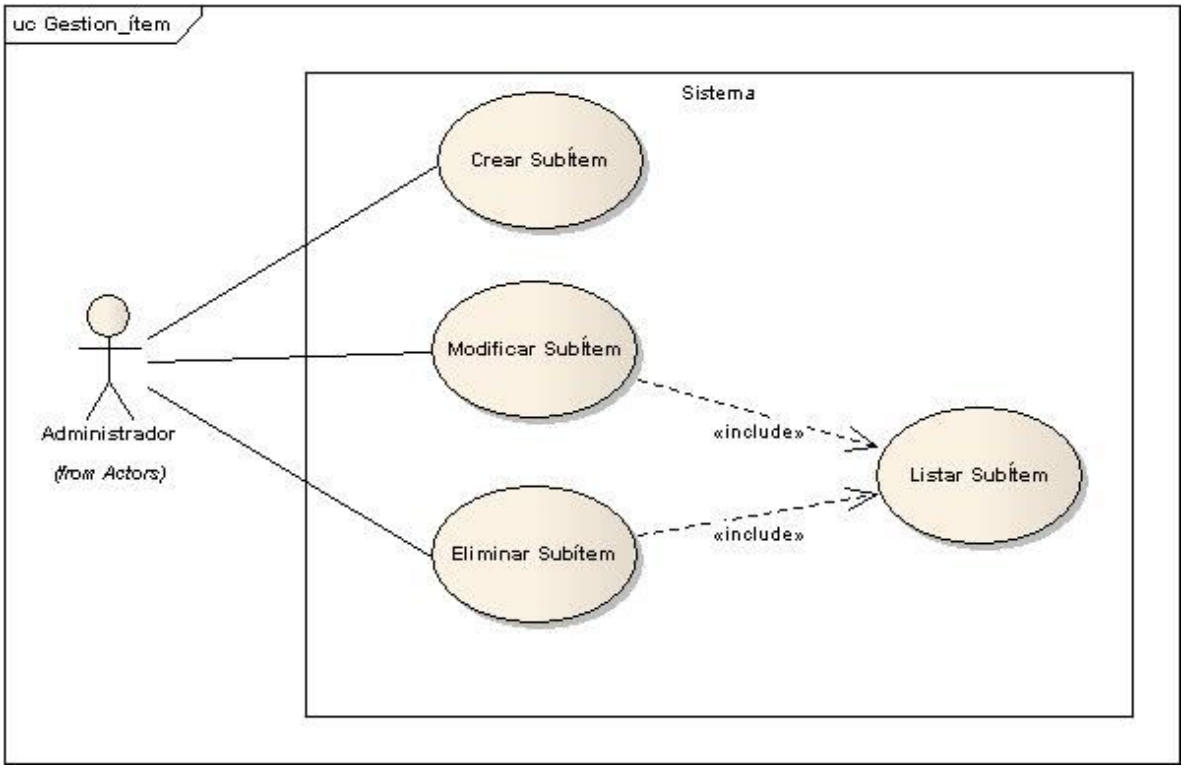


Figura 4.10: Caso de Uso Gestionar Sub-Ítem

En la figura 4.11 se aprecia el Caso de Uso Gestionar Valores

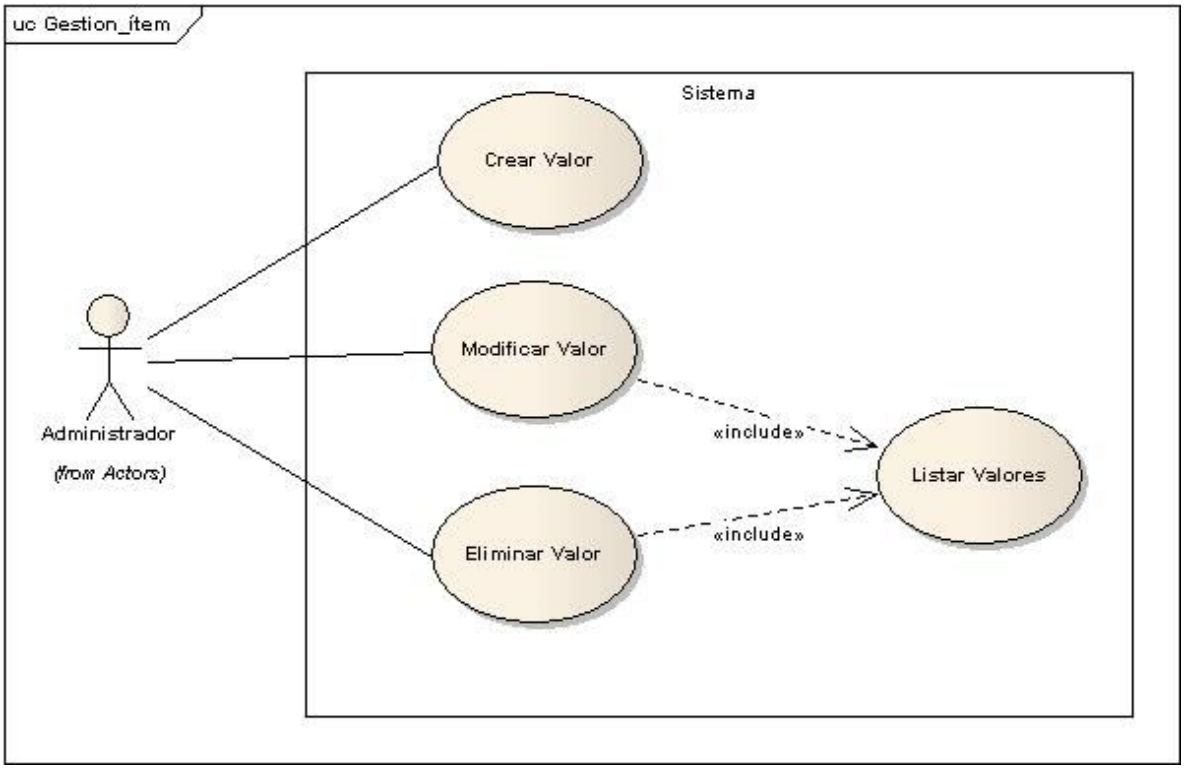


Figura 4.11: Caso de Uso Gestionar Valores

4.4 Diagramas de Actividad de las Principales Funciones.

Diagrama de actividad general de Mantenedores (Figura 4.12).

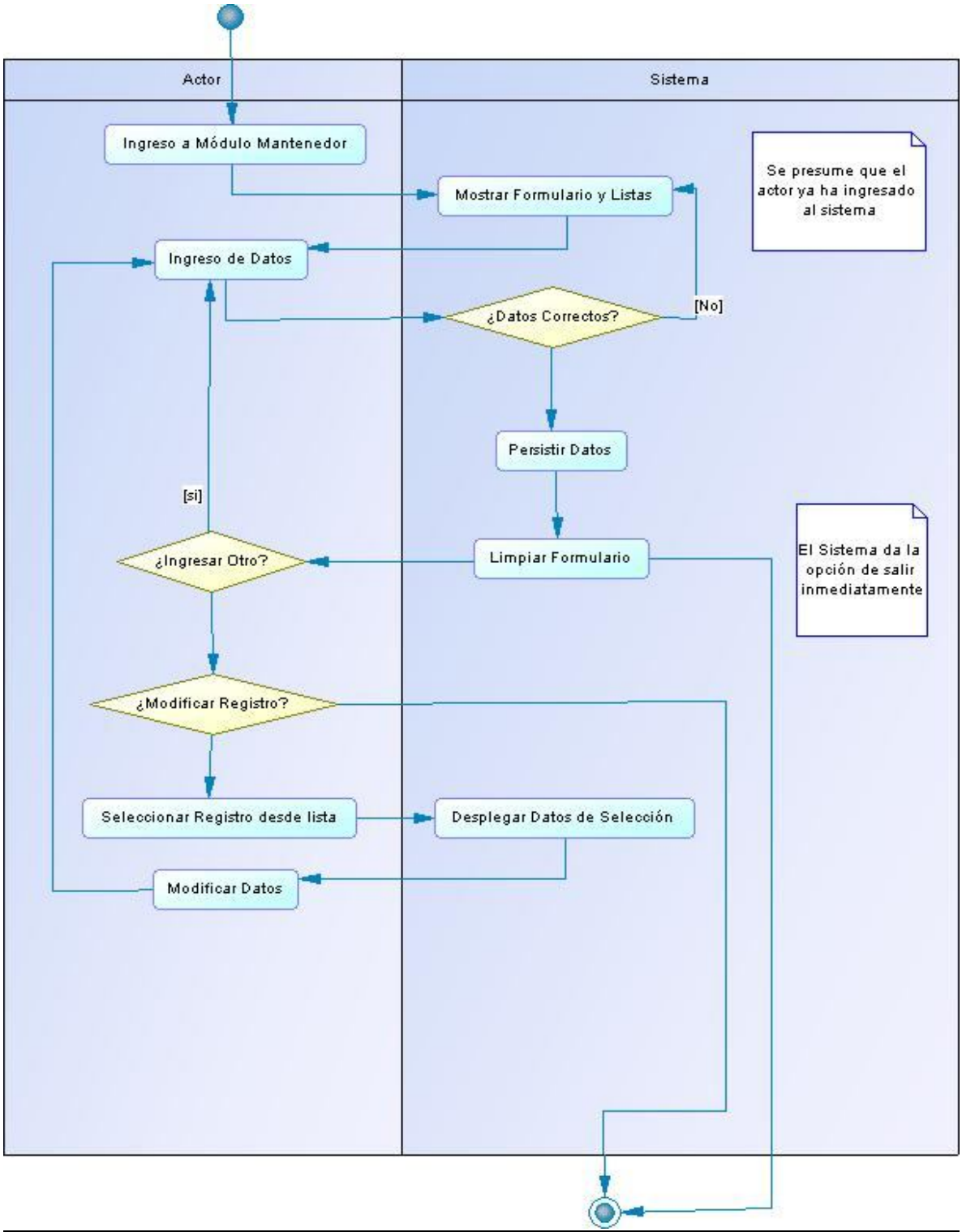


Figura 4.12: Diagrama de Actividad Mantenedores

Diagrama de Actividad Generar Informe (Figura 4.13):

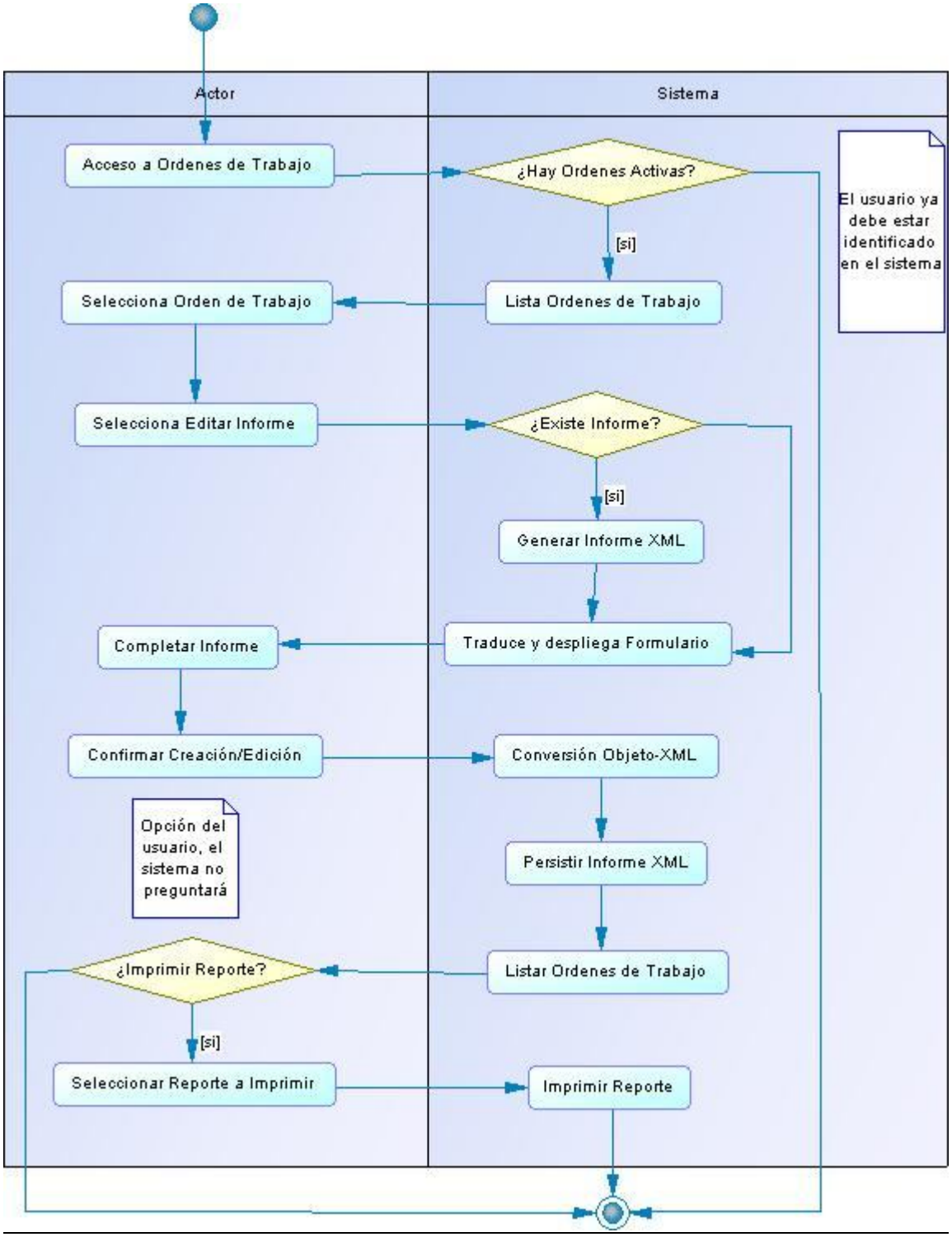


Figura 4.13: Diagrama de Actividad Generar Informe.

Diagrama de Actividad Estructurar Reporte (Figura 4.14):

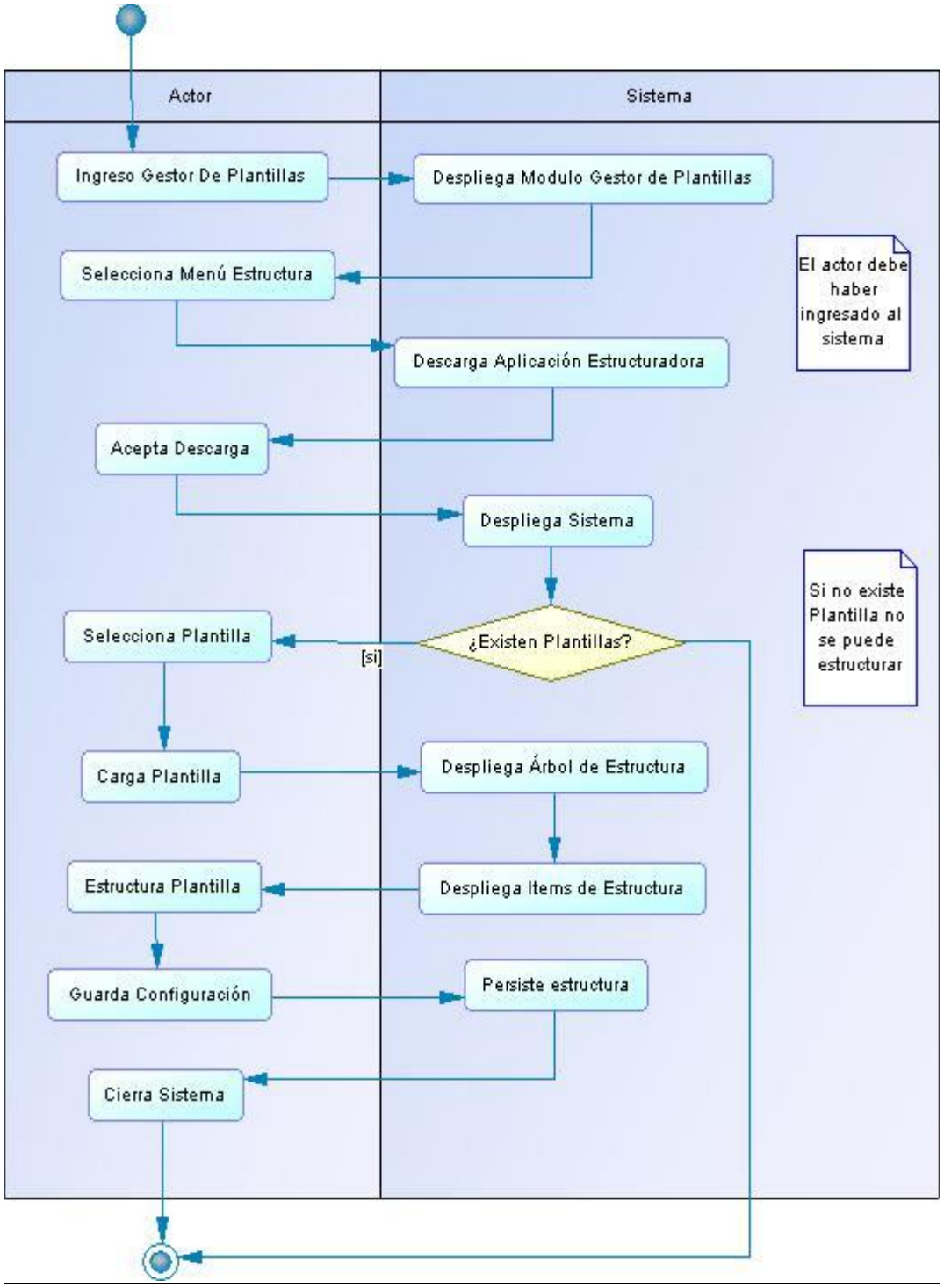


Figura 4.14: Diagrama de Actividad Estructurar Plantilla.

4.5 Diagramas de Clase.

Los diagramas de clase nos muestran la composición de los objetos que implementa el sistema y la relación entre ellos. En la figura 4.15 se puede apreciar el diagrama de clases del módulo principal del sistema "Home Inspector", éste se divide en 3 partes principales referido a estructura, informe y usuarios. Mas adelante se describen estas 3 secciones.

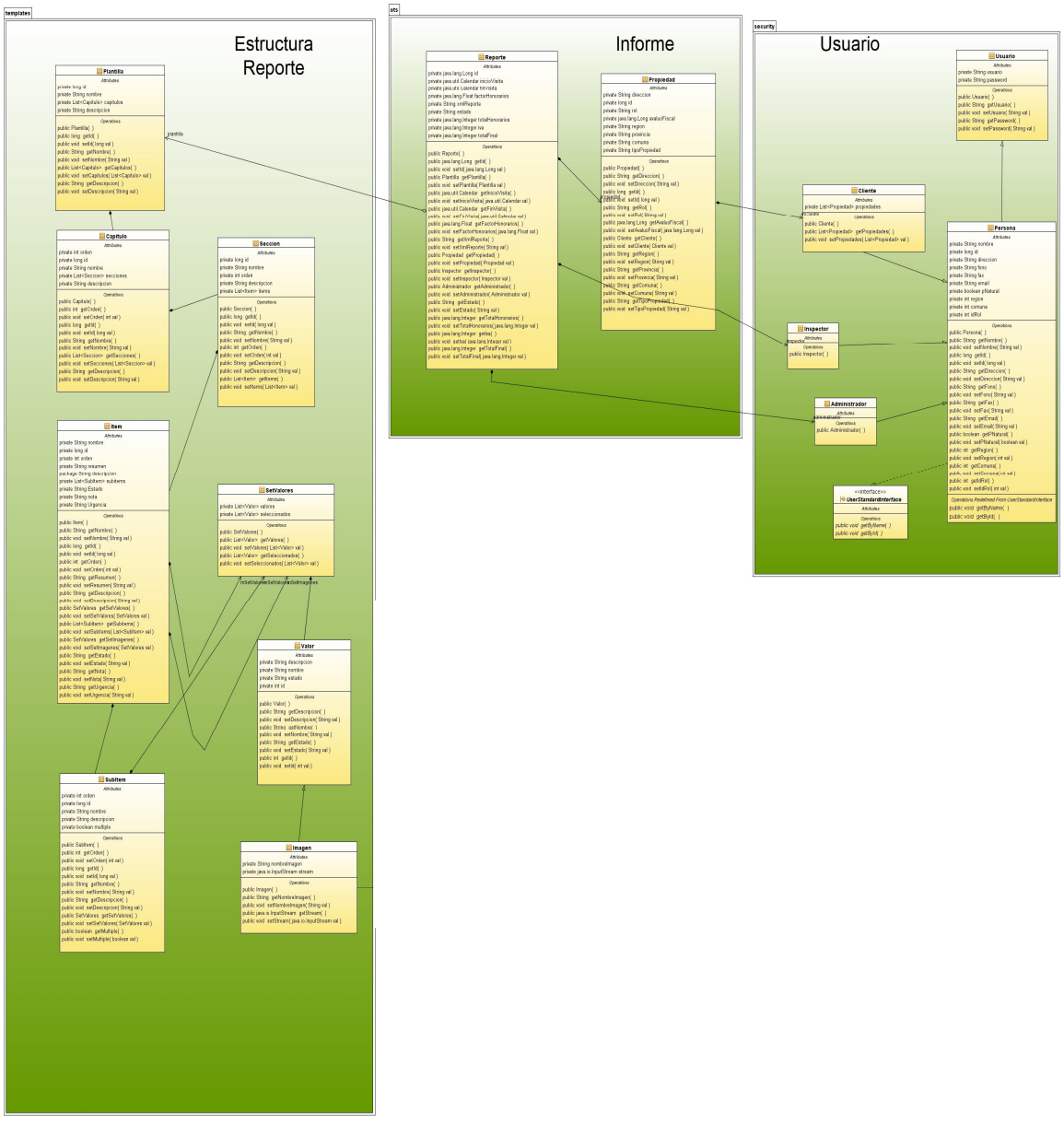


Figura 4.15: Diagrama de Clases del sistema Principal.

En la figura 4.16 se observan las clases correspondientes a la estructura de reporte, se presenta en 3 partes que se detallan a continuación:

Figura 4.16: Diagrama de clases correspondiente a estructura de reporte.

En la Figura 4.17 se observa la parte 1 que corresponde a la jerarquía de objetos de la estructura del reporte.

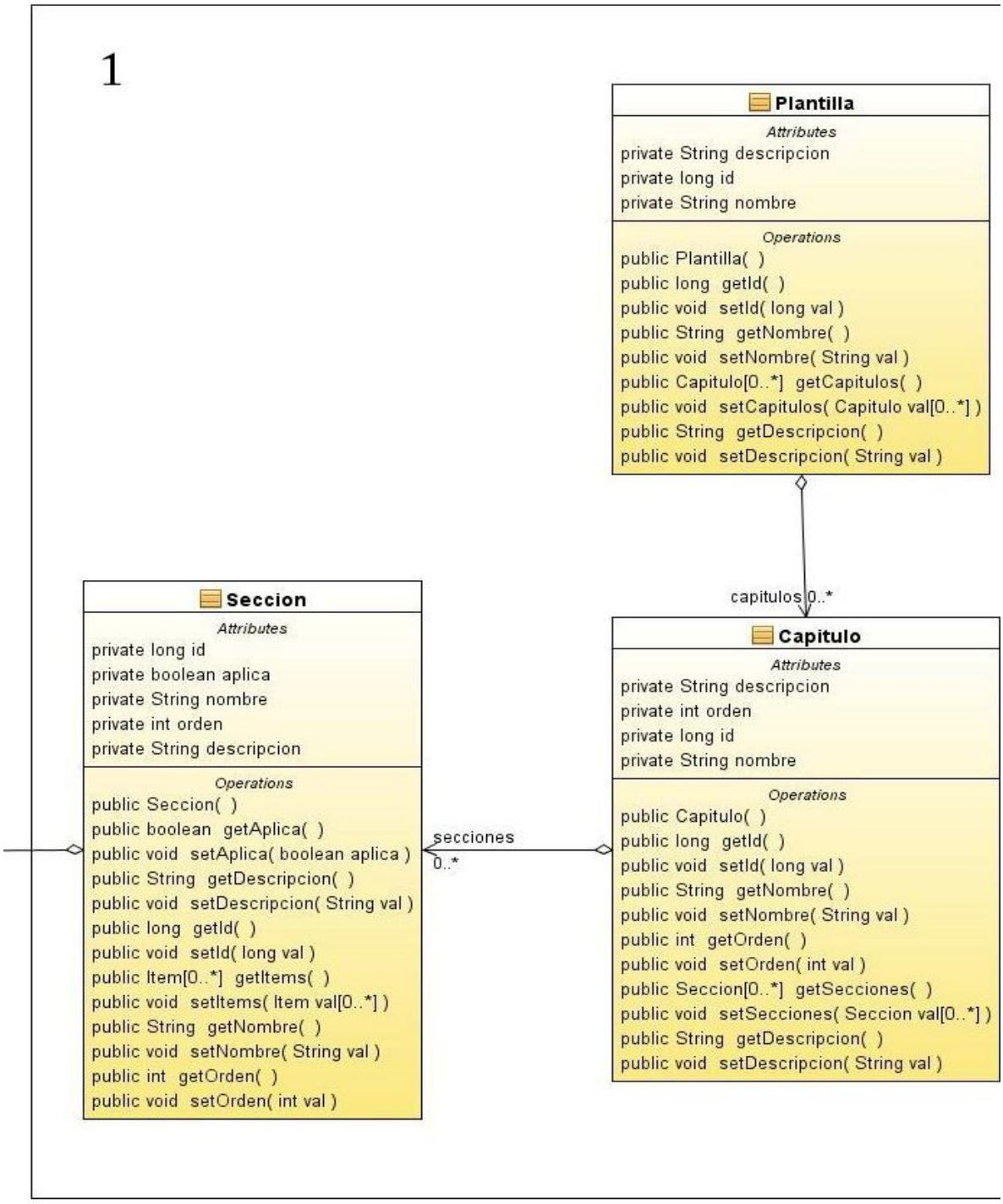


Figura 4.17: Sección uno Diagrama de Clases de estructura

En la figura 4.17 se observan las primeras 3 clases en la jerarquía de objetos de la estructura de un informe, la primera es la clase Plantilla, es esta quien tiene a su haber el resto de objetos que la conforman, comenzando desde Capitulo y terminando en valor.

