Universidad de las Ciencias Informáticas
Facultad 3

Título: Análisis y diseño del Observatorio Internacional de Gobierno Electrónico para Cuba.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Aron Parchment Llamos.

Tutor: MsC. Yarina Amoroso Fernandez.

Ciudad de la Habana, 2011

Declaración de Autoría

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a___ días del mes de _____ del año 2009.

_____ _____

Autor: Aron Parchment Llamos Ms C. Yarina Amoroso Fernández

Resumen

En la actualidad las Tecnologías de la Información y las Comunicaciones han tomado un gigantesco y acelerado avance marcando un cambio significativo en el mundo en cuanto a informática se refiere, ésta se ha expandido en varios niveles, surgiendo así nuevas tendencias como “Gobierno Electrónico”.

La presente investigación persigue como objetivo realizar el análisis y diseño del Observatorio Internacional de gobierno electrónico para Cuba. Para ello se realizó un estudio sobre: conceptos asociados a Observatorios Internacionales de Gobierno Electrónico y las funcionalidades de sistemas similares en el mundo, metodologías de desarrollo de software, herramientas case, lenguajes de modelado y de programación, además de algunos aspectos asociados al flujo de trabajo Análisis y Diseño definido en el Proceso Unificado de Desarrollo (RUP).

El sistema tiene como función garantizar un mecanismo que permita servir de espacio de colaboración multidisciplinaria a todos los implicados en el desarrollo del Gobierno Electrónico en el país.

Palabras claves: Gobierno electrónico, Observatorio, Observatorio internacional, Análisis, Diseño.

Índice

Introducción

En nuestros días, las Tecnologías de la Información y las Comunicaciones (TIC) han establecido una configuración moderna a la sociedad actual, un cambio no solo a la informática y su tecnología adjunta, sino también a todos los medios de comunicación en general; tienen como objetivo mejorar la calidad de vida de las personas y satisfacer las necesidades que tiene el hombre consigo mismo y con su entorno.

El avance de estas tecnologías a nivel mundial, la velocidad con la que se realizan distintos tipos de operaciones a través de la utilización de internet; provocan un cambio fundamental en la forma habitual de relacionarse, el gobierno en todos sus niveles se ha ido adecuando a estas nuevas tendencias , surgiendo así lo que se conoce como "Gobierno Electrónico"; el cual no es más que el empleo de la internet y las TIC para conseguir una mejor administración del gobierno mediante la transparencia y el acceso público a la información, logrando un mayor entendimiento entre el sector público y los ciudadanos, lo cual constituye un elemento estratégico para el desarrollo de muchos países, puesto que están resolviendo buena parte de los problemas de funcionamiento de sus gobiernos mediante la incorporación masiva de las nuevas tecnologías de la información y las comunicaciones. (Alanis, 2008)

Una manera de llevar lo anterior en práctica es a través de los observatorios internacionales EGOBS (Electronic Government Observatory), cuyo objetivo fundamental es el estudio independiente de las características de las realizaciones en materia de gobierno electrónico, o sea, es el conjunto de organizaciones y sistemas precisos para que se produzca la prestación de servicios administrativos a ciudadanos y empresas por las administraciones públicas con auxilio de los recursos que facilitan las TIC. Los trabajos realizados ponen especial énfasis en la comprobación referida a si la prestación se realiza con respeto a las normas, procedimientos y principios recogidos en las leyes, incluido el derecho a la participación en dichas actividades por ciudadanos, empresas e instituciones y, especialmente, si se satisface en su puesta en práctica las regulaciones sobre protección de datos y seguridad de las comunicaciones electrónicas. (Galindo, 2007)

Actualmente la creación de Observatorios de Gobierno electrónico en los países en desarrollo es muy recurrido, pues posibilitan a los mismos anticiparse a las oportunidades, prevenir las amenazas, facilitar la toma de decisiones, potenciar una percepción dinámica del cambio en cuanto a gobierno se refire . Estos servicios operan con éxito en países de Europa, Latinoamérica, Asia; que tienen como objetivo gestionar información, fomentar el desarrollo científico-técnico y promover iniciativas gubernamentales , apoyados en estrategias de la Inteligencia Empresarial y la Gestión de Información de forma general. Uno de los problemas

más importantes por el que atraviesan los decisores y gerentes de políticas de Gobierno Electrónico radica en saber priorizar y seleccionar los servicios que deben digitalizarse, en el marco de un plan estratégico. Esta actividad que pareciera ser simple y evidente a la vez, trae aparejado un conjunto de situaciones que deben analizarse para evitar el despilfarro de recursos y de tiempo en servicios "poco efectivos".Existen escasas dudas sobre la importancia creciente del uso de Internet por parte de los gobiernos, en América Latina y el Caribe. En los últimos años se ha observado la aparición de portales gubernamentales en la mayoría de los países de la región, los cuales tienen una cara pública en línea.