Cada una de éstas clases poseen atributos, los cuales son apreciables en cada figura que se presenta, por ejemplo, la clase Plantilla posee los atributos llamados Id, Nombre y Descripción, cada uno de éstos determina el tipo de objeto, asociados a éstos atributos se encuentran sus respectivos métodos, los cuales determinan el comportamiento de ésta clase. Cada método en este diagrama de clases (incluidas las tres divisiones hechas) se encarga de setear y capturar los valores reales de cada atributo, a éstos métodos se les llama Getter and Setter (de get y set).

La descripción anterior es válida para todo este diagrama de clases.

En la figura 4.18 se aprecia la sección 2 del diagrama de estructura de plantillas, que proviene desde la clase Sección, siguiendo con la jerarquía de objetos.

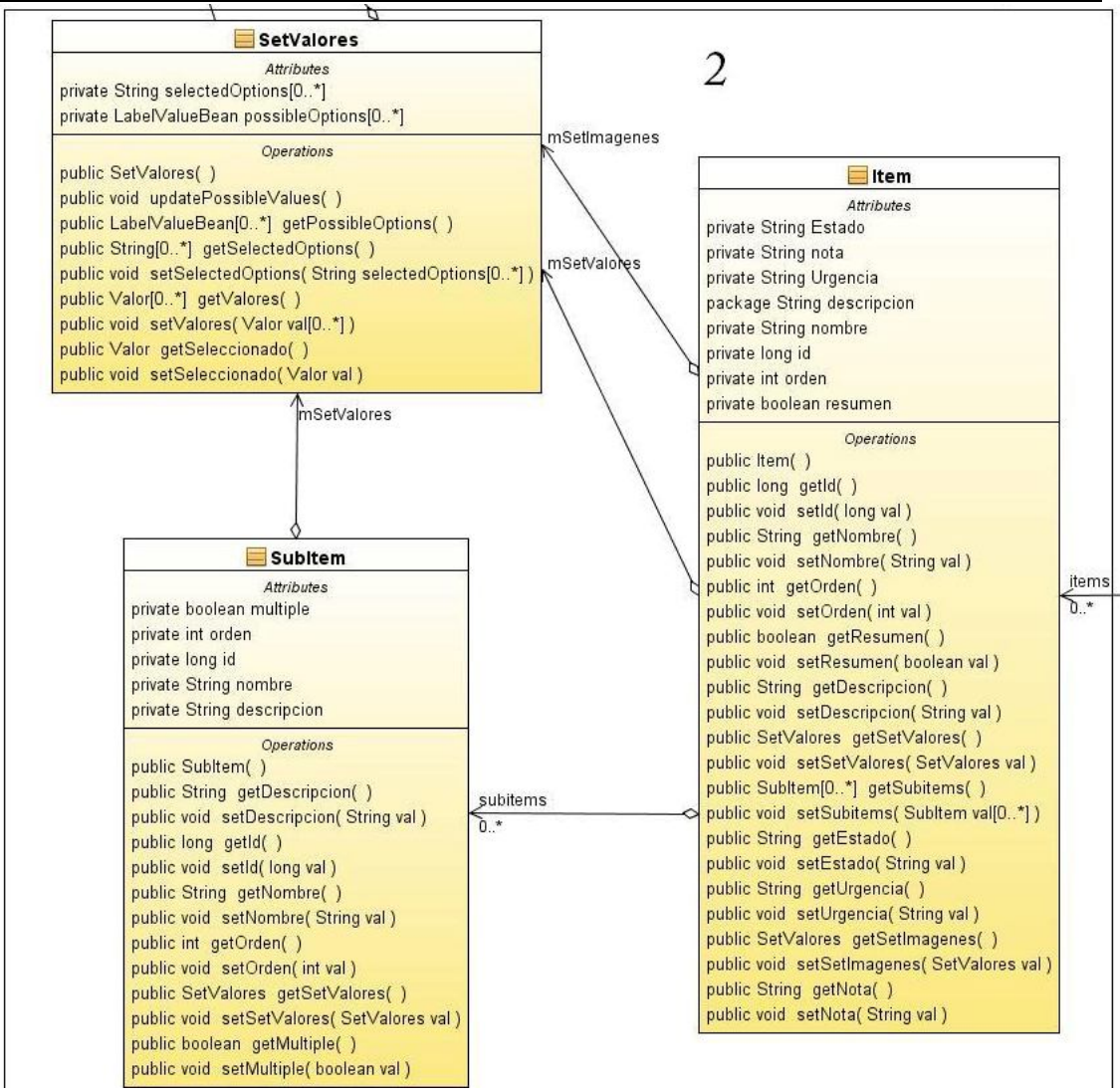


Figura 4.18: Sección 2 diagrama de clases de estructura de Plantilla

En la figura 4.18 se observa la clase Ítem y sub-Ítem, las cuales siguen en la jerarquía de objetos desde sección. Además se observa la clase setValores asociada a ellas, esta clase permite asociar múltiples objetos valor a la clase sub-ítem.

En la figura 4.19 se muestra la sección 3 correspondiente al diagrama de clases de estructura de Plantillas, se observan en esta sección la última clase en la jerarquía de objetos, la cual corresponde a Valor. Además se encuentra presente la clase Imagen, que corresponde a un objeto de tipo imagen, el cuál puede estar presente en el informe final.

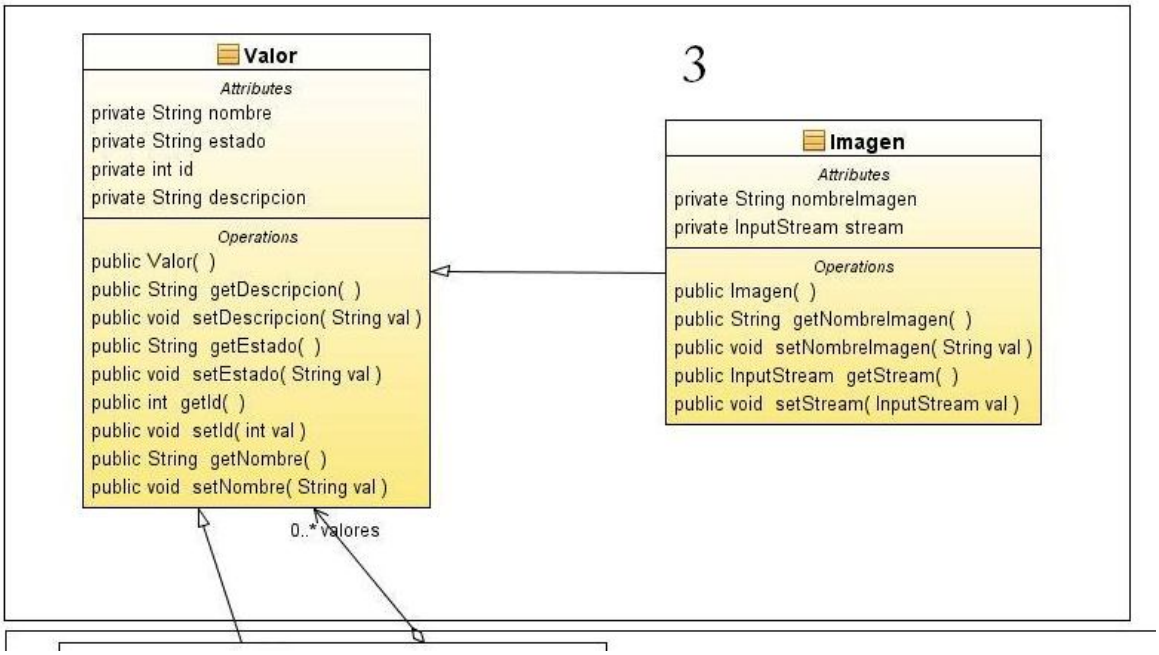


Figura 4.19: sección 3 diagrama de clases estructura de Plantilla.

En la Figura 4.20 se observa el Diagrama de Clases correspondiente al reporte, esta sección corresponde al diagrama general presentado en la figura 4.15

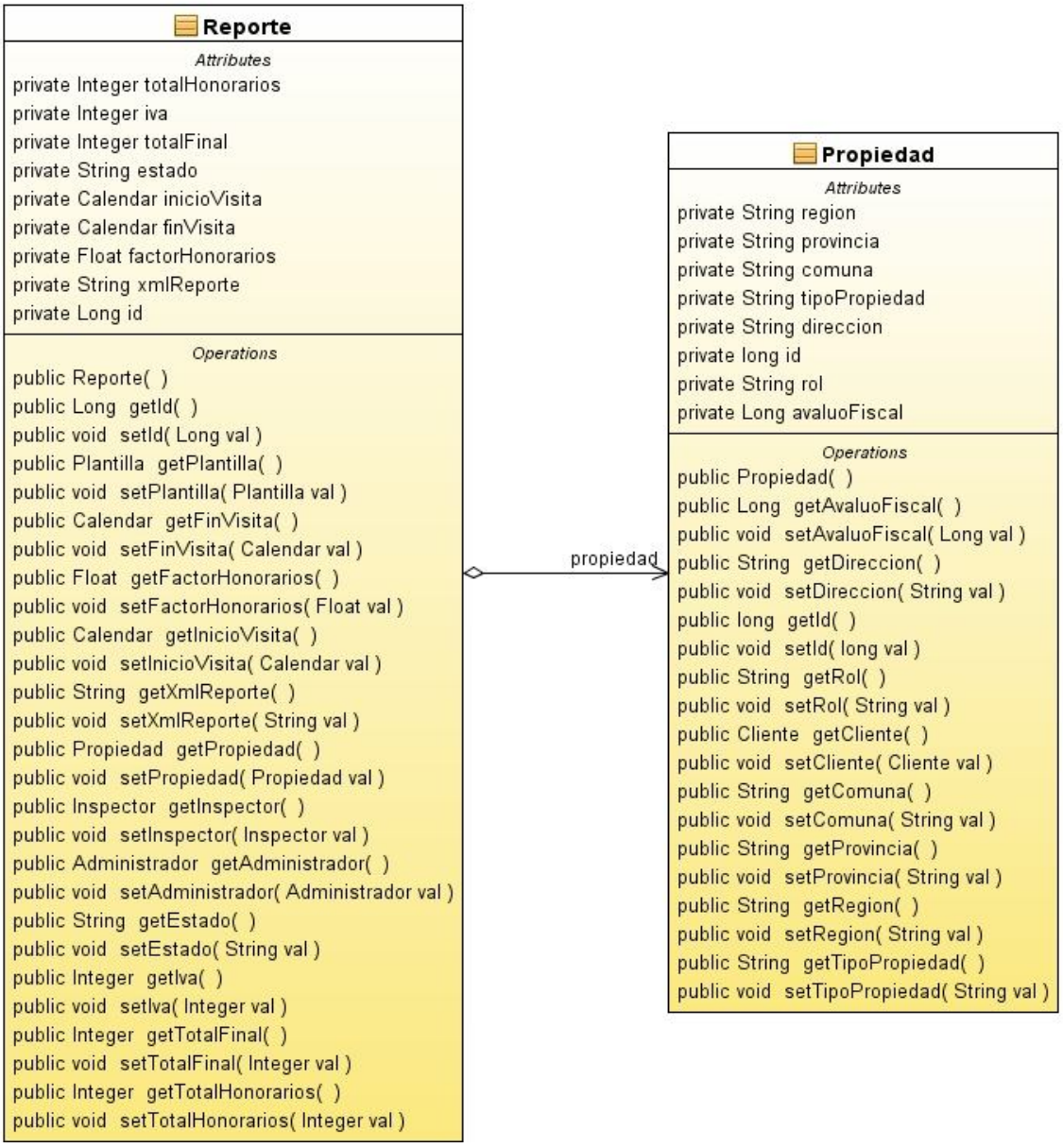


Figura 4.20: Diagrama de Clases sección Reporte.

En la figura 4.20 se puede observar la clase Propiedad, ésta clase contiene todos los atributos y métodos correspondientes a una propiedad de un cliente, contiene métodos getter y setter, los cuales cumplen la función de asignación y entrega de valores de cada atributo. Esta clase se asocia a la clase Reporte.

La clase reporte, se asocia a una propiedad y a su vez a una plantilla, en conjunto forman el objeto que se convertirá en el informe entregado por el sistema, éste contiene todos los atributos necesarios para la generación del informe o reporte, los cuales son apreciables en la clase y sus relaciones.

En la figura 4.21 se aprecia la sección del diagrama de clases principal correspondiente al los usuarios del sistema, la clase principal es la clase Persona, la cual posee todos los atributos de un usuario, además están las clases usuario, cliente, administrador e inspector, las cuales heredan los atributos y comportamiento de persona, agregando cada una de ellas además, comportamiento propio.

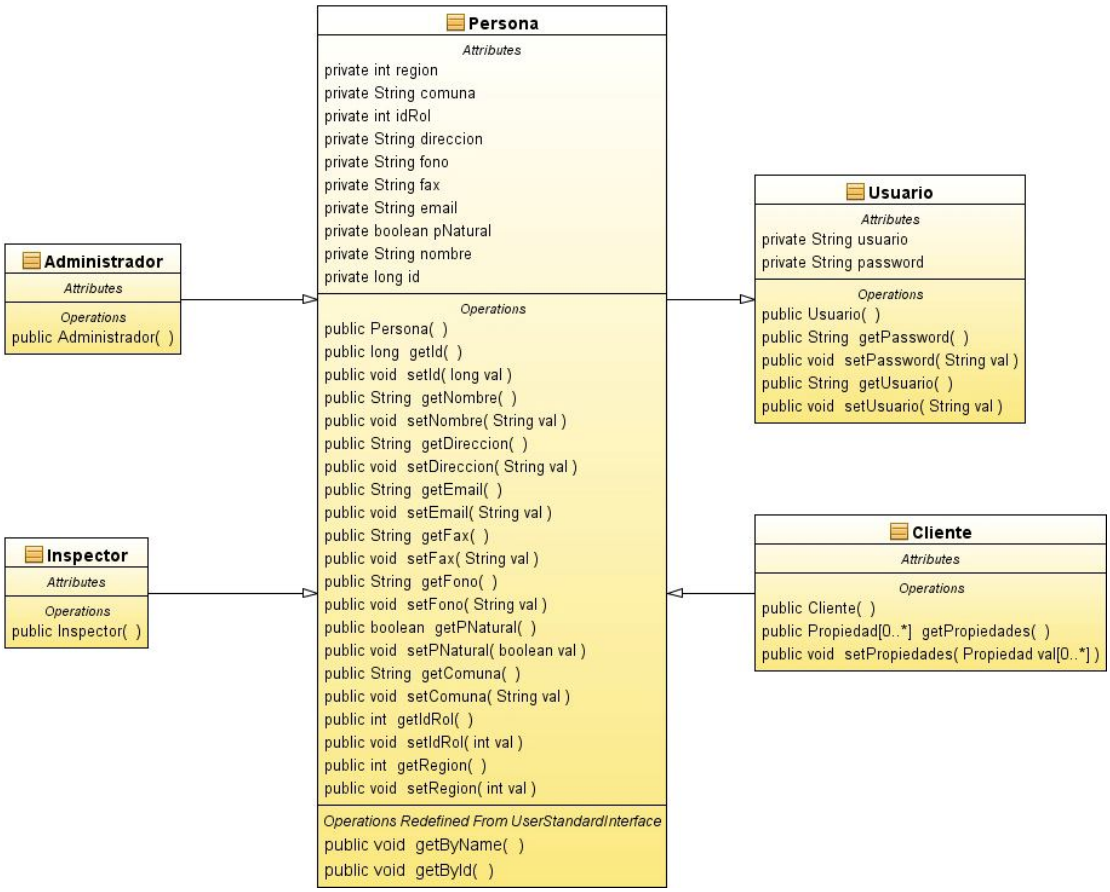


Figura 4.21: sección correspondiente a usuarios del diagrama de clases principal.

Capitulo 5

Diseño de BD e Interfaces

Introducción del Capitulo

En este capitulo se presenta la estructuración lógica y física de los datos, así como también el aspecto físico del sistema (capa de usuario).

5.1 Diseño de la Base de Datos

5.1.1 Diseño Lógico de la Base de datos

En la figura 5.1 se presenta el esquema lógico de la base de datos utilizada por el sistema

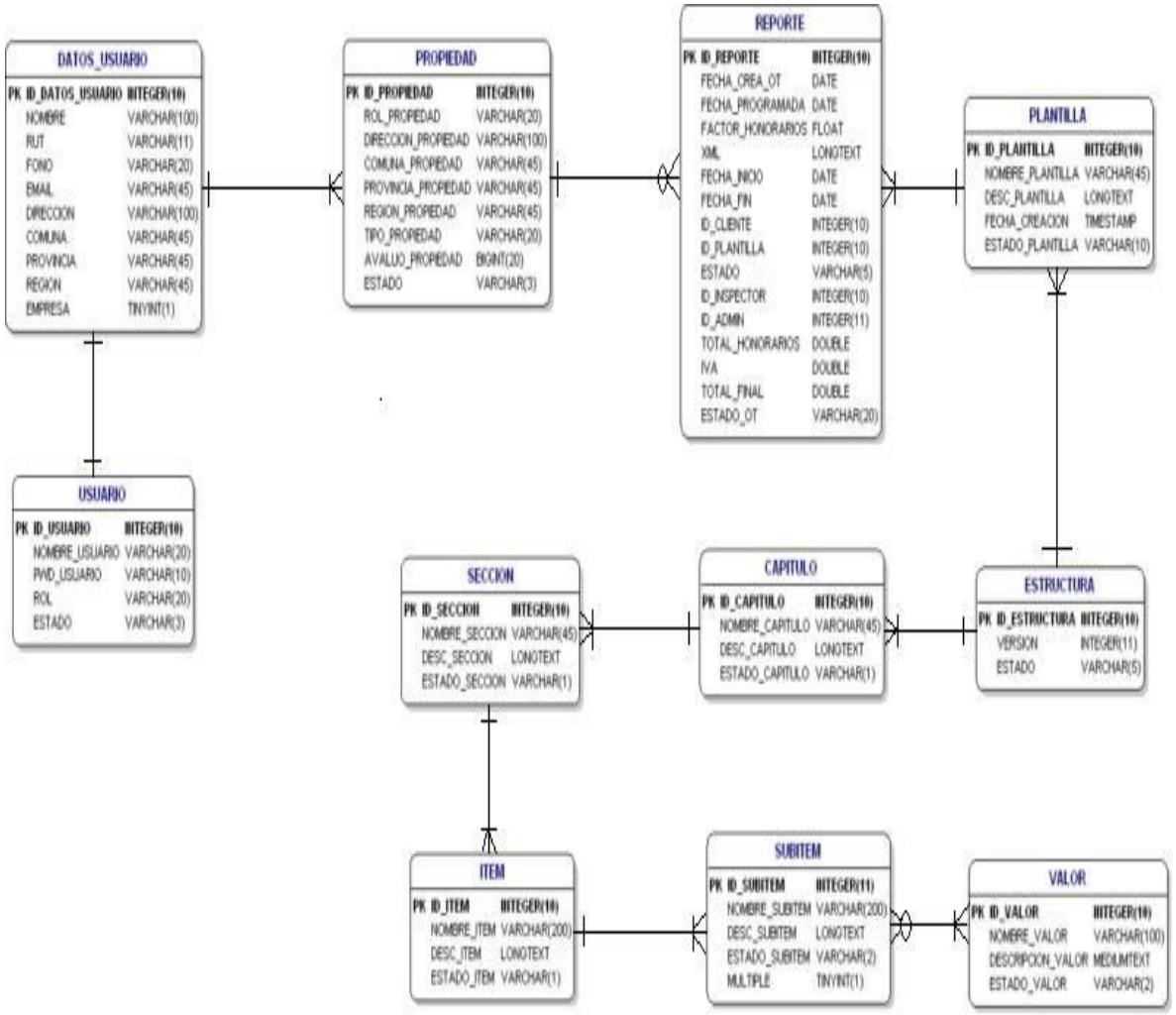
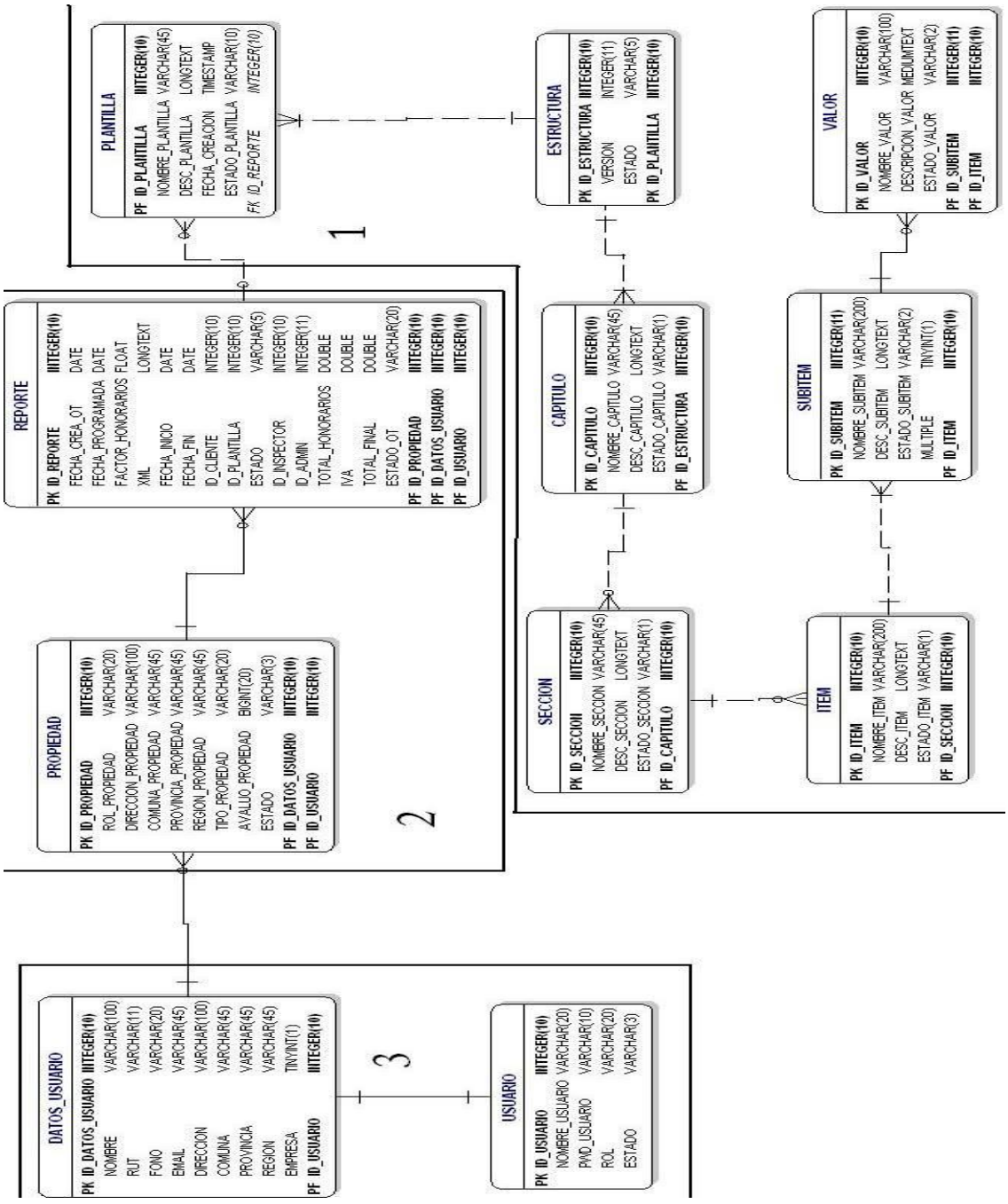


Figura 5.1: Modelo Relacional Lógico de la Base de Datos.

5.1.2 Diseño Físico de la Base de datos.

En la figura 5.2 se presenta el esquema físico de la base de datos utilizada por el sistema, éste esquema se muestra seccionado en tres partes, correspondientes a Informes y propiedades, Plantilla y estructura y usuarios.

Figura 5.2: Diseño físico de la Base de Datos



5.2 Diseño de Interfaces

5.2.1 Diseño Conceptual de Interfaces:

En esta sección se presenta el diseño de las pantallas previo a la realización del software, se debe considerar que estas pantallas pueden tener diferencias (no significativas) en comparación a las pantallas reales.

Figura 5.3 Pantalla de Acceso a Home Inspector:

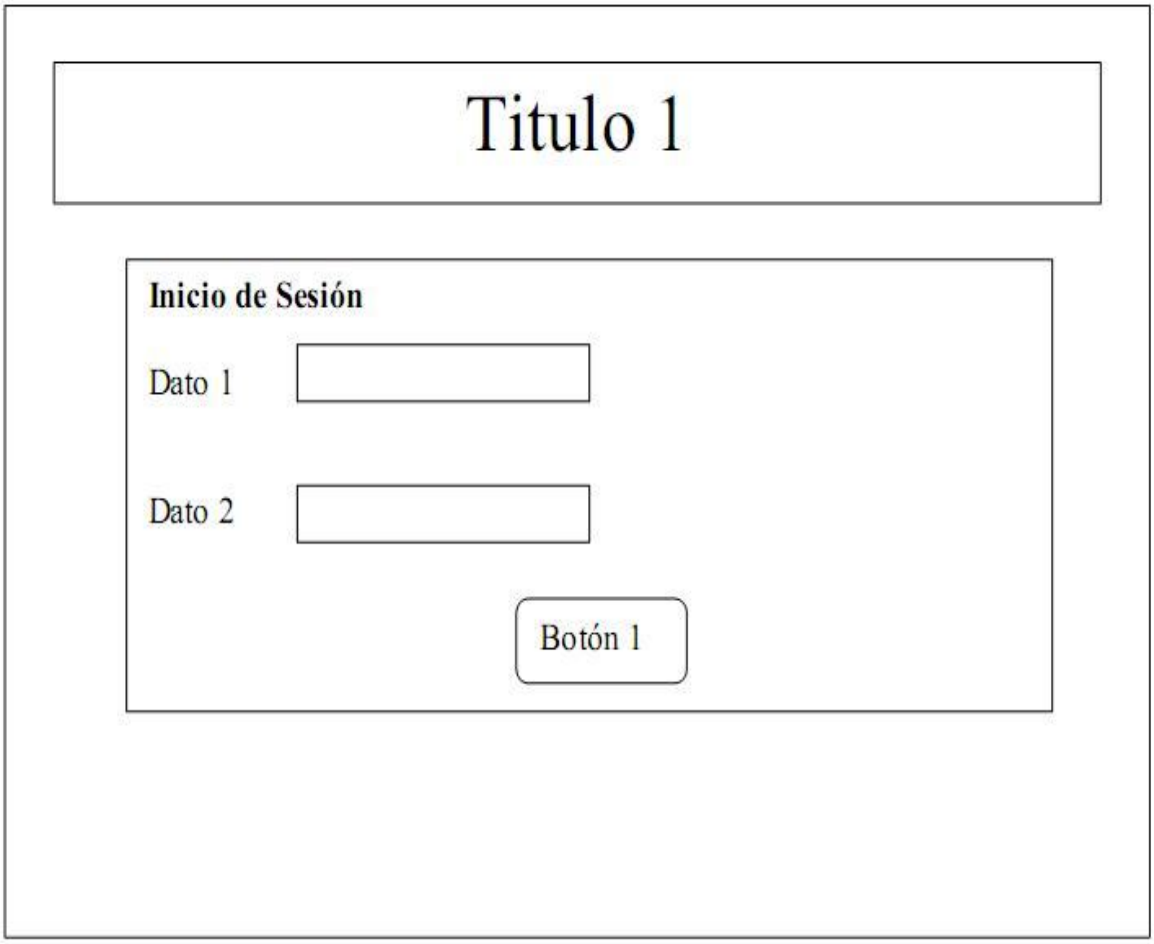


Figura 5.3: Diseño Conceptual Pantalla de Acceso

Descripción: En esta pantalla los usuarios acceden al sistema.
Botón 1:

Dato	Función
1	Nombre de Usuario otorgado por el administrador.
2	Clave de acceso al sistema.

Botón	Función
1	Para enviar los datos y validar el acceso.

Figura 5.4 Pantalla de Administración de usuarios:

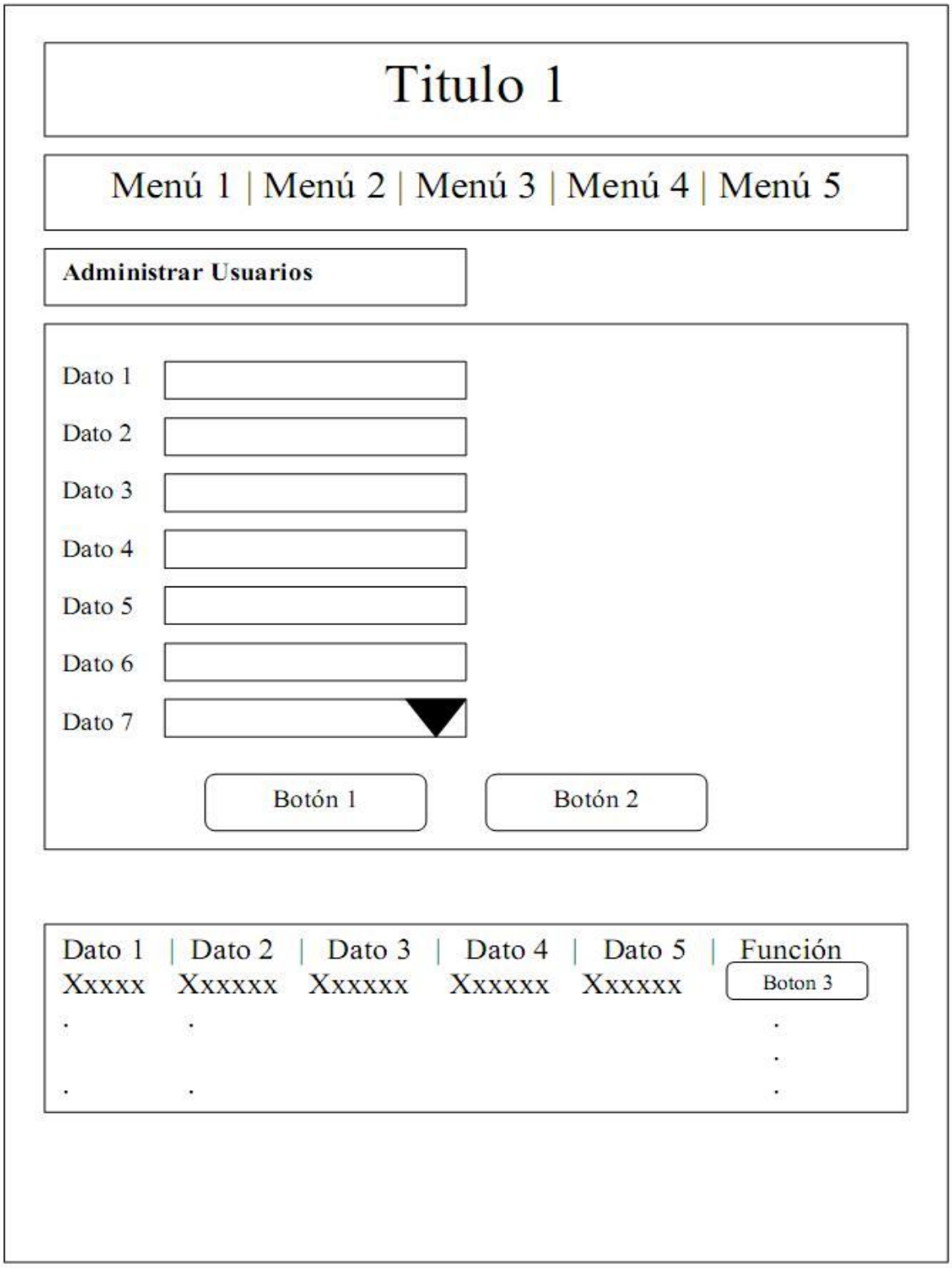


Figura 5.4: Diseño Conceptual Pantalla Gestión de Usuarios

Descripción: En esta pantalla el Administrador podrá ingresar, modificar y eliminar usuarios del sistema:

Menú	Función
1	Usuarios
2	Clientes.
3	Orden de Trabajo.
4	Plantillas.
5	Salida.

Dato	Función
1	Nombre del Usuario
2	Dirección del Usuario.
3	Correo Electrónico.
4	Teléfono.
5	fax.
6	Nombre de acceso al sistema (nick).
7	Tipo de Usuario (rol).

Botón	Función
1	Ingresar y/o Editar Usuario.
2	Limpiar formulario.
3	Acción para eliminar usuario

Xxxxxx: Datos de usuarios (lista)

Figura 5.5 Pantalla de Administración de Clientes y propiedades:

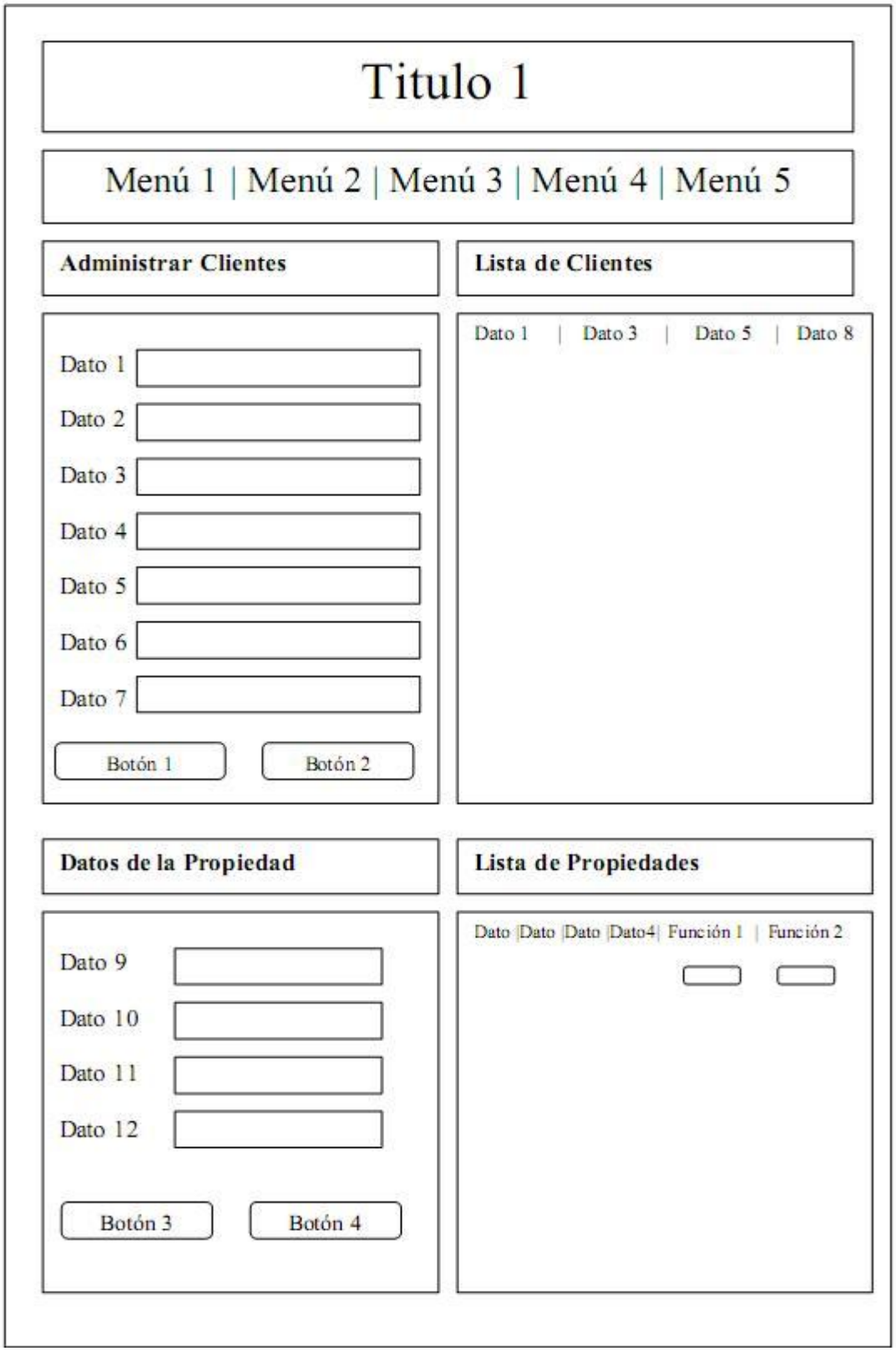


Figura 5.5: Diseño Conceptual Pantalla Gestión de Clientes y Propiedades

Descripción: En esta pantalla el Administrador podrá Ingresar, modificar y eliminar Usuarios (clientes) del sistema, así mismo podrá ingresar, modificar y eliminar propiedades asociadas al cliente. A través de las propiedades se podrá generar una nueva orden de trabajo asociada a una propiedad.

Menú	Función
1	Usuarios
2	Clientes.
3	Orden de Trabajo.
4	Plantillas.
6	Salida.

Dato	Función
1	Nombre
2	Dirección.
3	Teléfono.
4	Fax.
5	Correo Electrónico.
6	Nombre de usuario para acceso al sistema.
8	Número de Propiedades
9	Dirección de propiedad.
10	ROL de propiedad.
11	Avalúo Fiscal.
12	Tipo de Propiedad

Botón	Función
1	Editar usuario cliente.
2	Limpiar Formulario.
3	Ingresar/Editar propiedad.
4	Limpiar Formulario.
5	Eliminar Propiedad.
6	Generar Orden de Trabajo.

Figura 5.6 Pantalla de Administración Órdenes de Trabajo:

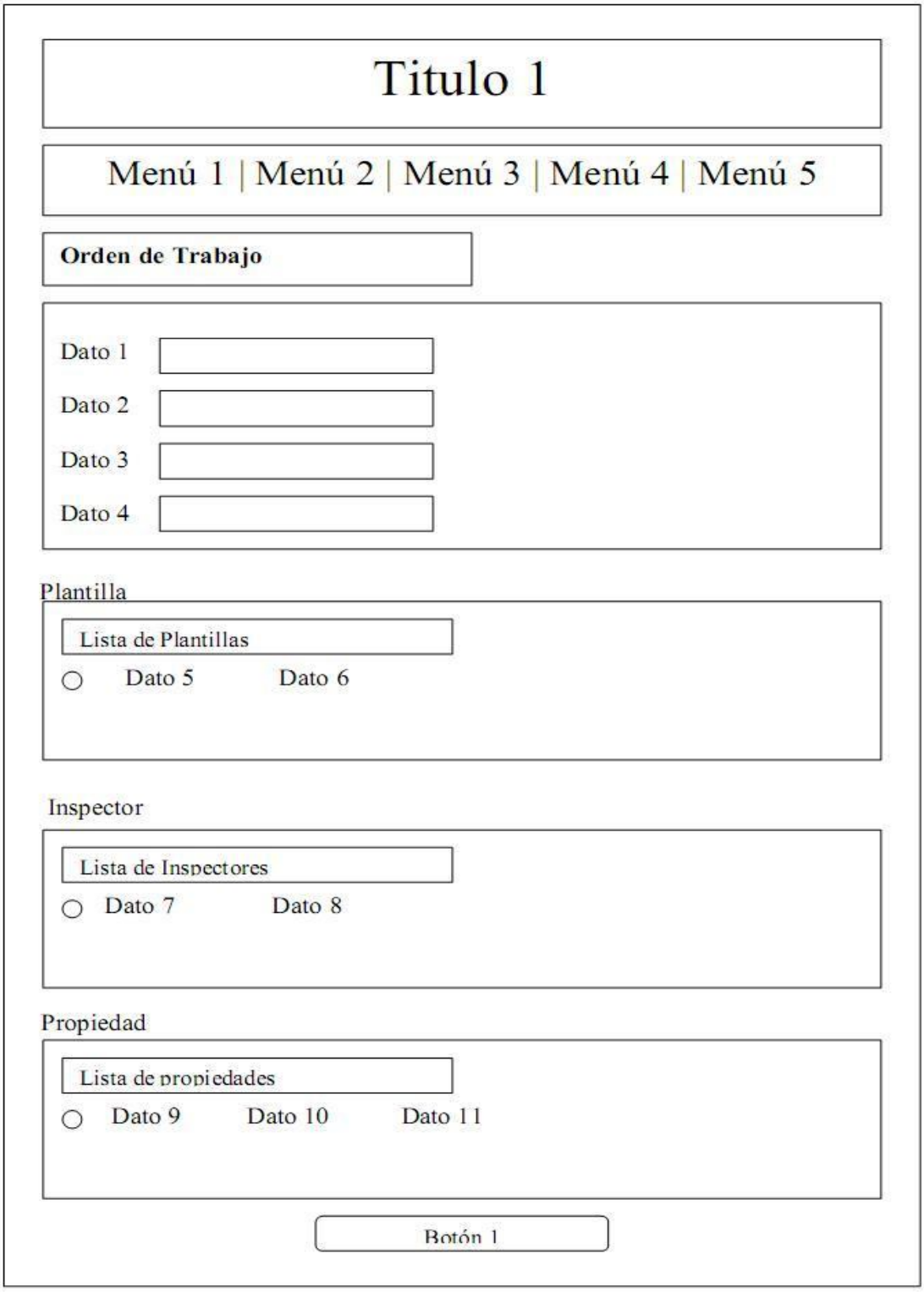


Figura 5.6: Diseño Conceptual Pantalla Gestión de OT

Descripción: En esta pantalla el Administrador podrá generar/Editar una Orden de Trabajo.

Menú	Función
1	Usuarios
2	Clientes.
3	Orden de Trabajo.
4	Plantillas.
6	Salida.

Dato	Función
1	Factor de Honorarios.
2	Sub Total.
3	IVA
4	Total.
5	Id de Plantilla
6	Nombre de Plantilla.
7	Id Inspector
8	Nombre Inspector.
9	Id Propiedad
10	Dirección Propiedad.
11	Dueño Propiedad.

Botón	Función
1	Modificar Plantilla

Figura 5.7 Pantalla de Lista de Orden de Trabajo:

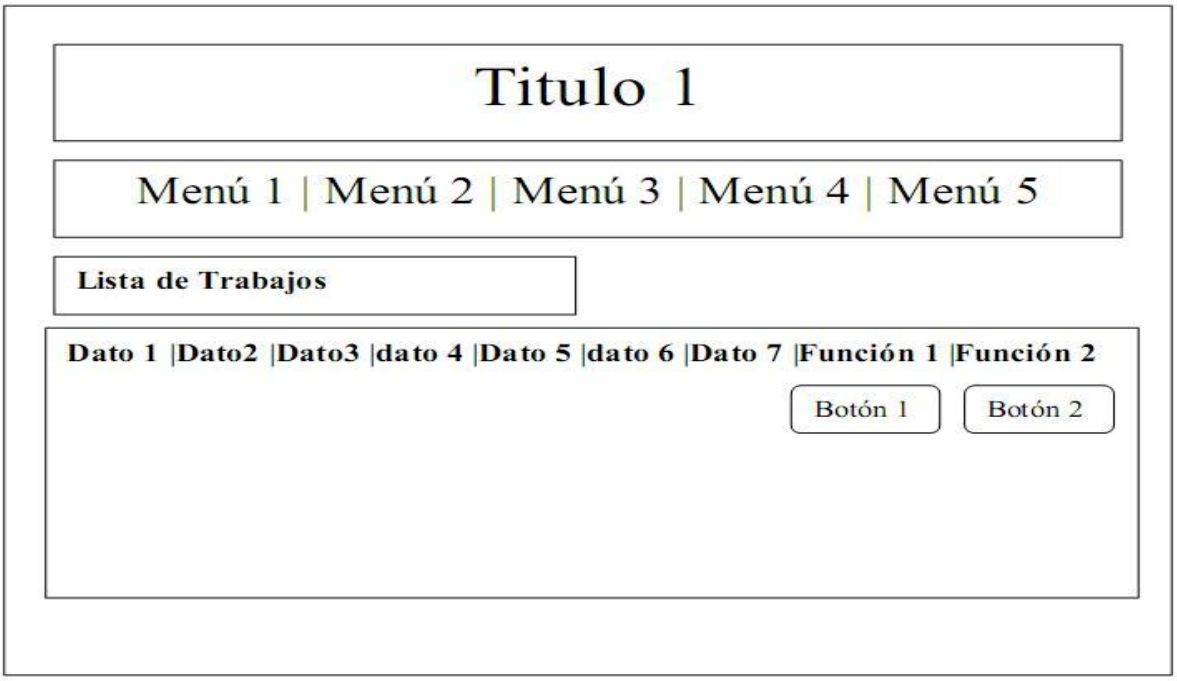


Figura 5.7: Diseño Conceptual Pantalla Lista de OT

Descripción: En esta sección se podrán listar las diferentes ordenes de trabajo, el administrador además podrá seleccionar para su edición. El Inspector podrá generar el informe asociado a la orden de trabajo, el cual será emitido en formato rtf.

Menú	Función
1	Usuarios
2	Clientes.
3	Orden de Trabajo.
4	Plantillas.
6	Salida.

Dato	Función
1	Id de OT.
2	Fecha Visita.
3	Duración.
4	Propiedad.
5	Inspector.
6	Administrador.

Botón	Función
1	Editar Reporte
2	Generar reporte

3	Editar OT

Figura 5.8 Pantalla de Administración de Plantillas (general para cada ítem de plantilla):

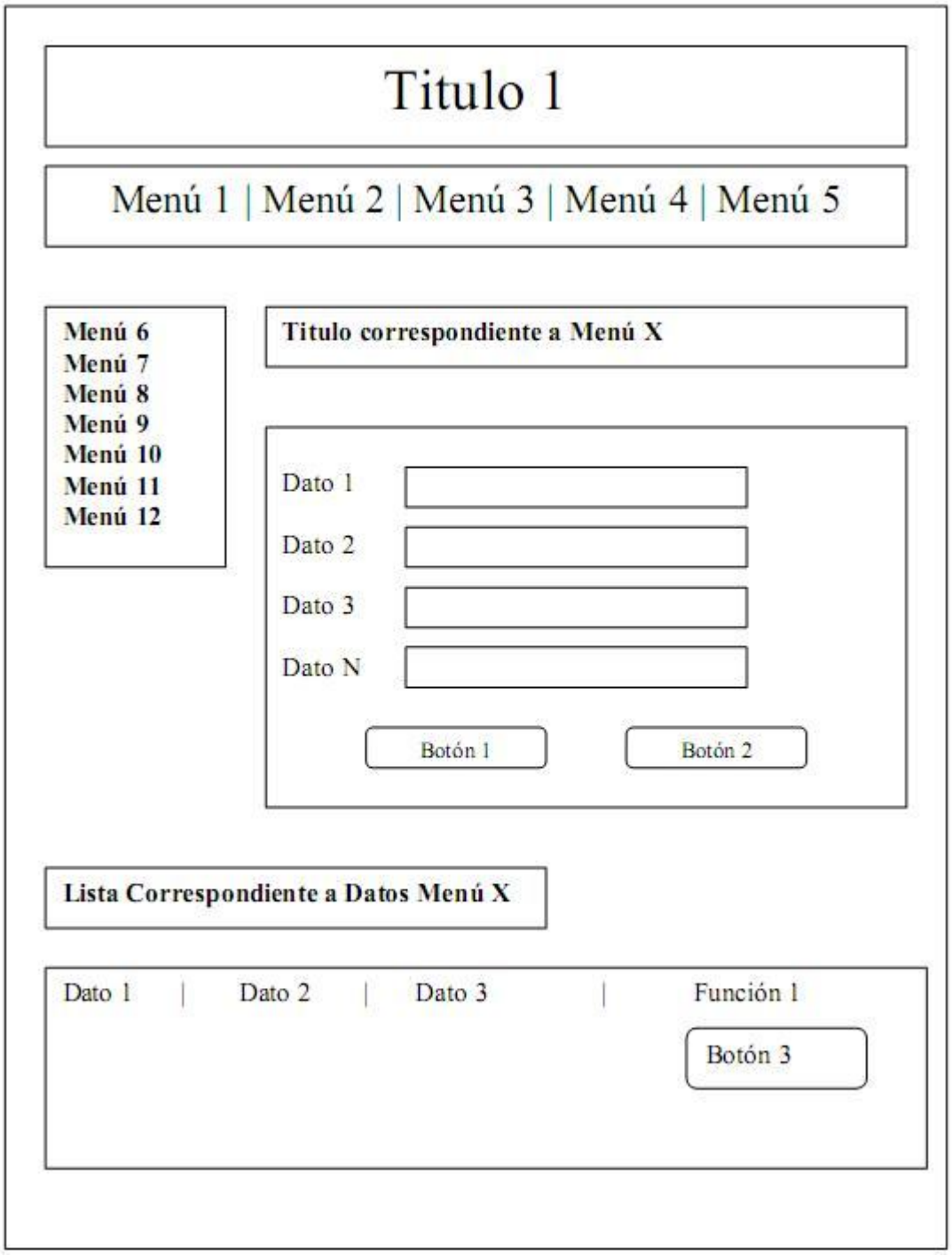


Figura 5.8: Diseño Conceptual Pantalla Administración de Plantillas.