La situación en América Latina y el Caribe en materia de Gobierno electrónico es heterogénea. Así lo muestra el Índice de Gobierno Electrónico de Naciones Unidas (IGE). Este índice es calculado por el Departamento de Asuntos Económicos y Sociales a través de una encuesta internacional de Naciones Unidas y representa el estado de desarrollo de los gobiernos en relación al Gobierno electrónico. Es una medida compuesta que da cuenta de la capacidad y voluntad de los gobiernos de usar las TIC para el desarrollo. Considera tres dimensiones: a) Servicios Web de los Gobiernos; b) Infraestructura de telecomunicaciones, y c) capital humano.El IGE 2010 varía entre 0 y 1; en América Latina, los mayores valores del mismo los tienen Colombia (0.6125), Chile (0.6014) y Uruguay (0.5848). Como se observa en la figura 1 hay países que con similares Producto Interno Bruto (PIB) per cápita tienen valores diferentes de este índice (Chile y México) y otros que teniendo diferentes niveles de PIB per cápita presentan similares valores del IGE (Costa Rica, Venezuela, San Kitts y Nevis y Trinidad y Tabago). (Proyectos, 2010)

El Centro de Gobierno Electrónico (CEGEL) de la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI) tiene entre sus objetivos rectores el desarrollo de soluciones informáticas orientadas a informatizar la gestión del gobierno para que, en los límites establecidos por la constitución y las leyes, la administración pública opere en forma eficiente, transparente y beneficiosa para los ciudadanos. En este sentido, el estudio de disímiles temáticas y experiencias en materia de gobierno electrónico, constituye un componente fundamental para lograr un desarrollo ascendente en el ámbito del gobierno electrónico.
En función de garantizar un mecanismo que permita servir de espacio de colaboración multidisciplinaria a todos los implicados en el desarrollo del gobierno electrónico en el país, se requiere de la definición de un Observatorio de Gobierno Electrónico que se enfoque en tres áreas fundamentales:

- Tecnológica: Utilización de tecnologías para la definición y desarrollo de plataformas de gobierno electrónico que permitan desarrollar estudios de tendencias. asociadas a esta temática.

- Socio-Jurídica: Componente orientada al recopilación, publicación y análisis de información asociada a temáticas jurídicas con impacto social.

- Normativo: Recopilación, publicación y análisis de normas internacionales que inciden en el gobierno electrónico en sentido general, y que permitan sentar las bases para desarrollar Entornos Normativos.

Este observatorio deberá contar con un portal web que sirva de espacio de debate de temáticas de interés, publicación de información y resultados de investigación y desarrollo, así como recopilación de datos para el estudio de tendencias en el mundo asociadas a esta temática.

Por lo anteriormente planteado, se tiene como **problema científico:** ¿Cómo traducir los procesos de gestión de la información de un Observatorio Internacional de Gobierno Electrónico, a una notación entendible para los desarrolladores permitiendo su posterior implementación?

Objeto de estudio: Proceso de Desarrollo de Software.

El mismo tiene como **campo de acción**: Análisis y Diseño de un Observatorio Internacional de Gobierno Electrónico.

La investigación que se desarrolla tiene como **objetivo general**: Realizar el análisis y diseño de un Observatorio Internacional de Gobierno Electrónico en Cuba que facilite la posterior implementación de los procesos identificados.

Como **objetivos específicos** se plantean los siguientes:

- Realizar estudio del estado del arte de los observatorios internacionales.
- Definir las bases estructurales y funcionales del Observatorio Internacional de Gobierno Electrónico.
- Realizar análisis y diseño del observatorio.

Para dar respuesta a la interrogante planteada en el problema científico se parte de la siguiente **Idea a defender:** Si se realiza el análisis y diseño de un Observatorio Internacional de Gobierno Electrónico, se facilitará la posterior implementación de los procesos identificados.

Las **tareas de la investigación** a desarrollar para cumplir los objetivos trazados son:

- Realización de un estudio del estado del arte sobre el gobierno electrónico en Cuba y el mundo.
- Análisis de las metodologías de desarrollo, herramientas de modelado, posibles a utilizar en el desarrollo de la solución.
- Elicitación de requisitos.
- Especificación de requisitos funcionales y no funcionales.
- Definición de los casos de uso.
- Descripción del Modelo de Casos de Uso del sistema.
- Determinación de las clases del análisis para el desarrollo del sistema.
- Realización del diagrama de secuencia.