Descripción: Esta pantalla es general para la edición de plantilla y cada uno de sus ítems, así mismo el administrador podrá realizar la estructuración de una plantilla previamente creada.

Dato	Función
1	Usuarios
2	Clientes.
3	Orden de Trabajo.
4	Plantillas.
5	Salida.
6	Estructura.
7	Plantilla
8	capitulo.
9	Sección.
10	ítem.
11	Sub ítem
12	Valores.

Datos 1, 2, 3...N: Datos correspondientes a cada ítem vistos desde el meú 6 al 9.

Botón	Función
1	Guardar Datos.
2	Limpiar Formulario.
3	Eliminar (según ítem Seleccionado).

Figura 5.9Pantalla para creación de Plantilla Dinámica (Estructuración):

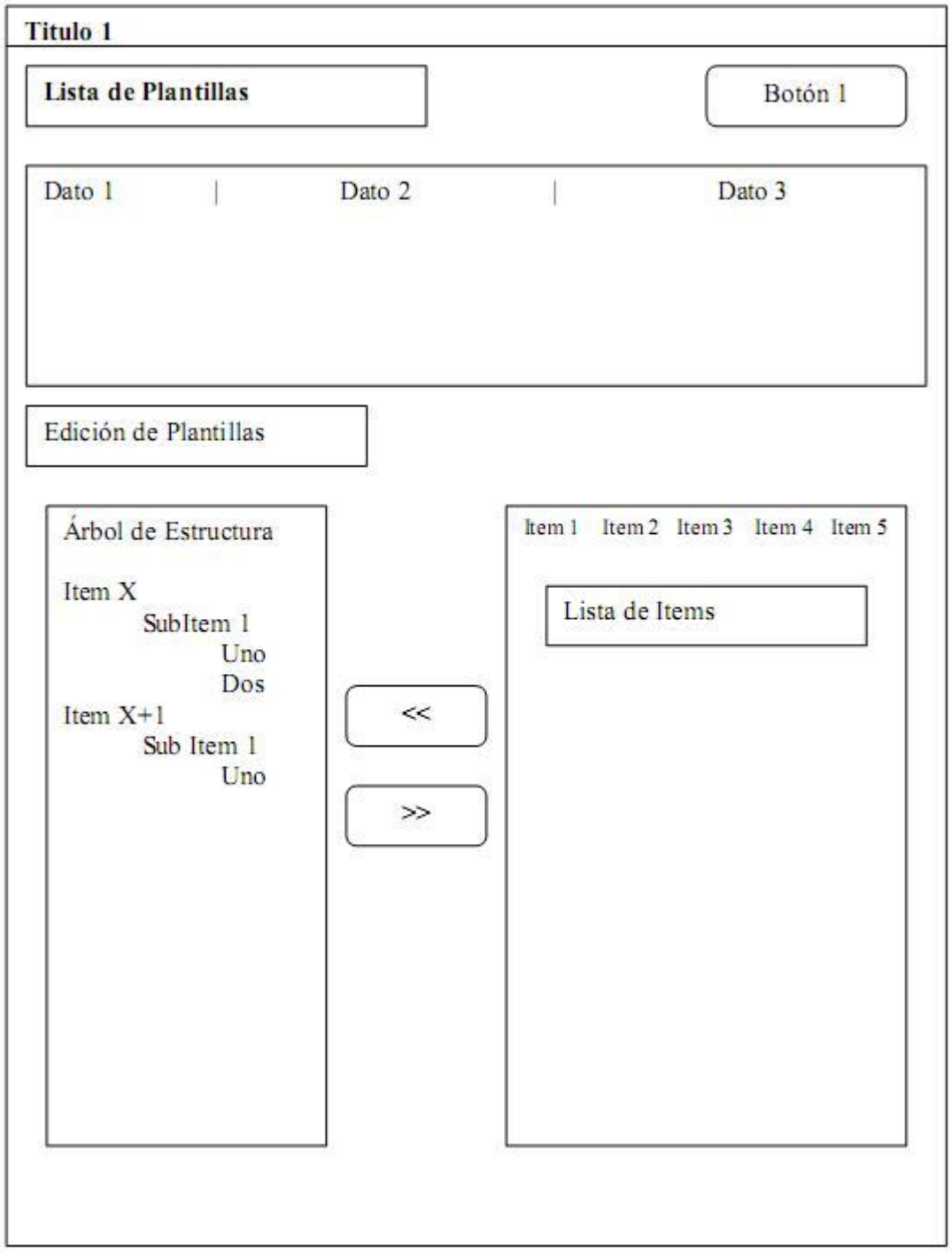


Figura 5.9: Diseño Conceptual de Pantalla Estructurar Plantilla

Descripción: En esta sección el Administrador o diseñador de plantillas podrá dar una estructuración específica a cada plantilla previamente creada, esto según Sección, capitulo, ítem, sub-ítem, y valores.

Dato	Función
1	Id de Plantilla
2	Nombre de Plantilla
3	Descripción de Plantilla.

Ítem	Descripción
1	Capitulo.
2	Sección.
3	Item.
4	Sub item.
5	Valor

- Entiéndase cada ítem como una pestaña distinta en el sistema.

Botón	Función
<<	Saca valores de la lista y los agrega a plantilla.
>>	Saca valores de la plantilla.
1	Carga una plantilla en el árbol de estructura.
2	Guarda la estructura

5.2.2 Diseño Físico de Interfaces.

En esta sección se muestra el aspecto real del software, indica las pantallas con las que el usuario interactuará con el sistema.

Pantalla de Ingreso a Home Inspector:

Figura 5.10: Pantalla de acceso al sistema.

Pantalla de Gestión de Usuarios:

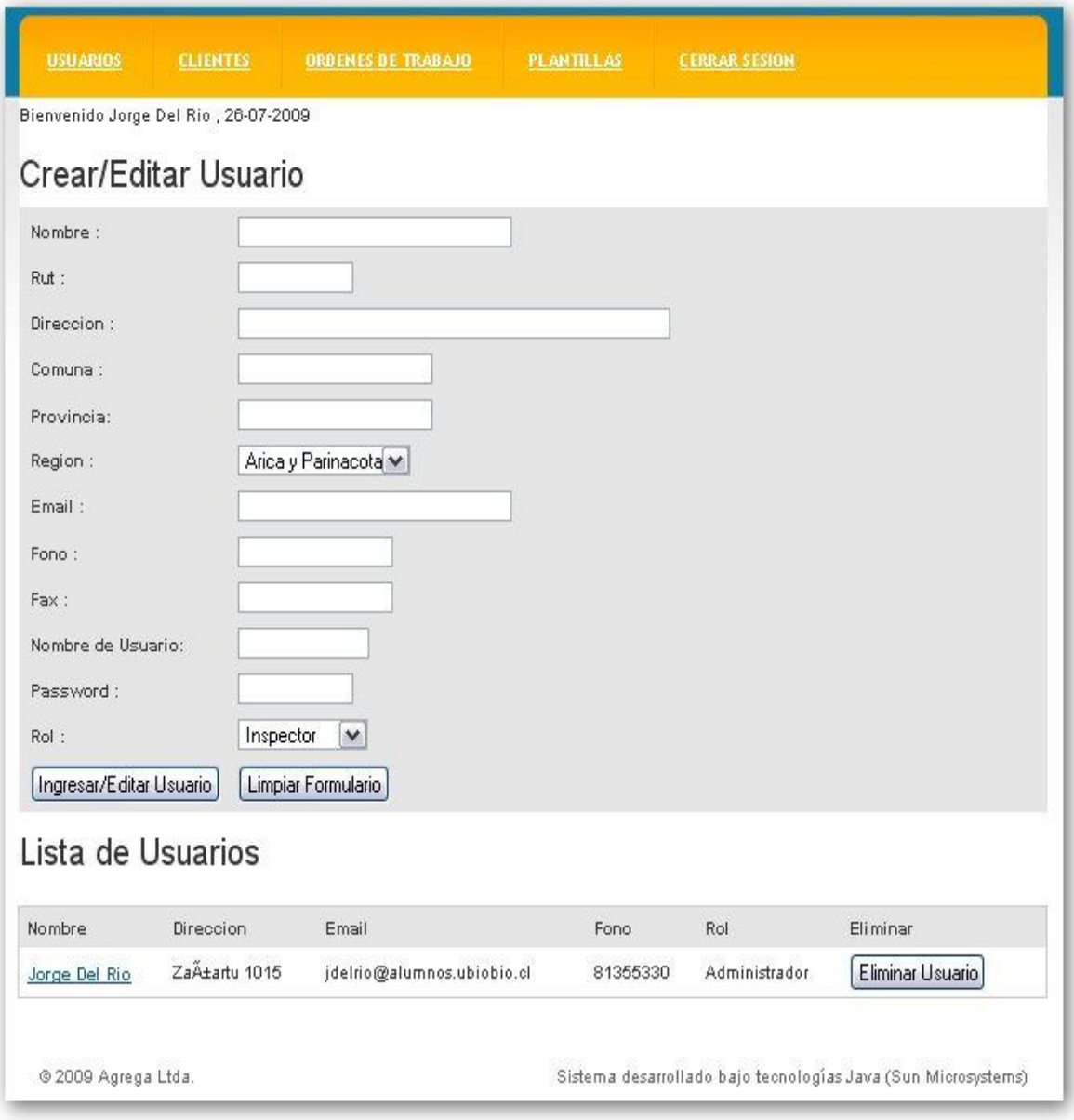


Figura 5.11: Pantalla de Gestión de Usuarios

Pantalla de Gestor de Clientes:

Figura 5.12: Pantalla de gestión de Clientes

Pantalla Módulo órdenes de Trabajo:

Figura 5.13: Pantalla de Listas de OT

Pantalla de creación de Órdenes de Trabajo

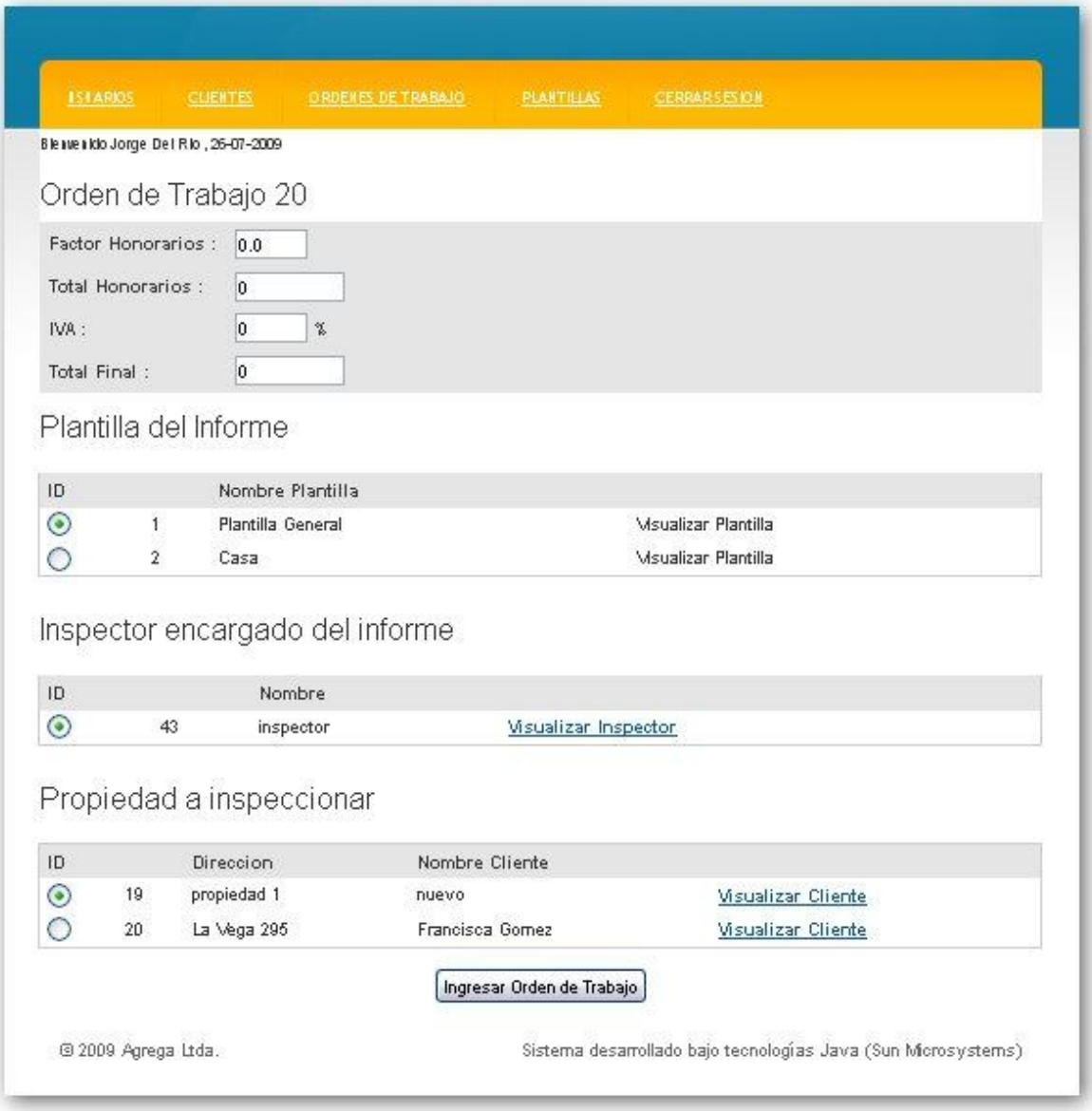


Figura 5.14: Pantalla Creación OT

Pantallas de Módulos de Plantillas:

Gestor de Plantillas (Plantilla, Capitulo, sección, ítem, sub ítem, valor)

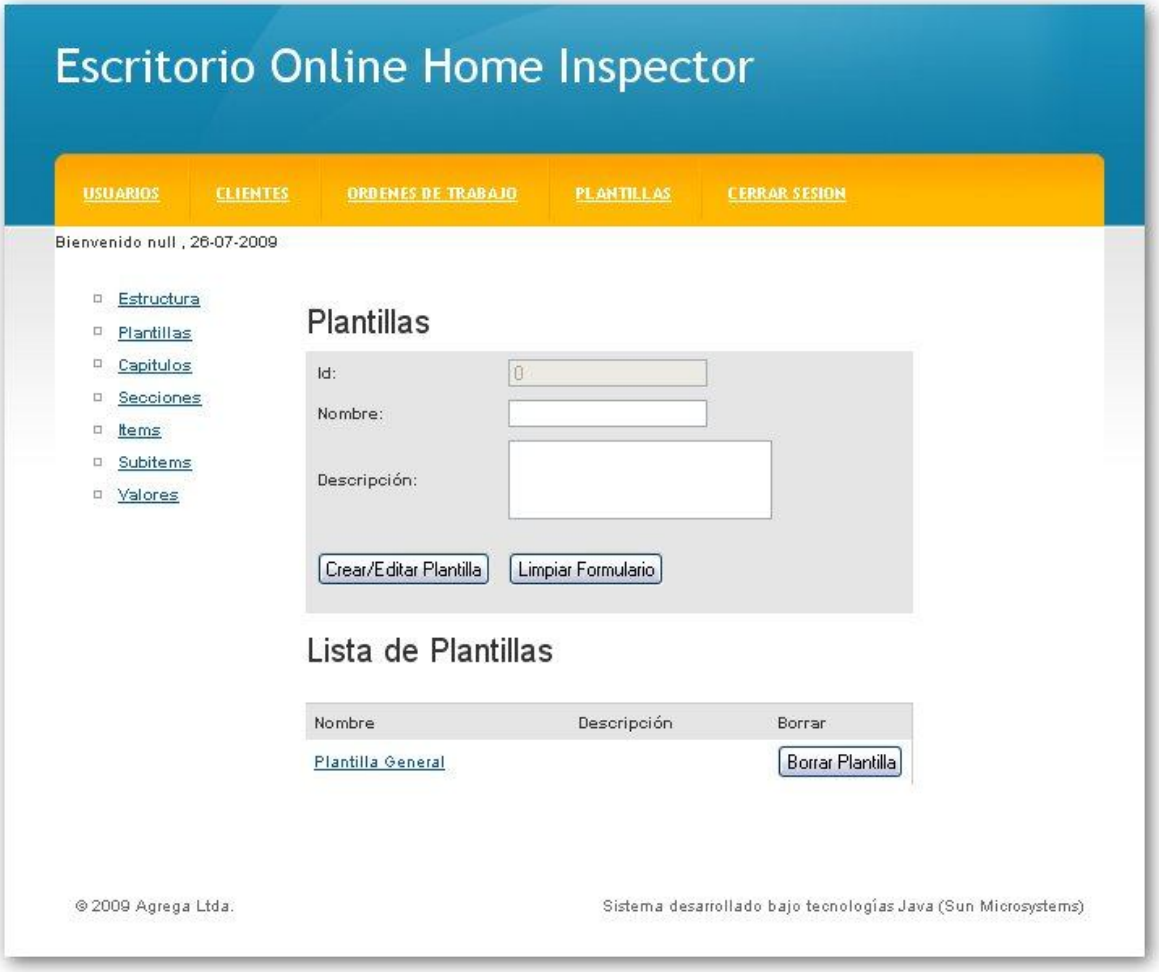


Figura 5.15: Gestión de Plantillas

Figura 5.16: Pantalla Gestión de Capítulos.

Figura 5.17: Pantalla de Gestión de Secciones

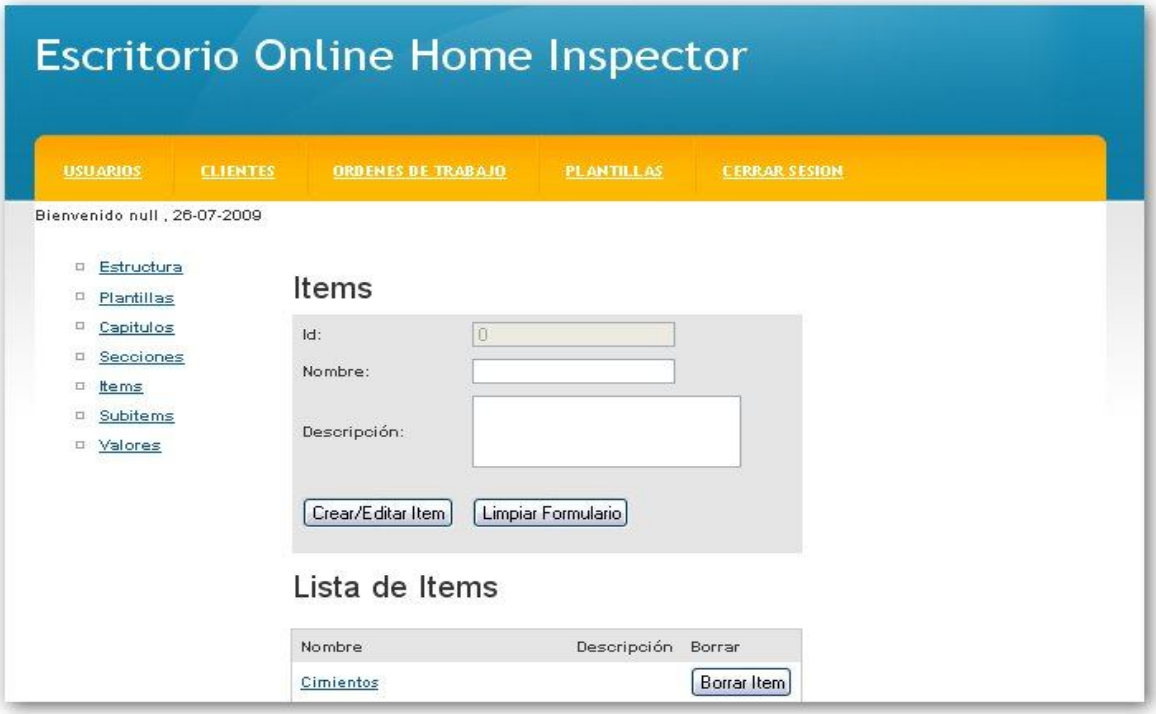


Figura 5.18: Pantalla de Gestión de ítems

Figura 5.19: Pantalla de Gestión de Sub- ítems

Figura 5.20: Pantalla de Gestión de Valores.

Pantallas para el llenado de formulario de Reportes (Figuras 5.21, 5.22, 5.23, 5.24)

Figura 5.21: Llenado de Inspección

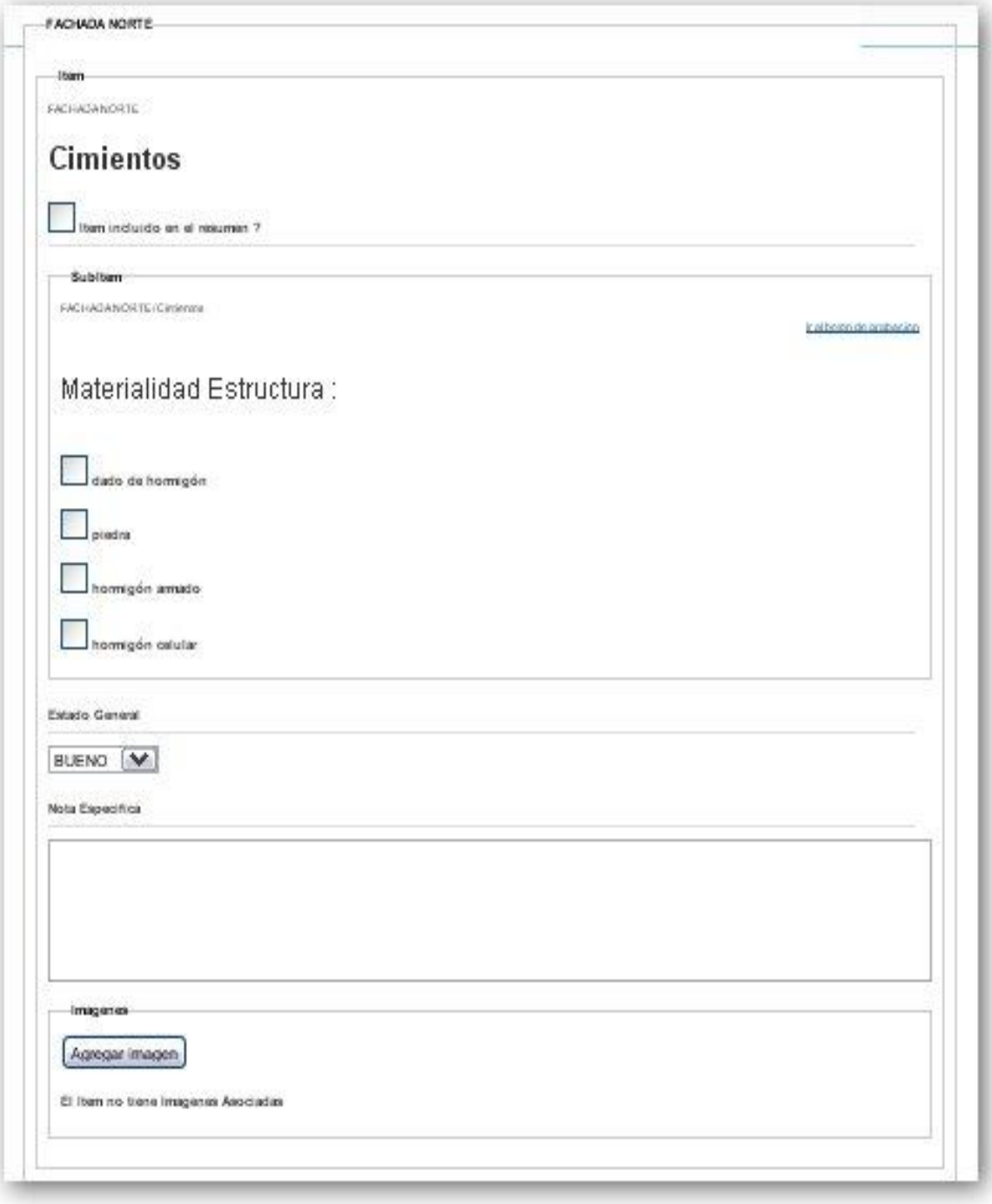


Figura 5.22: Llenado de Inspección

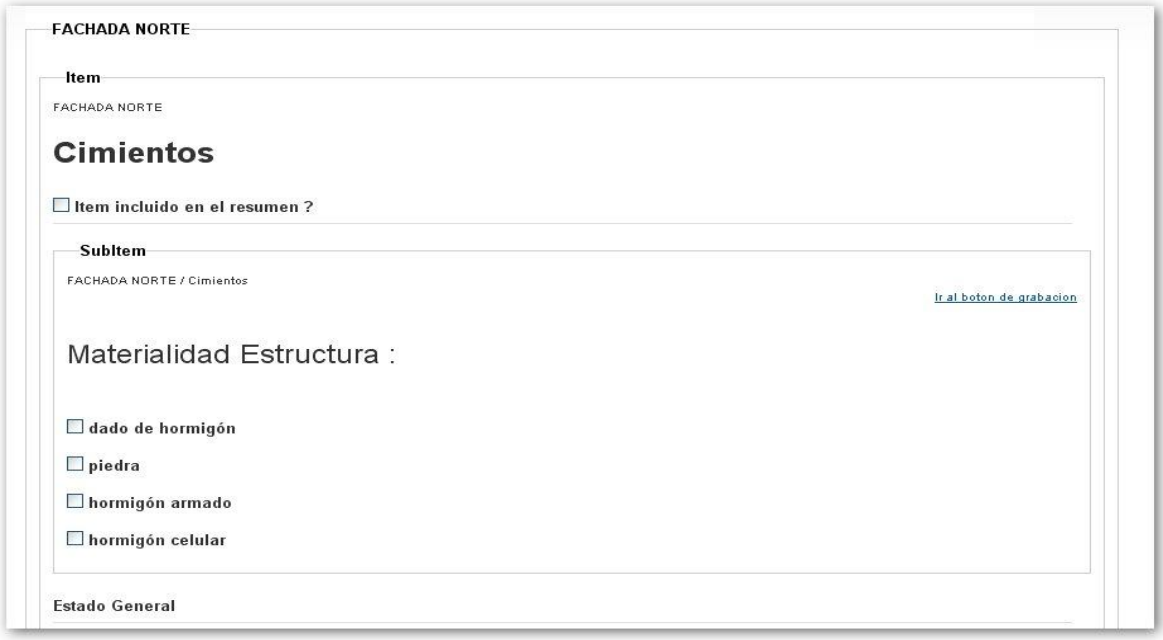


Figura 5.23: Llenado de Inspección

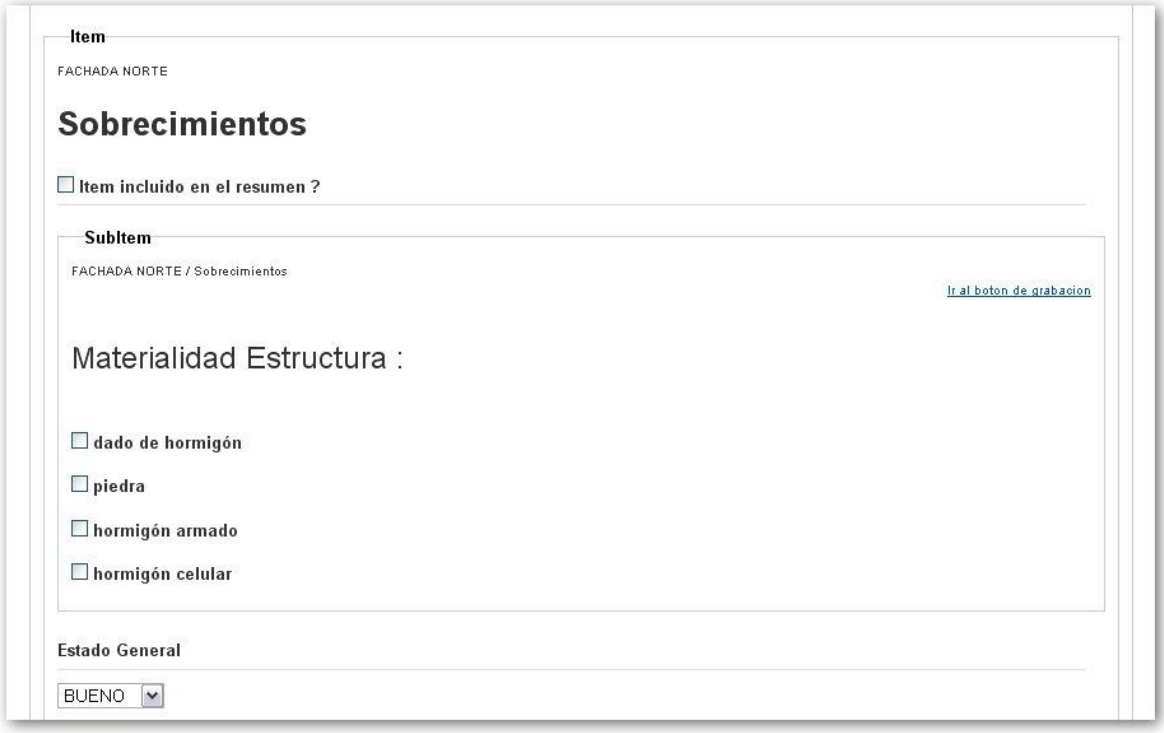


Figura 5.24: Llenado de Inspección

Cierre de Sesión

Figura 5.25: Pantalla de Cierre de Sesión

Capitulo 6

Pruebas de Software e Implementación

Introducción al Capitulo

En este capitulo se dan a conocer las pruebas de software aplicadas al sistema, estas pruebas tienen la finalidad de encontrar fallas en las respuestas del sistema, con la finalidad de determinar o verificar la calidad del producto software. Las pruebas de software se clasifican en validación y verificación, la primera consiste en verificar que el sistema cumple con los requisitos especificados en la etapa de diseño, y la segunda si el sistema cumple con lo esperado en funcionalidad de forma correcta.

6.1.- Enfoque de pruebas

Los dos enfoques de pruebas más utilizados son los denominados "caja blanca" y "caja negra".

El enfoque de "caja blanca" o también llamadas estructurales, tiene la particularidad de que en todo momento se está observando el código fuente de la aplicación, analizando bucles, decisiones, etc., siempre centrado en el área procedural del sistema.

El enfoque de "caja negra" se centra en probar que el sistema cumpla con los requisitos, tenga correcta navegabilidad y visualización, los datos de entrada estén correctamente validados y que las respuestas del sistema sean las esperadas. Esto se hace siempre a través de la interfaz del sistema, sin recurrir a revisión de código.

Para este sistema se determina ejecutar pruebas de caja negra, las cuales son definidas a continuación.

6.2.- Definición de pruebas

Para las pruebas a realizar en este sistema se han considerado las siguientes áreas:

- *Pruebas de Visualización*: Se revisará que los links entre páginas del sistema estén correctos y que la forma de mostrar los formularios y listas sea la adecuada.

- *Pruebas de Unidad*: Se verificará cada módulo de forma independiente, considerando que los valores de ingreso y salidas esperadas sean los correctos esto es formularios, campos obligatorios y rangos de validaciones.

- *Pruebas de Carga*: Se revisará que el sistema sea capaz de soportar un número adecuado de conexiones a la vez y mantenga sus tiempos de respuesta aceptables, para este caso se determina un tiempo máximo de 20 segundos en carga de páginas y respuestas desde el servidor. El número de conexiones será de 10 en forma simultánea.

- *Pruebas de Seguridad*: En estas pruebas se verificarán que las sesiones funcionen correctamente. Dando a cada usuario sus opciones permitidas y verificando que no se pueda acceder si no se ha iniciado la correspondiente sesión de trabajo.

6.3.- Plan de Pruebas

A continuación se presenta un esquema de las pruebas, tiempos de duración y personal responsable de llevarlas a cabo.

Pruebas	Duración Aproximada	Responsable
Visualización	1 hora	Desarrollador – Agrega Ltda.
Unidad	3-4 Hrs.	Desarrollador – Agrega Ltda.
Carga	1 hora	Desarrollador – Agrega Ltda.
Seguridad	1 hora	Desarrollador

6.4.- Datos de Prueba

En las tablas presentadas a continuación se observan las pruebas que serán realizadas, la salida esperada y el plan de solución en caso de existir errores.

Entrada	Conexión al sistema
Salida esperada	Página de inicio de sesión del sistema
Plan de Solución	Revisar servidor

Entrada	Ingreso al sistema
Salida esperada	Página de Inicio del sistema
Plan de Solución	Revisar destino de entrada

Entrada	Ingreso a la gestión de usuarios
Salida esperada	Pantalla de Gestión de Usuarios
Plan de Solución	Revisar links

Entrada	Ingreso a la gestión de clientes
Salida esperada	Pantalla de Gestión de clientes
Plan de Solución	Revisar links

Entrada	Ingreso a la gestión de Ordenes de Trabajo
Salida esperada	Pantalla de Gestión de Ordenes de Trabajo
Plan de Solución	Revisar links

Entrada	Ingreso a la gestión de Plantillas
Salida esperada	Pantalla de Gestión de plantillas
Plan de Solución	Revisar links

Entrada	Ingreso de Datos en formularios
Salida esperada	Que los datos sean grabados en la base de datos
Plan de Solución	Verificar Base de Datos

Entrada	Modificación de datos en los distintos formularios
Salida esperada	Datos modificados en la base de datos
Plan de Solución	Verificar Base de Datos

Entrada	Páginas de Mantención
Salida esperada	Listas de Ítems con los datos correctos
Plan de Solución	Verificar llamadas a Base de datos

Entrada	Selección de Ítems
Salida esperada	Despliegue de datos de ítem seleccionado en formularios
Plan de Solución	Verificar llamada a funciones

Entrada	Completar/Editar formulario de reporte
Salida esperada	Despliegue de formulario según plantilla y estructura
Plan de Solución	Verificar funciones Objeto/XML

Entrada	Estructurar Informe
Salida esperada	Descarga de sub programa para estructuración de informe
Plan de Solución	Verificar requerimientos técnicos necesarios

Entrada	Diseño de Estructura
Salida esperada	Estructura de Informe según lo especificado
Plan de Solución	Verificar Aplicación

Entrada	Impresión de Informe
Salida esperada	Impresión de Informe en formato RTF según Plantilla
Plan de Solución	Verificar funcionalidad

Entrada	Conexión múltiple de usuarios
Salida esperada	Usuarios conectados
Plan de Solución	Verificar funcionalidad del servidor

Entrada	Verificar sesiones de usuario
Salida esperada	Sesiones correspondientes al rol de usuario
Plan de Solución	Revisar manejo de sesiones

Entrada	Cierre de sesión
Salida esperada	Cierre del sistema, redirección a página de cierre
Plan de Solución	Verificar cierre de sesión.

Objetivo de Pruebas.

La realización de estas pruebas permitirá que el sistema final que se pondrá a disposición del cliente sea óptimo y no se produzcan errores posteriores en tiempos de ejecución.

6.5.- Plan de Implantación

Para este sistema se realizará una llamada implementación inmediata, ya que no hay software precedente a este sistema en la empresa.

Se cargará el sistema Web en el host destinado por Agrega Ltda. y se informará a la empresa Home Inspector el comienzo del funcionamiento del sistema.

Para esto se dará un tiempo de marcha blanca de un mes, en donde el cliente podrá hacer uso del sistema y verificar el buen funcionamiento y utilidad de éste.

La puesta en marcha estará a cargo de la empresa Agrega Ltda.

Conclusiones

Este proyecto tuvo como propósito el entregar una herramienta innovadora para una empresa innovadora y lograr que este nuevo mercado sea aún más interesante.

A través de la realización de este proyecto se ha logrado obtener una herramienta que hace más rápido y más liviano el trabajo realizado actualmente por un hombre o un grupo de hombres, por lo que se ha logrado utilizar la finalidad de la informática y de las máquinas de proceso, mejorar el estilo de vida del ser humano.

Con este sistema, la empresa adquiere una mejora sustancial en el proceso de inspección, mejorando tiempo, calidad, precisión de la información y una ventaja muy importante, orden en los datos ya que el uso de este sistema implica la disminución de muchos archivos inservibles, que con el tiempo se convertían en basura, además de ahora tener los datos claros y seguros, todos fáciles de ubicar en un mismo lugar, a través de la misma aplicación.

Como alumnos en proceso de obtención de titulo, el desarrollo de este proyecto ha sido de gran utilidad, ya que me he logrado acercar al mundo laboral real del área informática, desarrollando este sistema para una empresa del rubro y trabajando con gente experimentada y con mucha experiencia en el campo.

Lo mas complicado, indiscutiblemente, ha sido la adopción de nuevas tecnologías de desarrollo, desde el lenguaje hasta la metodología de trabajo, comprender todo un mundo nuevo dentro del área de la Ingeniería de Software. Pero he logrado comprender que estas nuevas tecnologías hacen del desarrollo de software una herramienta capaz de entregar soluciones útiles y que son muy fáciles de mantener a través del tiempo, dando gran énfasis a que los nuevos sistemas tengan un alto grado de desarrollo en el control de versiones.

Sin duda el beneficio personal obtenido en el desarrollo de este proyecto ha sido realmente bueno, pero aún falta mucho por descubrir y mucho por aprender, es por eso que nosotros, como Ingenieros de Software debemos estar siempre atentos a los nuevos adelantos, es la única forma que tenemos de llegar a ser los profesionales que nuestro país y el mundo necesitan.

Agradecimientos

Primero que todo agradezco a Dios, por darme la voluntad y la fuerza para llegar hasta el final de este proceso que comenzó hace ya varios años y que hoy lo veo realizado, a pesar de todas las dificultades en el camino. Gracias Dios.

Agradezco a mi madre, que a pesar de sus muy limitados recursos siempre ha estado luchando por darme una vida digna e inculcándome las buenas costumbres y valores para hacer de mi una mejor persona. Gracias por decirme una y otra vez que siempre debo buscar llegar más allá, gracias por tu esfuerzo y dedicación. Gracias Mamá. También agradezco a mi padre, quien también aportó para que yo llegara al final, ya no está conmigo, pero desde el cielo sé que está siempre velando por mí.

Agradezco a mi familia, a mi esposa, Francisca, quien desde el principio ha estado a mi lado, con su apoyo y sus palabras de aliento, y además me dió a mi pequeño hijo, Cristóbal, quien es una fuente de alegría y de motivación para seguir adelante.

Agradezco a dos grandes personas a quién descubrí en mi camino por la universidad, Jason Matamala y Gonzalo Tirapegui, quienes se convirtieron en dos grandes amigos y que además en todo momento me dieron su gran apoyo para lograr este proyecto, sin sus conocimientos, ayuda y enseñanzas, jamás lo hubiese logrado.

Por último agradezco a los profesores de la Facultad de Ciencias Empresariales, por entregarme las herramientas necesarias para llegar hasta aquí. Y muy especialmente a Manuel Crisosto Muñoz quien me apoyó en todo sentido desde el principio de este proyecto. Y a Patricio Gálvez, por su comprensión y apoyo.

Bibliografía

- JACOBSON, Ivar ., BOOCH, Grady., RUMBAUGH, James., El Proceso Unificado de Desarrollo de Software. Madrid, Addison-Wesley,2000.
- Apuntes de Asignatura, Metodología de Desarrollo de SIA.
- Apuntes de Asignaturas, Ingeniería de Software.
- Apuntes de Asignatura, Taller de Ingeniería de Software.
- Netbeans Documentación: www.netbeans.org [consulta: 26 agosto 2009]
- Documentación UML: www.uml.org [consulta: 26 agosto 2009]
- Documentación Oracle TopLink <http://www.oracle.com/technology/products/ias/toplink/index.html> [consulta: 26 agosto 2009]
- Artículos técnicos Dr. Winston Prackash <http://www.winstonprakash.com/articles/articles.html> [consulta: 26 agosto 2009]

  MySQL 5.0 Reference Manual <http://dev.mysql.com/doc/refman/5.0/es/index.html > [consulta: 26 agosto 2009]

Anexos

Anexo Nº 1

Metodologías Utilizadas

Rational Unified Process (RUP)

Historia

La Figura 1 ilustra la historia de RUP. El antecedente más importante se ubica en 1967 con la Metodología Ericsson (*Ericsson Approach*) elaborada por Ivar Jacobson, una aproximación de desarrollo basada en componentes, que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la compañía *Objectory* AB y lanza el proceso de desarrollo *Objectory* (abreviación de *Object Factory*).

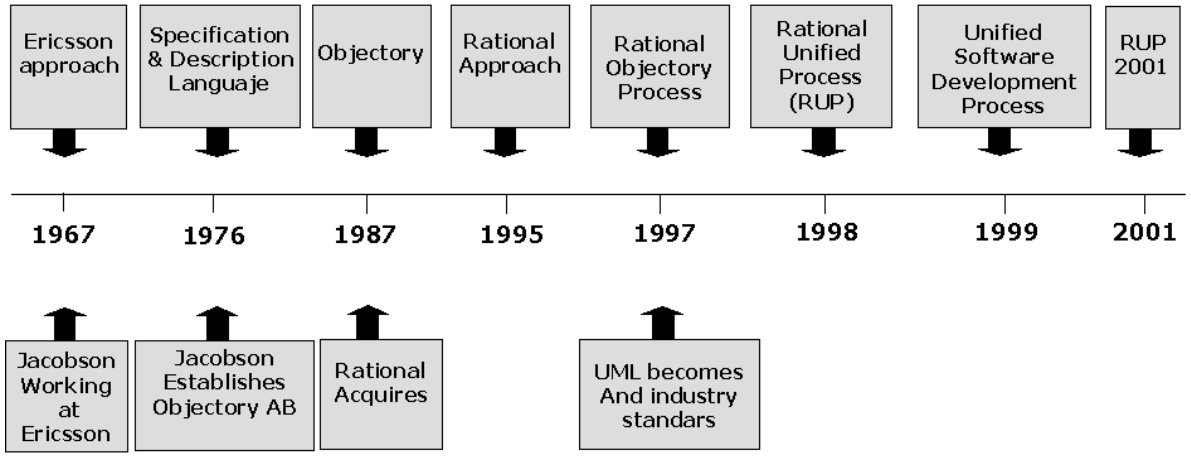


Figura 1: Historia de RUP

Posteriormente en 1995 *Rational Software Corporation* adquiere *Objectory AB* y entre 1995 y 1997 se desarrolla *Rational Objectory Process* (ROP) a partir de *Objectory* 3.8 y del Enfoque Rational (*Rational Approach*) adoptando UML como lenguaje de modelado.

Desde ese entonces y a la cabeza de Grady Booch, Ivar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir ROP, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio del 1998 se lanza *Rational Unified Process*.

1.- Características esenciales

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por los Casos de Uso, está centrado en la arquitectura, y es iterativo e incremental.

Proceso dirigido por Casos de Uso

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que seria bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo como se muestra en la Figura 2.

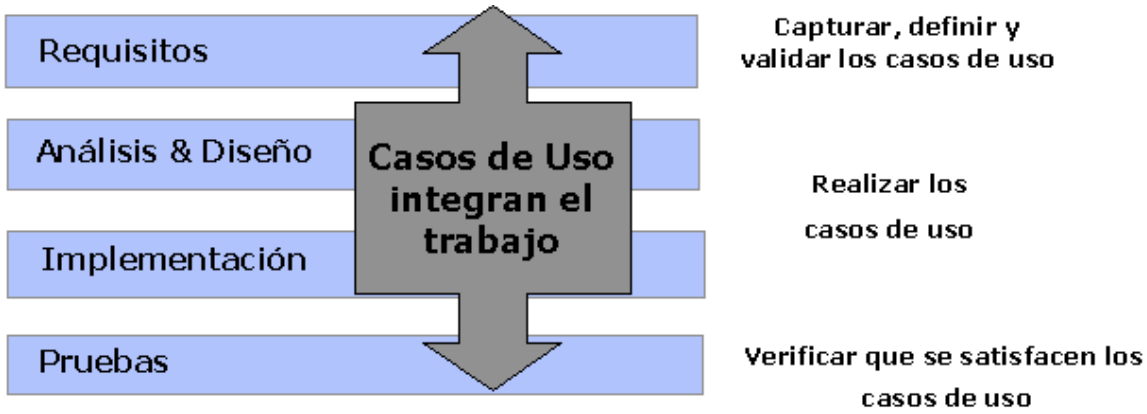


Figura 2: Los Casos de Uso integran el trabajo

Los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Como se muestra en la Figura 3, basándose en los Casos de Uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada Caso de Uso. Todos los modelos deben estar sincronizados con el modelo de Casos de Uso.