- Determinación de las clases del diseño.
- Realización del diagrama de clases del diseño.
- Validación de la propuesta de solución.

El presente trabajo de diploma deberá derivar como resultado concreto:

- Definición estructural y bases para el funcionamiento del Observatorio Internacional.

Estructura de la Tesis

El presente trabajo de diploma consta de tres capítulos:

Capítulo 1. Fundamentación Teórica. Se realiza un estudio sobre algunas de las metodologías para el desarrollo de software, los lenguajes de modelado, herramientas CASE que pueden utilizarse para modelar los artefactos que proponen dichas metodologías, así como los diferentes lenguajes de modelado y de programación. Se tratan aspectos concernientes a la ingeniería de requisitos y el diseño. Se abordan además los principales patrones de casos uso y de diseño existentes.

Capítulo 2. Solución Propuesta. En este capítulo se realizan las actividades de elicitación, análisis y especificación de los requisitos, obteniendo los requisitos funcionales y no funcionales del modelo. Se realizan los diagramas de casos de uso y sus respectivas especificaciones, además del diseño de los diagramas de clases y de secuencia.

Capítulo 3. Validación de los Resultados. En este capítulo se verifica y valida la solución propuesta, a partir de la aplicación de las métricas: para la calidad de la funcionalidad del Documento de Especificación de Requisitos, evaluación de la funcionalidad del diagrama de casos de uso, también se emplearon otras para la validación del diseño como, cohesión y tamaño de clases, realizando un análisis de los resultados obtenidos con las mismas.

***Capítulo 1: Fundamentación Teórica*.**

1.1 Introducción

En el presente capítulo se abordan diferentes aspectos y conceptos relacionados con los observatorios internacionales, y se hace un análisis de la importancia que brindan cada uno de estos sistemas.

Además se propone la tecnología en la cual se va a desarrollar el producto. Se describe la misma, y se plantean los argumentos que se tuvieron en cuenta a la hora de seleccionar este tipo de tecnología. Así como la descripción y ventajas de desarrollar la misma. Y finalmente se describe la metodología y las herramientas con las cuales se va a desarrollar el producto.

1.1.1. Surgimiento, evolución e importancia de los Observatorios de Gobierno Electrónico.

El vocablo “Observatorio” se asocia a un lugar (edificio) o posición que sirve para hacer observaciones por medio de instrumentos apropiados y dedicados a observaciones comúnmente astronómicas o meteorológicas; pero hoy las empresas lo han asumido con gran popularidad como herramienta colectiva para obtener y ofertar información relevante, orientado al apoyo de los procesos de innovación tecnológica de las organizaciones empresariales, sociales y/o gubernamentales.

Los observatorios se definen como el producto de la unión de un grupo de personas con una amplia visión en materia de negocios, basada en la experiencia que han acumulado a lo largo del tiempo de trabajo en la empresa y grupos de expertos en la materia a tratar. (Ortega, 2009)

Los observatorios de información, permiten hacer un estudio y análisis de indicadores en cualquier campo del conocimiento, y pueden abarcar los datos presentados por organismos nacionales e internacionales. Además les permiten a las instituciones que los utilicen, mantenerse informadas sobre todos los adelantos tecnológicos y científicos que acontecen en el mundo sobre un tema específico en este caso con todo lo referente a gobierno electrónico. Según un estudio denominado: "Gobierno electrónico en Chile, un modelo para América Latina", del Observatorio Internacional de Gobierno Electrónico, cuya idea es medir el real estado de avance del gobierno electrónico en Chile para compararlo luego con otros países de la región, realizado por un grupo de expertos, donde se obtuvo como resultado que el 90% de los sitios de gobierno tienen un nivel casi excelente por la rapidez y lo amigable del sitio; por lo que los expertos declararon en su informe como recomendación oficial que cada país debe contar con un servicio como éste. (Chile, 2006)

Luego de un estudio sobre los principales conceptos relacionados con los observatorios de gobierno electrónico se decide trazar los siguientes objetivos para el desarrollo del presente trabajo:

Realizar estudios independientes desde varias perspectivas.

- Estudiar la prestación de servicios administrativos a ciudadanos, empresas y otras administraciones.
- Comprobar el respeto a estándares técnicos y las normas, procedimientos y principios recogidos en las leyes (protección de datos, seguridad en las comunicaciones, etc.).