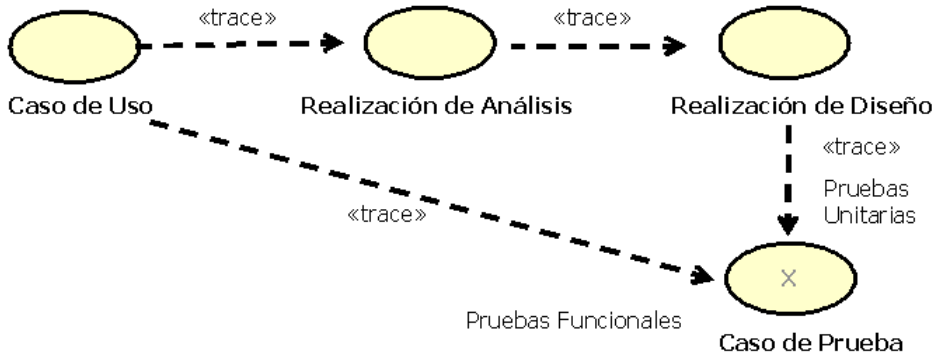


Figura 3: Trazabilidad a partir de los Casos de Uso

Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Al final de la fase de elaboración se obtiene una *línea base* de la arquitectura donde fueron seleccionados una serie de Casos de Uso arquitectónicamente relevantes (aquellos que ayudan a mitigar los riesgos más importantes, aquellos que son los más importantes para el usuario y aquellos que cubran las funcionalidades significativas)

Proceso iterativo e incremental

El equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la Figura 4. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

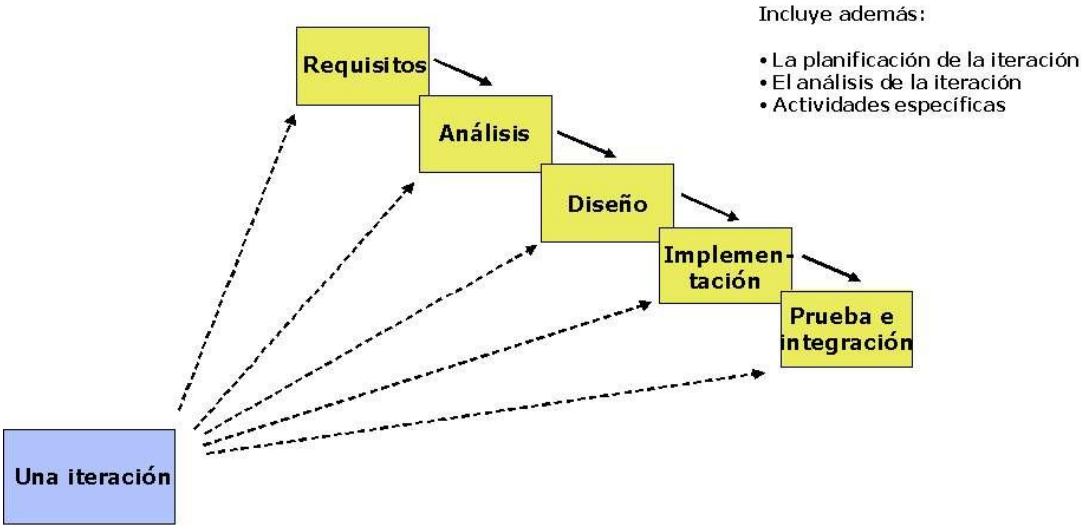


Figura 4: Una iteración RUP

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. En la Figura 5 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

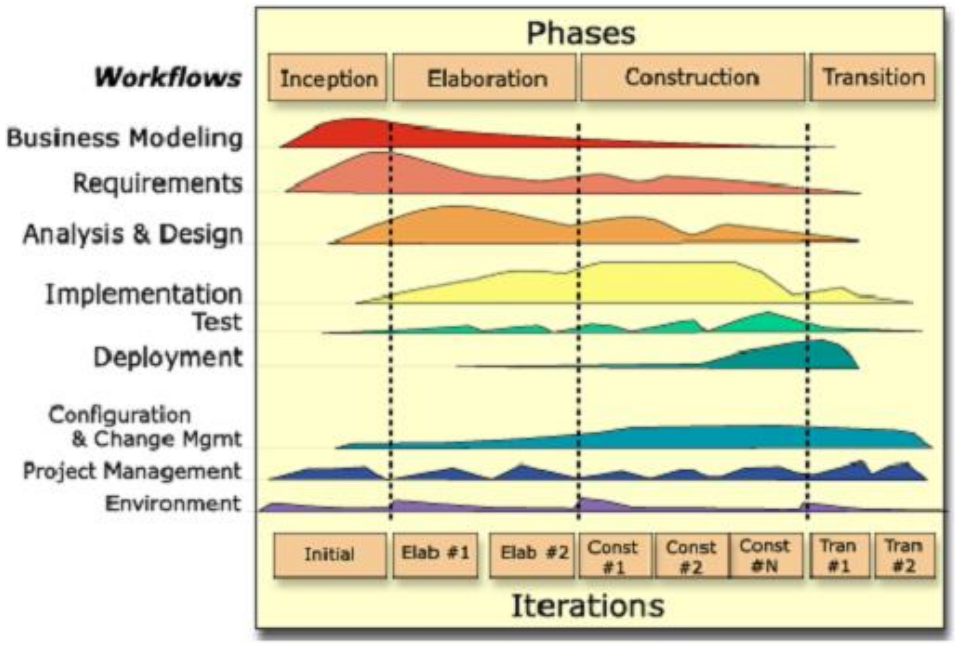


Figura 5: Esfuerzo en actividades según fase del proyecto

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una línea base de la arquitectura.

Durante la fase de inicio las iteraciones hacen ponen mayor énfasis en actividades modelado del negocio y de requisitos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la *línea base* de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

1.- Estructura del proceso

El proceso puede ser descrito en dos dimensiones o ejes:

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Figura 8 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Como se mencionó anteriormente cada fase se subdivide a la vez en iteraciones.

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

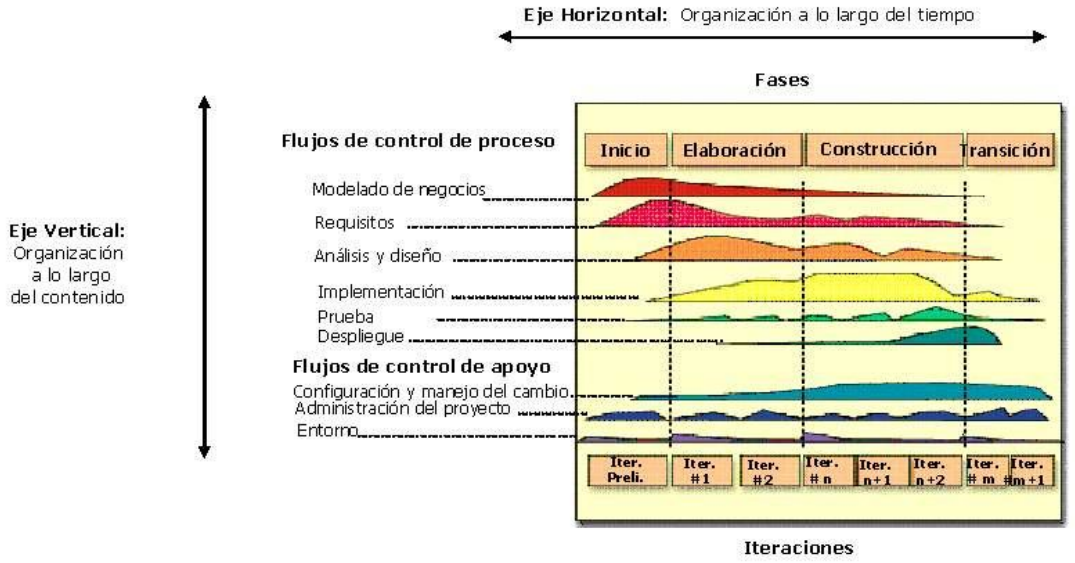


Figura 6: Estructura de RUP

Estructura Dinámica del proceso. Fases e iteraciones

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo concluye con una generación del producto para los clientes. Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a la vez en iteraciones, el número de iteraciones en cada fase es variable.

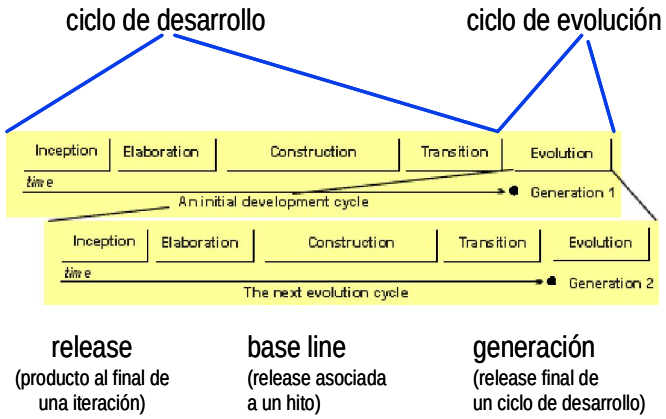


Figura 7: Ciclos, releases, baseline

Cada fase se concluye con un hito bien definido, un punto en el tiempo en el cual se deben tomar ciertas decisiones críticas y alcanzar las metas clave antes de pasar a la siguiente fase, ese hito principal de cada fase se compone de hitos menores que podrían ser los criterios aplicables a cada iteración. Los hitos para cada una de las fases son: Inicio - *Lifecycle Objectives,* Elaboración - *Lifecycle Architecture,* Construcción - *Initial Operational Capability,* Transición - Product Release.
Las fases y sus respectivos hitos se ilustran en la Figura 8.

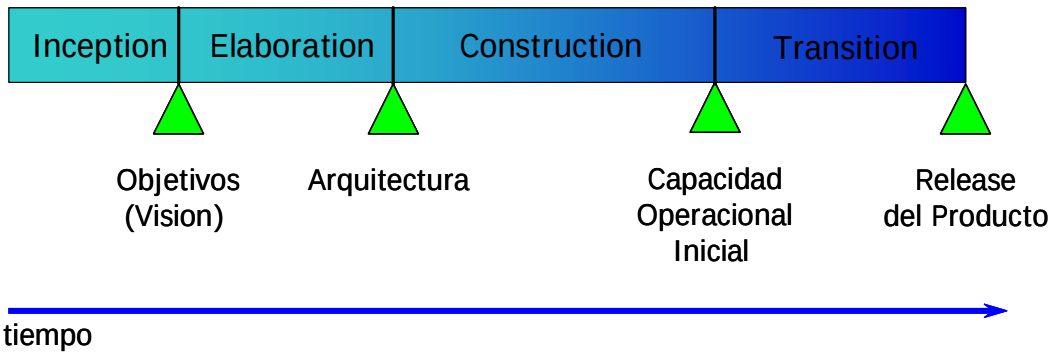


Figura 8: Fases e hitos en RUP

Inicio

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados al proyecto.

Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad

- Mostrar al menos una arquitectura candidata para los escenarios principales
- Estimar el coste en recursos y tiempo de todo el proyecto
- Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser:

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales
- Modelo inicial de Casos de Uso (10-20% completado)
- Un glosario inicial: Terminología clave del dominio
- El caso de negocio.
- Lista de riesgos y plan de contingencia
- Plan del proyecto, mostrando fases e iteraciones
- Modelo de negocio, si es necesario
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales
- Las estimaciones de tiempo, coste y riesgo son creíbles
- Comprensión total de cualquier prototipo de la arquitectura desarrollado
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Los objetivos de esta fase son :

- Definir, validar y cimentar la arquitectura
- Completar la visión
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes resultados:

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico
- Descripción de la arquitectura software
- Un prototipo ejecutable de la arquitectura
- Lista de riesgos y caso de negocio revisados
- Plan de desarrollo para el proyecto
- Un caso de desarrollo actualizado que especifica el proceso a seguir
- Un manual de usuario preliminar (opcional).

En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable
- La arquitectura es estable
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

Construcción

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Los objetivos concretos incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo
- Conseguir una calidad adecuada tan rápido como sea práctico
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los resultados de la fase de construcción deben ser:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado.

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios
- Son aceptables los gastos actuales versus los gastos planeados.

Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Se citan algunas de las cosas que puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto
- Conversión de las bases de datos operacionales
- Entrenamiento de los usuarios y técnicos de mantenimiento
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por si mismo
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los resultados de la fase de transición son:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho
- Son aceptables los gastos actuales versus los gastos planificados.

Roles

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos .

RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos son:

Analistas:

- Analista de procesos de negocio
- Diseñador del negocio
- Analista de sistema
- Especificador de requisitos.

Desarrolladores:

- Arquitecto de software
- Diseñador
- Diseñador de interfaz de usuario
- Diseñador de cápsulas
- Diseñador de base de datos
- Implementador
- Integrador.

Gestores:

- Jefe de proyecto
- Jefe de control de cambios
- Jefe de configuración
- Jefe de pruebas
- Jefe de despliegue
- Ingeniero de procesos
- Revisor de gestión del proyecto
- Gestor de pruebas.

Apoyo:

- Documentador técnico
- Administrador de sistema
- Especialista en herramientas
- Desarrollador de cursos
- Artista gráfico.

Especialista en pruebas:

- Especialista en Pruebas (*tester*)
- Analista de pruebas
- Diseñador de pruebas.

Otros roles:

- *Stakeholders*
- Revisor
- Coordinación de revisiones
- Revisor técnico
- Cualquier rol.

Actividades

Una actividad en concreto es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

Artefactos

Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final.

Un artefacto puede ser cualquiera de los siguientes:

- Un documento, como el documento de la arquitectura del software
- Un modelo, como el modelo de Casos de Uso o el modelo de diseño
- Un elemento del modelo, un elemento que pertenece a un modelo como una clase, un Caso de Uso o un subsistema.

Flujos de trabajo

Con la enumeración de roles, actividades y artefactos no se define un proceso, necesitamos contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables. A continuación se dará una explicación de cada flujo de trabajo.

Modelado del negocio

Con este flujo de trabajo pretendemos llegar a un mejor entendimiento de la organización donde se va a implantar el producto.

Los objetivos del modelado de negocio son:

- Entender la estructura y la dinámica de la organización para la cual el sistema va ser desarrollado (organización objetivo)
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo
- Derivar los requisitos del sistema necesarios para apoyar a la organización objetivo.

Para lograr estos objetivos, el modelo de negocio describe como desarrollar una visión de la nueva organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario a estos modelos, se desarrollan otras especificaciones tales como un Glosario.

Requisitos

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Los objetivos del flujo de datos Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros *stakeholders* sobre lo que el sistema podría hacer
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema
- Definir el ámbito del sistema
- Proveer una base para la planeación de los contenidos técnicos de las iteraciones
- Proveer una base para estimar costos y tiempo de desarrollo del sistema
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Los requisitos se dividen en dos grupos. Los requisitos funcionales representan la funcionalidad del sistema. Se modelan mediante diagramas de Casos de Uso. Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final.

Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema.

Los objetivos del análisis y diseño son:

- Transformar los requisitos al diseño del futuro sistema
- Desarrollar una arquitectura para el sistema
- Adaptar el diseño para que sea consistente con el entorno de implementación, diseñando para el rendimiento.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos.

El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

Otro producto importante de este flujo es la documentación de la arquitectura de software, que captura varias vistas arquitectónicas del sistema.

Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración
- Cada implementador decide en que orden implementa los elementos del subsistema
- Si encuentra errores de diseño, los notifica
- Se prueban los subsistemas individualmente
- Se integra el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son:

- Encontrar y documentar defectos en la calidad del software
- Generalmente asesora sobre la calidad del software percibida
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas
- Verificar las funciones del producto de software según lo diseñado
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida nos sirva para ir refinando el producto a desarrollar.

Despliegue

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final
- Empaquetar el software para su distribución
- Distribuir el software
- Instalar el software
- Proveer asistencia y ayuda a los usuarios
- Formar a los usuarios y al cuerpo de ventas
- Migrar el software existente o convertir bases de datos.

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

Gestión del proyecto

La Gestión del proyecto es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

Los objetivos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos
- Proveer guías prácticas realizar planeación, contratar personal, ejecutar y monitorear el proyecto
- Proveer un marco de trabajo para gestionar riesgos.

La planeación de un proyecto posee dos niveles de abstracción: un plan para las fases y un plan para cada iteración.

Configuración y control de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

Entorno

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

El principal artefacto que se usa en este flujo de trabajo es el *caso de desarrollo* que especifica para el proyecto actual en concreto, como se aplicará el proceso, que productos se van a utilizar y como van a ser utilizados. Además se tendrán que definir las guías para los distintos aspectos del proceso, como pueden ser el modelado del negocio y los Casos de Uso, para la interfaz de usuario, el diseño, la programación, el manual de usuario.

Fuentes

- Jacaboson, I., Booch, G., Rumbaugh J., El Proceso Unificado de Desarrollo de Software, 2000 Addison Wesley
- Kruchten, P., The Rational Unified Process: An Introduction, 2000 Addison Wesley
- Kruchten, P. Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software 12 (6), November 1995, pp. 42-50.
- Rational Software Corporation, Product: Rational Software Corporation, 2002
- Rational Software Corporation, Rational Unified Process. Best Practices for Software Development Teams, 1998

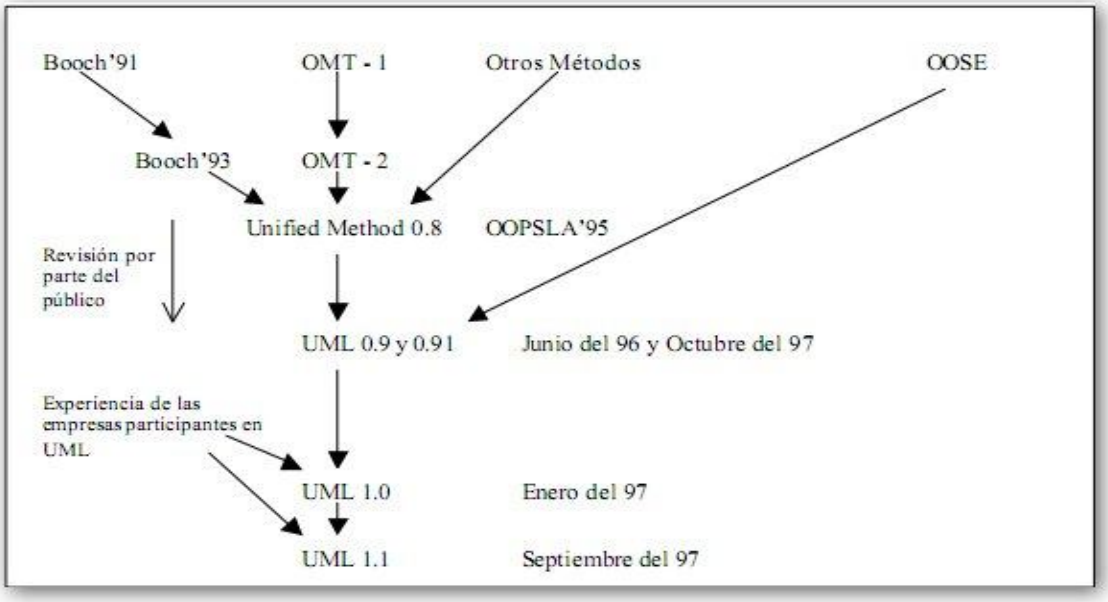
UML (Unified Modeling Language)

UML (*Unified Modeling Language*) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE. UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

Historia de UML



El objetivo principal cuando se empezó a gestar UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la imagen anterior se puede ver cuál ha sido la evolución de UML hasta la creación de UML 1.1.

Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación. En este apartado se sigue el método propuesto por Craig Larman que se ajusta a un ciclo de vida iterativo e incremental dirigido por casos de uso.

Notación Básica (Diagramas de Clase, Casos de Uso y Secuencia)

En esta parte se verá cómo se representan gráficamente en UML los conceptos principales de la orientación a objetos.

Modelos

Un modelo representa a un sistema software desde una perspectiva específica. Al igual que la planta y el alzado de una figura en dibujo técnico nos muestran la misma figura vista desde distintos ángulos, cada modelo nos permite fijarnos en un aspecto distinto del sistema.

Los modelos de UML que se tratan en esta parte son los siguientes:
· Diagrama de Estructura Estática
· Diagrama de Casos de Uso
· Diagrama de Secuencia
· Diagrama de Colaboración
· Diagrama de Estados.

Elementos Comunes a Todos los Diagramas

Notas

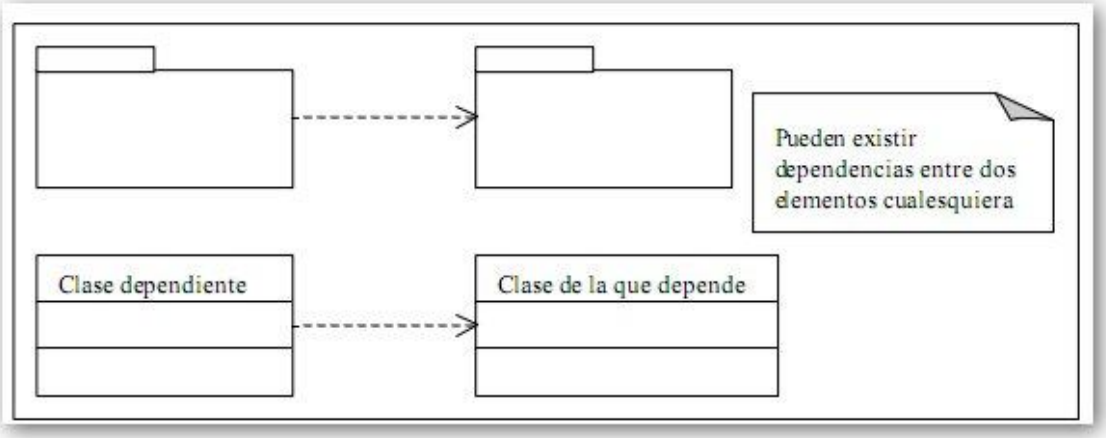
Una nota sirve para añadir cualquier tipo de comentario a un diagrama o a un elemento de un diagrama. Es un modo de indicar información en un formato libre, cuando la notación del diagrama en cuestión no nos permite expresar dicha información de manera adecuada.

Una nota se representa como un rectángulo con una esquina doblada con texto en su interior. Puede aparecer en un diagrama tanto sola como unida a un elemento por medio de una línea discontinua. Puede contener restricciones, comentarios, el cuerpo de un procedimiento, etc.

Dependencias

La relación de dependencia entre dos elementos de un diagrama significa que un cambio en el elemento destino puede implicar un cambio en el elemento origen (por tanto, si cambia el elemento destino habría que revisar el elemento origen).

Una dependencia se representa por medio de una línea de trazo discontinuo entre los dos elementos con una flecha en su extremo. El elemento dependiente es el origen de la flecha y el elemento del que depende es el destino (junto a él está la flecha).



Diagramas de Estructura Estática

Con el nombre de Diagramas de Estructura Estática se engloba tanto al Modelo Conceptual de la fase de Análisis como al Diagrama de Clases de la fase de Diseño. Ambos son distintos conceptualmente, mientras el primero modela elementos del dominio el segundo presenta los elementos de la solución software. Sin embargo, ambos comparten la misma notación para los elementos que los forman (clases y objetos) y las relaciones que existen entre los mismos (asociaciones).

Clases

Una clase se representa mediante una caja subdividida en tres partes: En la superior se muestra el nombre de la clase, en la media los atributos y en la inferior las operaciones. Una clase puede representarse de forma esquemática

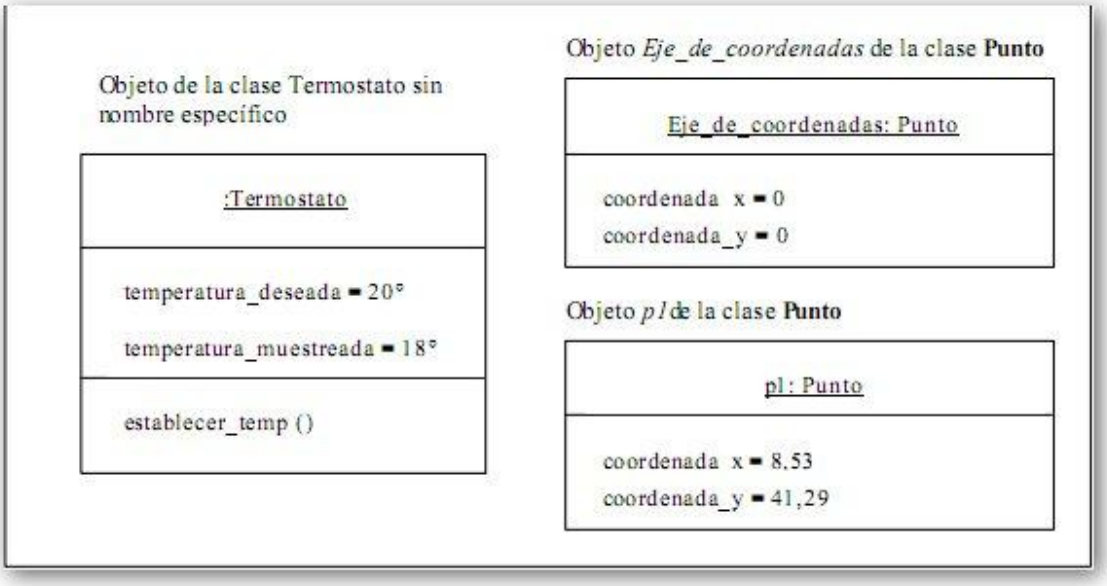
(plegada), con los detalles como atributos y operaciones suprimidos, siendo entonces tan solo un rectángulo con el nombre de la clase.

Objetos

Un objeto se representa de la misma forma que una clase. En el compartimiento superior aparece el nombre del objeto junto con el nombre de la clase subrayado, según la siguiente sintaxis:

nombre_del_objeto: nombre_de_la_clase

Puede representarse un objeto sin un nombre específico, entonces sólo aparece el nombre de la clase.



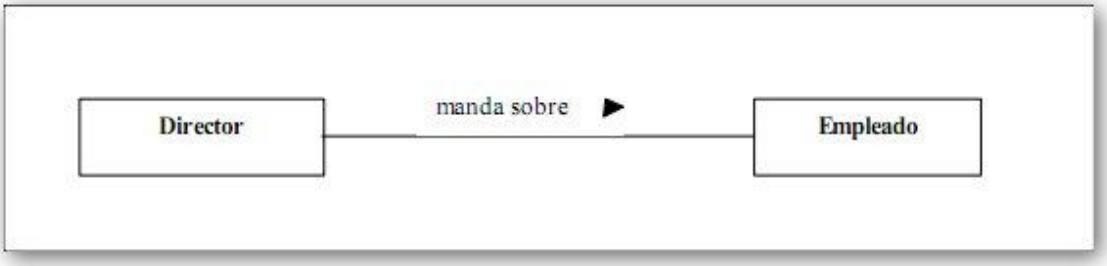
Asociaciones

Las asociaciones entre dos clases se representan mediante una línea que las une. La línea puede tener una serie de elementos gráficos que expresan características particulares de la asociación.

A continuación se verán los más importantes de entre dichos elementos gráficos.

Nombre de la Asociación y Dirección

El nombre de la asociación es opcional y se muestra como un texto que está próximo a la línea. Se puede añadir un pequeño triángulo negro sólido que indique la dirección en la cual leer el nombre de la asociación. En el ejemplo de la imagen siguiente se puede leer la asociación como "Director manda sobre Empleado".



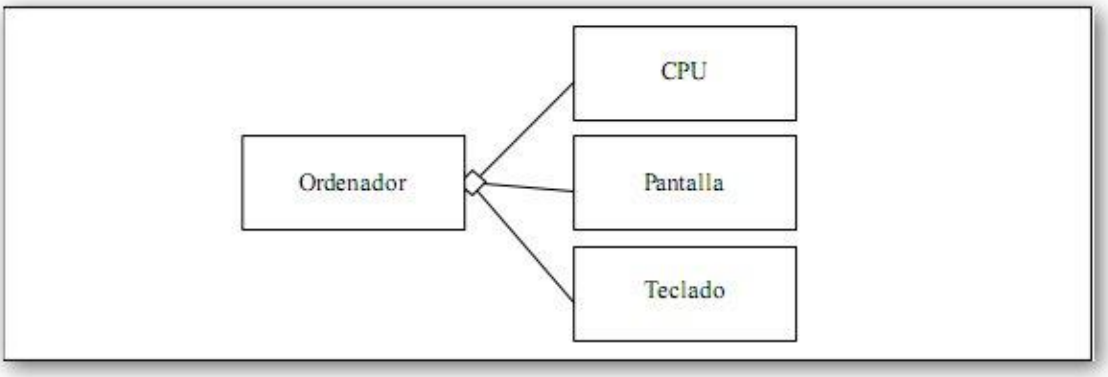
Los nombres de las asociaciones normalmente se incluyen en los modelos para aumentar la legibilidad. Sin embargo, en ocasiones pueden hacer

demasiado abundante la información que se presenta, con el consiguiente riesgo de saturación. En ese caso se puede suprimir el nombre de las asociaciones consideradas como suficientemente conocidas.

En las asociaciones de tipo agregación y de herencia no se suele poner el nombre.

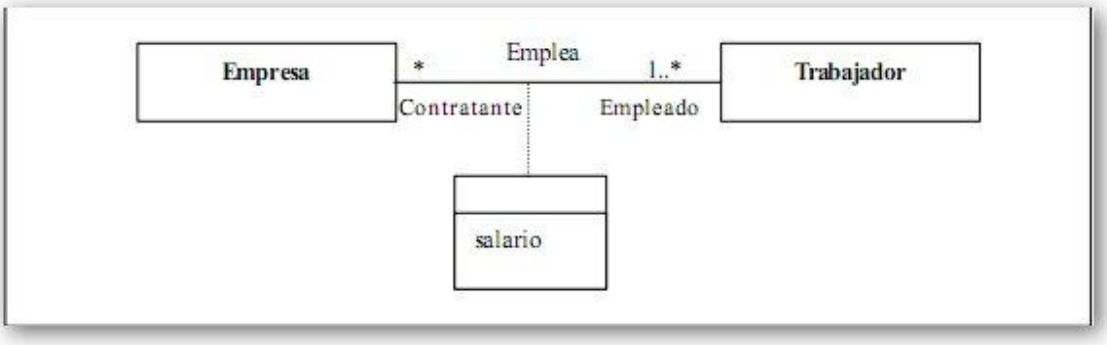
Agregación

El símbolo de agregación es un diamante colocado en el extremo en el que está la clase que representa el "todo".



Clases Asociación

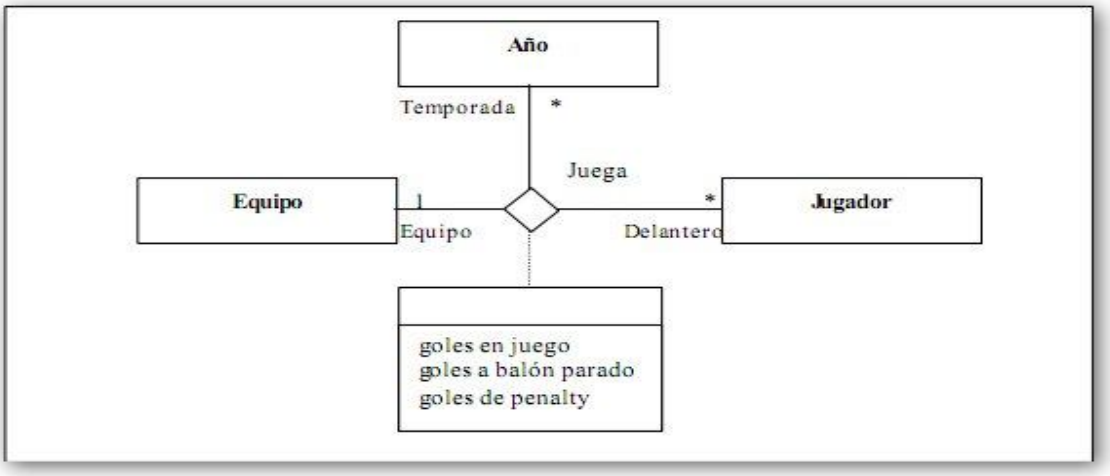
Cuando una asociación tiene propiedades propias se representa como una clase unida a la línea de la asociación por medio de una línea a trazos. Tanto la línea como el rectángulo de clase representan el mismo elemento conceptual: la asociación. Por tanto ambos tienen el mismo nombre, el de la asociación. Cuando la clase asociación sólo tiene atributos el nombre suele ponerse sobre la línea. Por el contrario, cuando la clase asociación tiene alguna operación o asociación propia, entonces se pone el nombre en la clase asociación y se puede quitar de la línea.



Asociaciones N-Arias

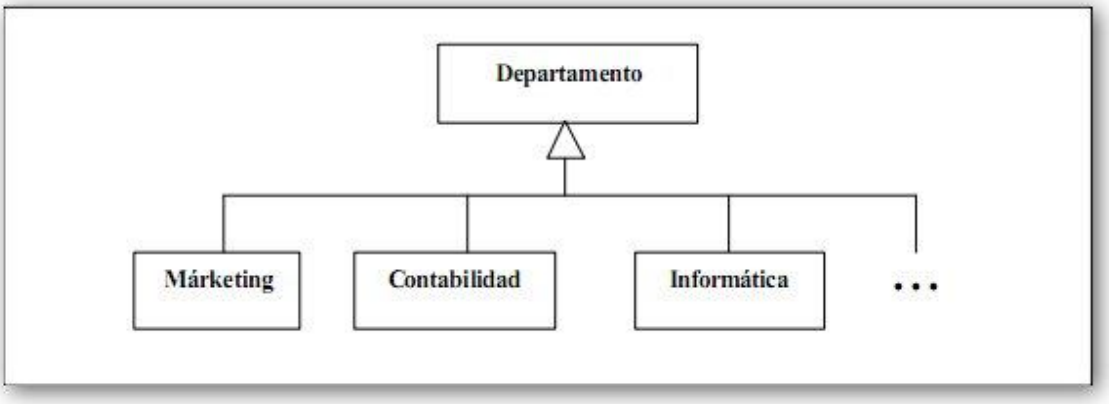
En el caso de una asociación en la que participan más de dos clases, las clases se unen con una línea a un diamante central. Si se muestra multiplicidad en un rol, representa el número potencial de tuplas de instancias en la asociación cuando el resto de los N-1 valores están fijos.

En la imagen siguiente se ha impuesto la restricción de que un jugador no puede jugar en dos equipos distintos a lo largo de una temporada, porque la multiplicidad de "Equipo" es 1 en la asociación ternaria.



Herencia

La relación de herencia se representa mediante un triángulo en el extremo de la relación que corresponde a la clase más general o clase "padre".



Si se tiene una relación de herencia con varias clases subordinadas, pero en un diagrama concreto no se quieren poner todas, esto se representa mediante puntos suspensivos. En el ejemplo de la imagen anterior, sólo aparecen en el diagrama 3 tipos de departamentos, pero con los puntos suspensivos se indica que en el modelo completo (el formado por todos los diagramas) la clase

"Departamento" tiene subclases adicionales, como podrían ser "Recursos Humanos" y "Producción".

Diagrama de Casos de Uso

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones:

- Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
- Incluye (Usa): Cuando un caso de uso utiliza a otro.

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<usa>> según sea el tipo de relación.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

En la siguiente imagen se muestra un ejemplo de Diagrama de Casos de Uso para un cajero automático.

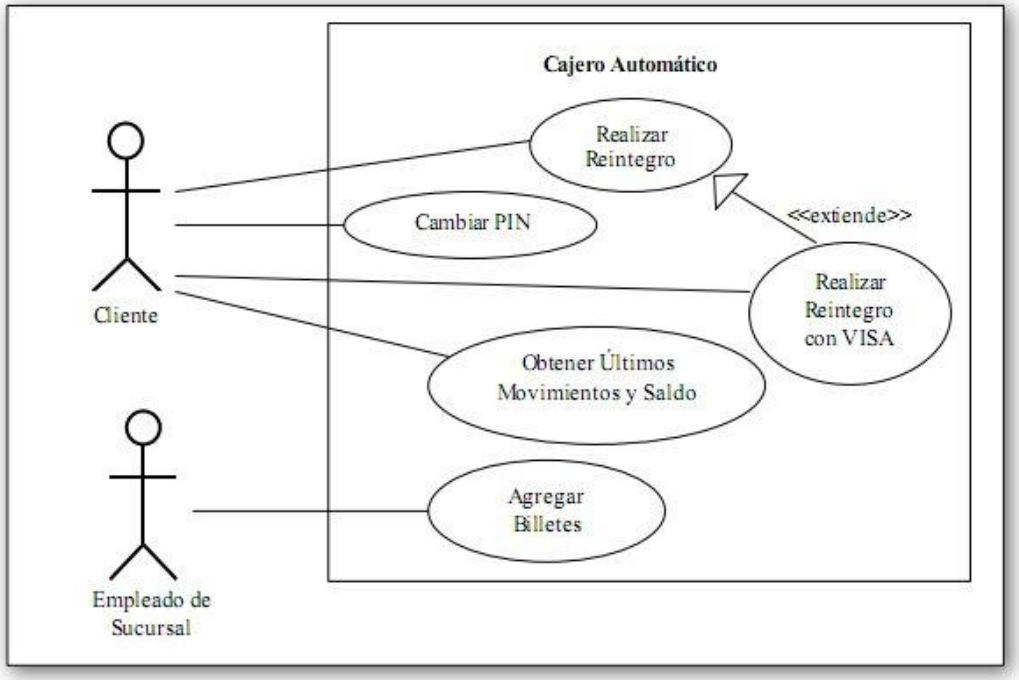
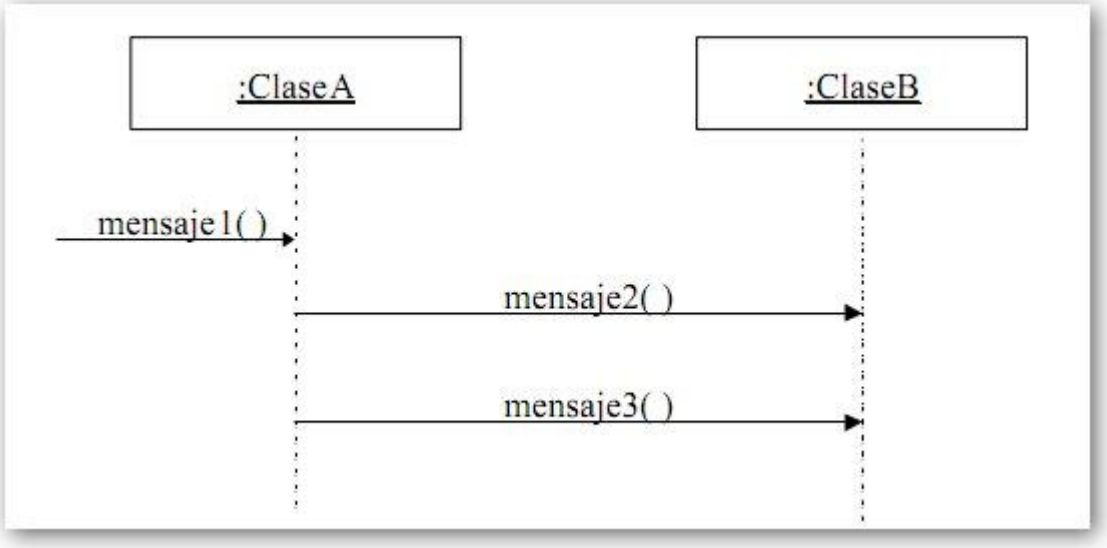


Diagrama de Secuencia

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.



Fuente

- Paper Universidad Tecnológica de Pereira

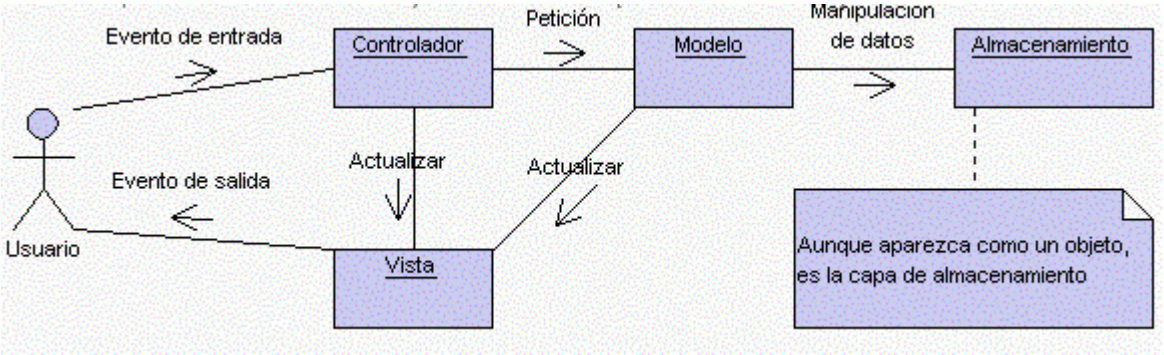
Patrón de Diseño MVC (Modelo-Vista-Controlador)

Para el diseño de aplicaciones con sofisticados interfaces se utiliza el patrón de diseño Modelo-Vista-Controlador. La lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño ofuscado, es decir, un pastiche que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando necesitemos cambiar el interfaz, tendremos que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error.

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

- Modelo: datos y reglas de negocio
- Vista: muestra la información del modelo al usuario
- Controlador: gestiona las entradas del usuario



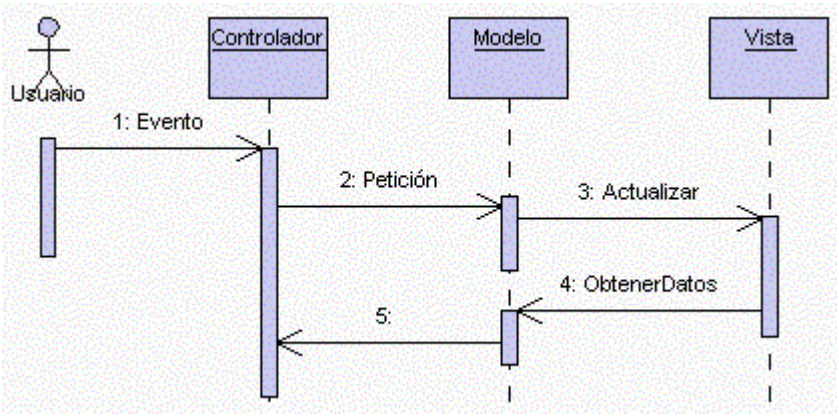
Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Veamos cada componente:

1. El **modelo** es el responsable de:
    - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
    - Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc).

2. El **controlador** es responsable de:
   - Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
   - Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_orden_de_venta)".
3. Las **vistas** son responsables de:
   - Recibir datos del modelo y los muestra al usuario.
   - Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
   - Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.



Pasos:

1. El usuario introduce el evento.
2. El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista)
3. El modelo (si es necesario) llama a la vista para su actualización
4. Para cumplir con la actualización la Vista puede solicitar datos al Modelo
5. El Controlador recibe el control.

Beneficios del patrón de diseño MVC en J2EE

La aplicación de la división de modelo-vista-controlador para del desarrollo de aplicaciones Web J2EE tiene varios beneficios:

- Puede distribuir el esfuerzo de desarrollo hasta cierto punto, tal que los cambios de implementación en una parte de la aplicación Web no requieran cambios en otros. El responsable del desarrollo para escribir la lógica de negocio puede trabajar independientemente de los responsables del desarrollo del flujo de control, y de los diseñadores de páginas Web y a su vez estos también pueden trabajar en forma independiente
- Puede prototipar más fácilmente sus trabajos
- Puede más fácilmente migrar aplicaciones legacy, porque la vista esta separada del modelo y del control y puede ser tolerable a diferentes plataformas y categorías de usuarios
- Puede mantener un ambiente que comprenda tecnologías diferentes a través de deferentes ubicaciones
- El diseño MVC tiene una estructura organizada que mejora la escalabilidad de soporte (construir aplicaciones más grandes) y fácil de modificar y mantener (debido a la clara separación de tareas).

Vista, controlador y Struts

La contribución de Struts para la vista es doble:

- Struts proporciona la clase Java org.apache.struts.action.ActionForm, que un desarrollador Java subclasea para crear un form bean. En tiempo de ejecución, el bean se usa en 2 formas:

--Cuando una JSP prepara el formulario HTML relacionado para mostrar, la JSP accede al bean, que toma valores para ser ocupados dentro del formulario. Estos valores se proporcionan para la lógica de negocio o desde previos ingresos del usuario.
--Cuando la entrada del usuario se retorna desde un Web browser, el bean valida y mantiene cualquier entrada para ser usada por la lógica de negocio

o (si la validación falla) para volver a mostrar la pantalla de ingresos de datos o alguna pantalla que despliegue el error.

- Struts proporciona numerosos tags de JSP a medida que son simples para usar pero potentes en el sentido que que ellos ocultan información. El diseñador de la pagina no necesita saber mucho acerca de beans de formularios, por ejemplo, no necesita saber más allá del nombre del bean y del nombre de cada uno de los campos en un bean dado.

Las contribuciones de Struts para el controlador son las siguientes:

- El servlet de action de un Struts maneja eventos en tiempo de ejecución acorde con un conjunto de reglas que se proporcionan en tiempos de desarrollo. Estas reglas están contenidas en un archivo de configuración de Struts y especifica como la respuesta del servlet para cada uno de los resultados recibidos desde la lógica de negocio, los cambios para el flojo de control requieren cambios solo en el archivo de configuración. El action servlet es una instanciación de la clase org.apache.struts.action.ActionServlet, conocida también como un simple ActionServlet.
- Los Struts también proveen la clase Java org.apache.struts.action.Action, que un desarrollador Java subclase a para crear un "action class".

Los Struts no contribuyen directamente al desarrollo del modelo, pero las acciones de los Struts y los archivos de configuración proporcionan una manera elegante para controlar las circunstancias bajo las cuales los componentes del modelo son invocados.

Anexo Nº 2

Tecnología a Utilizar

1.- Entorno de Desarrollo: IDE NetBeans

¿Qué es NetBeans?

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y creciendo!) en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform. NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Ambos productos son de código abierto y gratuitos para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la Common Development and Distribution License (CDDL) v1.0 and the GNU General Public License (GPL) v2.

2.- Lenguaje Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar la metodología de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Orientado a Objetos

La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto

de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su "comportamiento" (soldar dos piezas, etc.), el objeto "aluminio" puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de "código abierto" (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

Independencia de la plataforma

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, '''write once, run everywhere'''.

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode)—instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhere" como "Write once, debug everywhere" (o "Escríbelo una vez, ejecútalo en cualquier parte" por "Escríbelo una vez, depúralo en todas partes")

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

3.- Struts

Struts es un marco de trabajo (framework) utilizado para desarrollar aplicaciones web mediante la plataforma J2EE (Java 2 Enterprise Edition).

Implementa el patrón Modelo Vista Controlador (MVC) y se distribuye como software libre por la Apache Software Foundation.

- La parte de **Modelo** es donde recogemos la lógica de negocio de la aplicación web y nuestros objetos de negocio. Normalmente implica acceder a bases de datos. Es el punto más débil de Struts.
- La parte de **Vista** se implementa mediante la utilización de la tecnología JSP (Java Server Page) y taglibs.
- La parte de **Controlador** es implementada por una única Servlet proporcionada por Struts, ActionServlet, configurable mediante el fichero de propiedades struts-config.xml. Se encarga de la coordinación de las actividades a ejecutar, y del manejo de errores que estas actividades generan.

La utilización de esta metodología conlleva una serie de ventajas que nos ayudan a **reducir el tiempo requerido para el desarrollo** y **facilitan el mantenimiento** de la aplicación web:

- Transporte automático de los datos introducidos en el cliente (JSP) hasta el controlador (Action) mediante formularios (ActionForm).
- Transporte automático de los datos enviados por el controlador (Action) a la parte de presentación (JSP) mediante formularios (ActionForm).
- Implementa la parte común a todas las aplicaciones en la parte de Controlador (ActionServlet); la parte particular de cada aplicación es fácilmente configurable (struts-config.xml).
- La separación de los componentes en capas (MVC) simplifica notablemente el desarrollo y facilita su mantenimiento.

4.- Servlets

Los Servlet son la respuesta de la tecnología en Java a la programación de la Interfaz de Compuerta Común (CGI). Son programas que se ejecutan en el servidor, realizando la función de una capa intermedia entre una petición proveniente de un navegador Web u otro cliente HTTP, y las aplicaciones del servidor, pudiendo utilizar toda la paquetería y potencialidades del lenguaje. Su función principal es proveer páginas web dinámicas y personalizadas, utilizando para este objetivo el accesos a bases de datos, flujos de trabajo y otros recursos.

Desarrollo

Para poder utilizar los Servlets dentro de nuestras aplicaciones web, debemos, por norma general, complementar éste con un contenedor de Servlets, que no es mas que un servidor que tenga soporte para el API Servlet. En internet disponemos de multitud de contenedores de código libre y comerciales como el Tomcat de Apache, el WebLogic de IBM, el JSWDK de la SUN,etc. Sun mantiene una lista actualizada de contenedores de Servlets que se puede encontrar en: http://java.sun.com...rvlet/industry.html.

El API Servlet nos propone una jerarquía de clases bien definidas para el trabajo con los mismos, donde podemos encontrar desde servlets sin implementaciones definidas, en lo cuales tenemos que realizar todo el trabajo como la clase GenericServlet, o clases un poco mas avanzadas como la HttpServlet, donde solo tendríamos que implementar los métodos de acceso al mismo como el doGet() o el doPost(). El principal componente del API es la interfaz Servlet. La misma esta provista de los principales métodos para manipular, no solo los servlets, si no también la comunicación de estos con los clientes. Todos los servlets implementan esta interfaz de una manera directa o indirecta. A través de cualquiera de estas clases podemos comenzar a realizar nuestros propios servlets.

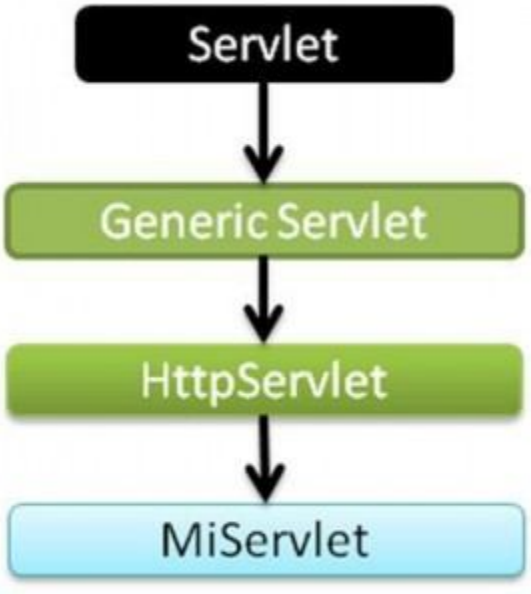


Figura 1. Jerarquía

Ventajas de los Sevlets

Eficientes

Con el CGI tradicional, se arranca un nuevo proceso por cada petición HTTP. Si el propio programa CGI es relativamente pequeño, el costo de arrancar el proceso puede superar el tiempo de ejecución. Además, si ocurren n peticiones simultáneas al mismo programa este carga su código en memoria la misma cantidad de veces. Con los Servlet la maquina virtual de java permanece en ejecución y administra cada petición mediante ligeros subprocesos de Java y no un pesado proceso del sistema operativo. A través de la programación por hilos pueden existir n subprocesos del mismo Servlet y este encontrarse en memoria una sola vez. Esto nos da una velocidad de respuesta mayor a cada petición y una eficiencia superior en el aprovechamiento de los recursos de la maquina donde son ejecutados.

Adecuados

Leer los datos enviados por un cliente HTTP desde un CGI tradicional es un trabajo bastante engorros, se necesita parsear la cadena completa y extraer de ella los datos uno a uno. Los Servlet cuentan con una extensa infraestructura para analizar y decodificar automáticamente los datos provenientes de un formulario HTML, leer y establecer encabezados HTTP, administrar las cookies, rastrear las sesiones y muchas otras utilidades de alto nivel como estas.

Transportables

Los Servlet no son mas que una clase Java, por lo que pueden ser tan portables como cualquier aplicación escrita en el mismo lenguaje. Esto los convierte en multiplataforma por defecto por lo que pueden ser ejecutados en cualquier sistema operativo. Además cuentan con un API estándar. La API Servlet no incluye nada acerca de cómo son cargados los servlets, ni el ambiente en el cual corren los servlets, ni el protocolo usado para transmitir los datos del usuario. Esto permite a los servlets poder ser usados por diferentes servidores Web. Se pueden cargar indiferentemente y de forma transparente tanto desde un disco local como desde una dirección remota, de forma totalmente transparentes. Es posible utilizarlos en servidores tan populares como el Apache, el FastTrack ó el Internet Information Server.
Los Servlets pueden comunicarse entre sí, y por tanto, es posible una reasignación dinámica de la carga de proceso entre diversas máquinas. Es decir, un servlet podría pasarle trabajo a otro servlet residente en otra máquina conociendo solamente la URL de este.

Seguros

Una de las principales fuentes de vulnerabilidad en los programas CGI tradicionales proviene del hecho de que por lo general son ejecutados por entornos de sistemas operativos de propósito general. Por ello, el programador de CGI debe tener cuidado de filtrar caracteres como las comillas tipográficas y los puntos y comas pues tienen un tratamiento especial por el entorno. Esto es más complejo de lo que podría pensarse, y los problemas derivados de esta situación son constantemente ignorados en diversas bibliotecas de CGI. Una segunda fuente de problemas es el hecho de que ciertos programas de CGI son procesados por lenguajes que no verifican automáticamente los límites de las matrices o cadenas. Por ejemplo, en C y C++ es perfectamente legal asignar una matriz de 100 elementos y describir en el "elemento" número 999, el cual puede ser un lugar aleatorio de la memoria del programa. Por ello, los programadores que olvidan realizar esta verificación, abren sus propios sistemas a posibles ataques de desbordamiento en el búfer ya sea por accidente o de manera deliberada. Los servlets no tienen estos problemas. Aun si un servlet ejecuta una llamada a un sistema distante para ejecutar un programa en el sistema operativo local, no utiliza el entorno del sistema operativo para lograrlo. Y por supuesto que la verificación de los límites de las matrices y otras características para la protección de la memoria es una parte central del lenguaje de programación Java.

Anexo Nº 3

Diccionario de Datos

Detalles del proyecto

Nombre del proyecto	Home Inspector DataBase
Descripción	Base de datos sistema Home Inspector
Autor	Jorge Del Rio Richter
Target DBMS	MySQL 5
Creado	2009-06-19
Modificado	2009-07-27

Detalles de Tablas

Tabla: CAPITULO

Detalles de Tabla:

Descripción	Almacena el catálogo de capítulos.

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_CAPITULO	INTEGER(10) UNSIGNED		Si		
	NOMBRE_CAPITULO	VARCHAR(45)		Si		
	DESC_CAPITULO	LONGTEXT		No		
	ESTADO_CAPITULO	VARCHAR(1)		Si		
PK, FK	ID_ESTRUCTURA	INTEGER(10) UNSIGNED		Si		

Tabla: DATOS_USUARIO

Detalles de Tabla:

Descripción	Almacena el catálogo de clientes

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_DATOS_USUARIO	INTEGER(10) UNSIGNED		Si		
	NOMBRE	VARCHAR(100)		Si		
	RUT	VARCHAR(11)		No		NULL
	FONO	VARCHAR(20)		No		NULL
	EMAIL	VARCHAR(45)		No		NULL
	DIRECCION	VARCHAR(100)		No		NULL
	COMUNA	VARCHAR(45)		No		
	PROVINCIA	VARCHAR(45)		No		

	REGION	VARCHAR(45)		No		
PK, FK	ID_USUARIO	INTEGER(10) UNSIGNED		Si		

Tabla: ESTRUCTURA

Detalles de Tabla:

Descripción	Almacena la estructura de una plantilla

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_ESTRUCTURA	INTEGER(10) UNSIGNED		Si		
	VERSION	INTEGER(11)		No		
	ESTADO	VARCHAR(5)		No		
PK	ID_PLANTILLA	INTEGER(10) UNSIGNED		Si		

Tabla: ITEM

Detalles de Tabla:

Descripción	Almacena el catálogo de Ítems

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_ITEM	INTEGER(10) UNSIGNED		Si		
	NOMBRE_ITEM	VARCHAR(200)		Si		
	DESC_ITEM	LONGTEXT		No		
	ESTADO_ITEM	VARCHAR(1)		Si		
PK, FK	ID_SECCION	INTEGER(10) UNSIGNED		Si		

Tabla: PLANTILLA

Detalles de Tabla:

Descripción	Almacena el catálogo de plantillas

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK, FK	ID_PLANTILLA	INTEGER(10) UNSIGNED		Si		
	NOMBRE_PLANTILLA	VARCHAR(45)		Si		

	DESC_PLANTILLA	LONGTEXT		Si		
	FECHA_CREACION	TIMESTAMP		Si		CURRENT_TIMESTAMP
	ESTADO_PLANTILLA	VARCHAR(10)		No		NULL
FK	ID_REPORTE	INTEGER(10) UNSIGNED		No		

Tabla: PROPIEDAD

Detalles de Tabla:

Descripción	Almacena Propiedades

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_PROPIEDAD	INTEGER(10) UNSIGNED		Si		
	ROL_PROPIEDAD	VARCHAR(20)		No		NULL
	DIRECCION_PROPIEDAD	VARCHAR(100)		Si		
	COMUNA_PROPIEDAD	VARCHAR(45)		No		NULL
	PROVINCIA_PROPIEDAD	VARCHAR(45)		No		NULL
	REGION_PROPIEDAD	VARCHAR(45)		No		NULL
	TIPO_PROPIEDAD	VARCHAR(20)		No		NULL
	AVALUO_PROPIEDAD	BIGINT(20)		No		NULL
	ESTADO	VARCHAR(3)		No		
PK, FK	ID_DATOS_USUARIO	INTEGER(10) UNSIGNED		Si		
PK, FK	ID_USUARIO	INTEGER(10) UNSIGNED		Si		

Tabla: REPORTE

Detalles de Tabla:

Descripción	Almacena el informe de una inspección

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_REPORTE	INTEGER(10) UNSIGNED		Si		
	FECHA_CREA_OT	DATE		Si		
	FECHA_PROGRAMADA	DATE		No		NULL
	FACTOR_HONORARIOS	FLOAT		No		NULL
	XML	LONGTEXT		No		
	FECHA_INICIO	DATE		No		NULL
	FECHA_FIN	DATE		No		NULL
	ID_CLIENTE	INTEGER(10)		No		

	ID_PLANTILLA	INTEGER(10)		No		
	ESTADO	VARCHAR(5)		No		
	ID_INSPECTOR	INTEGER(10)		No		
	ID_ADMIN	INTEGER(11)		No		
	TOTAL_HONORARIOS	DOUBLE		No		
	IVA	DOUBLE		No		
	TOTAL_FINAL	DOUBLE		No		
	ESTADO_OT	VARCHAR(20)		No		
PK, FK	ID_PROPIEDAD	INTEGER(10) UNSIGNED		Si		
PK, FK	ID_DATOS_USUARIO	INTEGER(10) UNSIGNED		Si		
PK, FK	ID_USUARIO	INTEGER(10) UNSIGNED		Si		

Tabla: SECCION

Detalles de Tabla:

Descripción	Almacena el catalogo de Secciones

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_SECCION	INTEGER(10) UNSIGNED		Si		
	NOMBRE_SECCION	VARCHAR(45)		Si		
	DESC_SECCION	LONGTEXT		No		
	ESTADO_SECCION	VARCHAR(1)		Si		
PK, FK	ID_CAPITULO	INTEGER(10) UNSIGNED		Si		

Tabla: SUBITEM

Detalles de Tabla:

Descripción	Almacena el catalogo de subitems

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_SUBITEM	INTEGER(11)		Si		
	NOMBRE_SUBITEM	VARCHAR(200)		Si		
	DESC_SUBITEM	LONGTEXT		No		
	ESTADO_SUBITEM	VARCHAR(2)		Si		
	MULTIPLE	TINYINT(1)		Si		
PK, FK	ID_ITEM	INTEGER(10) UNSIGNED		Si		

Tabla: USUARIO

Detalles de Tabla:

Descripción	Almacena el catálogo de usuarios

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_USUARIO	INTEGER(10) UNSIGNED		Si		
	NOMBRE_USUARIO	VARCHAR(20)		Si		
	PWD_USUARIO	VARCHAR(10)		Si		
	ROL	VARCHAR(20)		No		NULL
	ESTADO	VARCHAR(3)		Si		

Tabla: VALOR

Detalles de Tabla:

Descripción	Catálogo de valores, siempre asociados a un Ítem o Subitem

Columnas:

Key	Nombre Columna	Tipo de Dato	Dominio	No nulo	Check	Por Defecto
PK	ID_VALOR	INTEGER(10) UNSIGNED		Si		
	NOMBRE_VALOR	VARCHAR(100)		Si		
	DESCRIPCION_VALOR	MEDIUMTEXT		No		
	ESTADO_VALOR	VARCHAR(2)		Si		
PK, FK	ID_SUBITEM	INTEGER(11)		Si		

Detalles de Columnas

Columna: ID_CAPITULO

Columna detalles:

Tabla nombre	CAPITULO
Descripción	Identificador único de capítulos
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_CAPITULO

Columna detalles:

Tabla nombre	CAPITULO
Descripción	Nombre del capitulo
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	Si

Columna: DESC_CAPITULO

Columna detalles:

Tabla nombre	CAPITULO
Descripción	Descripción de capitulo
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	No

Columna: ESTADO_CAPITULO

Columna detalles:

Tabla nombre	CAPITULO
Descripción	Activo o no
Clave Primaria	No
Tipo de Dato	VARCHAR(1)
Script Tipo de Dato	VARCHAR(1)
No nulo	Si

Columna: ID_ESTRUCTURA

Columna detalles:

Tabla nombre	CAPITULO
Descripción	Identificador estructura
Clave Primaria	Si
Referencia a	ID_ESTRUCTURA
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_DATOS_USUARIO

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Identificador unico del usuario
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Nombre del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(100)
Script Tipo de Dato	VARCHAR(100)
No nulo	Si

Columna: RUT

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	RUT del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(11)
Script Tipo de Dato	VARCHAR(11)
No nulo	No
Por Defecto	NULL

Columna: FONO

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Telefono del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	No
Por Defecto	NULL

Columna: EMAIL

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Correo electronico del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No
Por Defecto	NULL

Columna: DIRECCION

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Direccion del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(100)
Script Tipo de Dato	VARCHAR(100)
No nulo	No
Por Defecto	NULL

Columna: COMUNA

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Comuna en que reside el usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No

Columna: PROVINCIA

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Provincia en que reside el usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No

Columna: REGION

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Region en que reside el usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No

Columna: ID_USUARIO

Columna detalles:

Tabla nombre	DATOS_USUARIO
Descripción	Identificador único del usuario
Clave Primaria	Si
Referencia a	ID_USUARIO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_ESTRUCTURA

Columna detalles:

Tabla nombre	ESTRUCTURA
Descripción	Identificador único de estructura
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: VERSION

Columna detalles:

Tabla nombre	ESTRUCTURA
Descripción	Versión de la estructura de plantilla
Clave Primaria	No
Tipo de Dato	INTEGER(11)
Script Tipo de Dato	INTEGER(11)
No nulo	No

Columna: ESTADO

Columna detalles:

Tabla nombre	ESTRUCTURA
Descripción	Activa o no
Clave Primaria	No
Tipo de Dato	VARCHAR(5)
Script Tipo de Dato	VARCHAR(5)
No nulo	No

Columna: ID_PLANTILLA

Columna detalles:

Tabla nombre	ESTRUCTURA
Descripción	Identificador de plantilla
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_ITEM

Columna detalles:

Tabla nombre	ITEM
Descripción	Identificador Ãºnico de Ãtem
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_ITEM

Columna detalles:

Tabla nombre	ITEM
Descripción	Nombre del Ãtem
Clave Primaria	No
Tipo de Dato	VARCHAR(200)
Script Tipo de Dato	VARCHAR(200)
No nulo	Si

Columna: DESC_ITEM

Columna detalles:

Tabla nombre	ITEM
Descripción	Descripción del ítem
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	No

Columna: ESTADO_ITEM

Columna detalles:

Tabla nombre	ITEM
Descripción	Activo o no
Clave Primaria	No
Tipo de Dato	VARCHAR(1)
Script Tipo de Dato	VARCHAR(1)
No nulo	Si

Columna: ID_SECCION

Columna detalles:

Tabla nombre	ITEM
Descripción	sección padre
Clave Primaria	Si
Referencia a	ID_SECCION
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_PLANTILLA

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Identificador único de plantillas
Clave Primaria	Si
Referencia a	ID_PLANTILLA
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_PLANTILLA

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Nombre de la plantilla
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	Si

Columna: DESC_PLANTILLA

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Descripción de la plantilla
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	Si

Columna: FECHA_CREACION

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Almacena la fecha en que se creó la plantilla
Clave Primaria	No
Tipo de Dato	TIMESTAMP
Script Tipo de Dato	TIMESTAMP
No nulo	Si
Por Defecto	CURRENT_TIMESTAMP

Columna: ESTADO_PLANTILLA

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Activa o no
Clave Primaria	No
Tipo de Dato	VARCHAR(10)
Script Tipo de Dato	VARCHAR(10)
No nulo	No
Por Defecto	NULL

Columna: ID_REPORTE

Columna detalles:

Tabla nombre	PLANTILLA
Descripción	Identificador único del reporte
Clave Primaria	No
Referencia a	ID_REPORTE
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	No

Columna: ID_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Identificador único de propiedad
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ROL_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Rol SII de la propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	No
Por Defecto	NULL

Columna: DIRECCION_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Direccion de la propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(100)
Script Tipo de Dato	VARCHAR(100)
No nulo	Si

Columna: COMUNA_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Comuna de la propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No
Por Defecto	NULL

Columna: PROVINCIA_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Provincia de la propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No
Por Defecto	NULL

Columna: REGION_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Region de la Propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	No
Por Defecto	NULL

Columna: TIPO_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Identifica si es casa, departamento, etc.
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	No
Por Defecto	NULL

Columna: AVALUO_PROPIEDAD

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Avalúo SII
Clave Primaria	No
Tipo de Dato	BIGINT(20)
Script Tipo de Dato	BIGINT(20)
No nulo	No
Por Defecto	NULL

Columna: ESTADO

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Estado de la propiedad
Clave Primaria	No
Tipo de Dato	VARCHAR(3)
Script Tipo de Dato	VARCHAR(3)
No nulo	No

Columna: ID_DATOS_USUARIO

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Propietario
Clave Primaria	Si
Referencia a	ID_DATOS_USUARIO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_USUARIO

Columna detalles:

Tabla nombre	PROPIEDAD
Descripción	Propietario
Clave Primaria	Si
Referencia a	ID_USUARIO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_REPORTE

Columna detalles:

Tabla nombre	REPORTE
Descripción	Identificador unico del reporte
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: FECHA_CREA_OT

Columna detalles:

Tabla nombre	REPORTE
Descripción	Fecha de creacion de la OT
Clave Primaria	No
Tipo de Dato	DATE
Script Tipo de Dato	DATE
No nulo	Si

Columna: FECHA_PROGRAMADA

Columna detalles:

Tabla nombre	REPORTE
Descripción	Fecha y Hora en que debe realizarse la inspeccion
Clave Primaria	No
Tipo de Dato	DATE
Script Tipo de Dato	DATE
No nulo	No
Por Defecto	NULL

Columna: FACTOR_HONORARIOS

Columna detalles:

Tabla nombre	REPORTE
Descripción	Factor impuesto por la empresa
Clave Primaria	No
Tipo de Dato	FLOAT
Script Tipo de Dato	FLOAT
No nulo	No
Por Defecto	NULL

Columna: XML

Columna detalles:

Tabla nombre	REPORTE
Descripción	Plantilla y datos del reporte en formato XML
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	No

Columna: FECHA_INICIO

Columna detalles:

Tabla nombre	REPORTE
Descripción	Fecha de inicio del informe
Clave Primaria	No
Tipo de Dato	DATE
Script Tipo de Dato	DATE
No nulo	No
Por Defecto	NULL

Columna: FECHA_FIN

Columna detalles:

Tabla nombre	REPORTE
Descripción	Fecha de finalizaciÃ³n del informe
Clave Primaria	No
Tipo de Dato	DATE
Script Tipo de Dato	DATE
No nulo	No
Por Defecto	NULL

Columna: ID_CLIENTE

Columna detalles:

Tabla nombre	REPORTE
Descripción	Identificador dueño de la propiedad
Clave Primaria	No
Tipo de Dato	INTEGER(10)
Script Tipo de Dato	INTEGER(10)
No nulo	No

Columna: ID_PLANTILLA

Columna detalles:

Tabla nombre	REPORTE
Descripción	Plantilla del informe
Clave Primaria	No
Tipo de Dato	INTEGER(10)
Script Tipo de Dato	INTEGER(10)
No nulo	No

Columna: ESTADO

Columna detalles:

Tabla nombre	REPORTE
Descripción	estado del informe
Clave Primaria	No
Tipo de Dato	VARCHAR(5)
Script Tipo de Dato	VARCHAR(5)
No nulo	No

Columna: ID_INSPECTOR

Columna detalles:

Tabla nombre	REPORTE
Descripción	Inspector que realiza la inspeccion
Clave Primaria	No
Tipo de Dato	INTEGER(10)
Script Tipo de Dato	INTEGER(10)
No nulo	No

Columna: ID_ADMIN

Columna detalles:

Tabla nombre	REPORTE
Descripción	Administrador del informe
Clave Primaria	No
Tipo de Dato	INTEGER(11)
Script Tipo de Dato	INTEGER(11)
No nulo	No

Columna: TOTAL_HONORARIOS

Columna detalles:

Tabla nombre	REPORTE
Descripción	valor Inspeccion
Clave Primaria	No
Tipo de Dato	DOUBLE
Script Tipo de Dato	DOUBLE
No nulo	No

Columna: IVA

Columna detalles:

Tabla nombre	REPORTE
Descripción	Impuesto
Clave Primaria	No
Tipo de Dato	DOUBLE
Script Tipo de Dato	DOUBLE
No nulo	No

Columna: TOTAL_FINAL

Columna detalles:

Tabla nombre	REPORTE
Descripción	valor final de la inspeccion
Clave Primaria	No
Tipo de Dato	DOUBLE
Script Tipo de Dato	DOUBLE
No nulo	No

Columna: ESTADO_OT

Columna detalles:

Tabla nombre	REPORTE
Descripción	estado OT
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	No

Columna: ID_PROPIEDAD

Columna detalles:

Tabla nombre	REPORTE
Descripción	Identificador de la propiedad
Clave Primaria	Si
Referencia a	ID_PROPIEDAD
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_DATOS_USUARIO

Columna detalles:

Tabla nombre	REPORTE
Descripción	Identificador del propietario
Clave Primaria	Si
Referencia a	ID_DATOS_USUARIO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_USUARIO

Columna detalles:

Tabla nombre	REPORTE
Descripción	Identificador del propietario
Clave Primaria	Si
Referencia a	ID_USUARIO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_SECCION

Columna detalles:

Tabla nombre	SECCION
Descripción	Identificador unico de secciones
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_SECCION

Columna detalles:

Tabla nombre	SECCION
Descripción	Nombre de la seccion
Clave Primaria	No
Tipo de Dato	VARCHAR(45)
Script Tipo de Dato	VARCHAR(45)
No nulo	Si

Columna: DESC_SECCION

Columna detalles:

Tabla nombre	SECCION
Descripción	Descripcion de la seccion
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	No

Columna: ESTADO_SECCION

Columna detalles:

Tabla nombre	SECCION
Descripción	Activa o no
Clave Primaria	No
Tipo de Dato	VARCHAR(1)
Script Tipo de Dato	VARCHAR(1)
No nulo	Si

Columna: ID_CAPITULO

Columna detalles:

Tabla nombre	SECCION
Descripción	Capitulo padre
Clave Primaria	Si
Referencia a	ID_CAPITULO
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_SUBITEM

Columna detalles:

Tabla nombre	SUBITEM
Descripción	Identificador Ãºnico de Subitems
Clave Primaria	Si
Tipo de Dato	INTEGER(11)
Script Tipo de Dato	INTEGER(11)
No nulo	Si

Columna: NOMBRE_SUBITEM

Columna detalles:

Tabla nombre	SUBITEM
Descripción	Nombre sub item
Clave Primaria	No
Tipo de Dato	VARCHAR(200)
Script Tipo de Dato	VARCHAR(200)
No nulo	Si

Columna: DESC_SUBITEM

Columna detalles:

Tabla nombre	SUBITEM
Descripción	Descripción del subitem
Clave Primaria	No
Tipo de Dato	LONGTEXT
Script Tipo de Dato	LONGTEXT
No nulo	No

Columna: ESTADO_SUBITEM

Columna detalles:

Tabla nombre	SUBITEM
Descripción	activo o no
Clave Primaria	No
Tipo de Dato	VARCHAR(2)
Script Tipo de Dato	VARCHAR(2)
No nulo	Si

Columna: MULTIPLE

Columna detalles:

Tabla nombre	SUBITEM
Descripción	Determina si es objeto de seleccion multiple
Clave Primaria	No

Tipo de Dato	TINYINT(1)
Script Tipo de Dato	TINYINT(1)
No nulo	Si

Columna: ID_ITEM

Columna detalles:

Tabla nombre	SUBITEM
Descripción	Item padre
Clave Primaria	Si
Referencia a	ID_ITEM
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: ID_USUARIO

Columna detalles:

Tabla nombre	USUARIO
Descripción	Identificador unico del usuario
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_USUARIO

Columna detalles:

Tabla nombre	USUARIO
Descripción	Nombre de acceso al sistema
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	Si

Columna: PWD_USUARIO

Columna detalles:

Tabla nombre	USUARIO
Descripción	Contraseña del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(10)
Script Tipo de Dato	VARCHAR(10)
No nulo	Si

Columna: ROL

Columna detalles:

Tabla nombre	USUARIO
Descripción	Tipo de Usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(20)
Script Tipo de Dato	VARCHAR(20)
No nulo	No
Por Defecto	NULL

Columna: ESTADO

Columna detalles:

Tabla nombre	USUARIO
Descripción	Estado del usuario
Clave Primaria	No
Tipo de Dato	VARCHAR(3)
Script Tipo de Dato	VARCHAR(3)
No nulo	Si

Columna: ID_VALOR

Columna detalles:

Tabla nombre	VALOR
Descripción	Identificador único de valores
Clave Primaria	Si
Tipo de Dato	INTEGER(10) UNSIGNED
Script Tipo de Dato	INTEGER(10) UNSIGNED
No nulo	Si

Columna: NOMBRE_VALOR

Columna detalles:

Tabla nombre	VALOR
Descripción	Nombre del valor que aparecerá en la lista selecionable.
Clave Primaria	No
Tipo de Dato	VARCHAR(100)
Script Tipo de Dato	VARCHAR(100)
No nulo	Si

Columna: DESCRIPCION_VALOR

Columna detalles:

Tabla nombre	VALOR
Descripción	DescripciÃ³n del valor
Clave Primaria	No
Tipo de Dato	MEDIUMTEXT
Script Tipo de Dato	MEDIUMTEXT

Constraints:

No nulo	No

Columna: ESTADO_VALOR

Columna detalles:

Tabla nombre	VALOR
Descripción	Activo o no
Clave Primaria	No
Tipo de Dato	VARCHAR(2)
Script Tipo de Dato	VARCHAR(2)
No nulo	Si

Columna: ID_SUBITEM

Columna detalles:

Tabla nombre	VALOR
Descripción	sub item padre
Clave Primaria	Si
Referencia a	ID_SUBITEM
Tipo de Dato	INTEGER(11)
Script Tipo de Dato	INTEGER(11)
No nulo	Si