Las posibles focalizaciones temáticas de los observatorios son muy variadas y, a nivel mundial, incluyen la investigación en temas de elecciones, convivencia entre el poder político y los intereses de las corporaciones mediáticas, crimen y violencia, derechos humanos, salud, intimidad, comunicación local, formas de trabajo de los periodistas y condiciones de producción de los mensajes, cibersociedad y nuevas tecnologías de la información, políticas de comunicación, propiedades de los medios y aspectos empresariales de los medios de comunicación, etc. A continuación se muestran algunos de los observatorios estudiados:

Observatorio de Competitividad de la República Dominicana: Monitorear y dar seguimiento a los avances del país en el marco del Plan Nacional de Competitividad Sistémica. Apoya la formulación de políticas públicas en materia de competitividad, promoviendo la toma de decisiones más acertadas.

Observatorio de Gobierno electrónico en Perú: El observatorio de Gobierno electrónico en el Perú es un canal que permitirá: Dar seguimiento a las políticas e informar a los ciudadanos sobre los avances referentes al gobierno electrónico en ese país.

Observatorio de Prospectiva Científica y Tecnológica de Argentina: Relevar el desarrollo de la Sociedad de la Información y el impacto de las Tecnologías de la Información y la Comunicación en dicho país.

Observatorio de Gobierno electrónico de Jalisco: Este observatorio permitirá conocer todo los avances en cuanto a gobierno electrónico se realice en Jalisco, en cada uno de los municipios encontrarás la forma y características de gobierno electrónico, así como información tecnológica en cada uno de ellos.

El Observatorio Cubano de Ciencia y Tecnología: tiene como misión analizar y evaluar las perspectivas de los temas estratégicos del desarrollo de la ciencia y la innovación tecnológica en Cuba y su relación con las prioridades del desarrollo económico, social y medioambiental nacional.

El Observatorio Nacional de Ciencia, Tecnología e Innovación (ONCTI): Tiene entre sus competencias, generar los indicadores y estadísticas del Sistema Nacional de Ciencia, Tecnología e Innovación (SNCTI) y promover el monitoreo científico y tecnológico.

Teniendo en cuenta los grandes beneficios que ofrecen los Observatorios, se realizó un estudio de las características de algunos de los que existen actualmente. Se tomó la decisión de no usar ninguno de los anteriormente planteados debido a que los servicios que ofrecen están orientados hacia las necesidades

específicas de las propias entidades y países que los sustentan. Se requiere de un observatorio mediante el cual se puedan realizar diferentes estudios concernientes al desarrollo del gobierno electrónico, que brinde la posibilidad de estudiar las tendencias relacionadas con el tema, para a la hora de implantar algún servicio tener una base según los países que mejores resultados han tenido en la utilización de los mismos y por ello se decide utilizar la Guía web para la creación de Observatorios de gobiernos electrónico de Chile en su versión 2.0, pues la misma facilita desarrollar un sitio con las características propias del observatorio de gobierno electrónico en Cuba siguiendo las normas , leyes , y estándares existentes.

1.1.2 Estándares

Los estándares se definen como las especificaciones que determinan la manera en que se construye y funciona a una tecnología en particular, con el objetivo de regular la realización de sus procesos; también se conoce de esta manera a la forma en que se construyen elementos de hardware o software, para que quienes generen elementos adicionales a estos logren realizar dicha tarea correctamente y consigan que esos nuevos elementos se acoplen a los anteriores sin problemas. Los estándares determinan la forma de construir páginas y componentes, ya que se engloba bajo este nombre al conjunto de normas que dan origen al lenguaje en el que se escriben las páginas del sitio. En el caso de la web los estándares son fijados por el World Wide Web Consortium (W3C), una organización internacional que agrupa a más de 400 entidades miembros entre las cuales se cuentan empresas, universidades, medios de comunicación, fundaciones y centros de investigación.

1.1.2.1 Estándares utilizados

Disponibilidad de Dominio GOB y GOV: permite revisar si el sitio web del organismo ha sido inscrito en el servidor de dominios de Gobierno (NIC) del Ministerio del Interior; esta verificación ayuda al cumplimiento del Artículo 3 del Decreto Supremo 100/2006 del Ministerio Secretaría General de la Presidencia. Es importante considerar que para que se pueda utilizar esta herramienta, la consulta debe realizarse desde un computador ubicado dentro de la red de Gobierno.

Verificación de HTML: permite revisar el cumplimiento del estándar de la versión de HTML o XHTML que se haya elegido; la herramienta compara el código de la página web que se revisa contra la norma correspondiente y da a conocer cuáles son las infracciones que se han cometido (en caso de existir) y además, ofrece información acerca de cómo resolver el problema. Esta verificación es exigida en el Artículo 5 del Decreto Supremo 100/2006 del Ministerio Secretaría General de la Presidencia.

Verificación de enlaces rotos: permite revisar que no haya enlaces rotos o imágenes perdidas en el sitio web, siguiendo la recomendación establecida en el Artículo 5 del Decreto Supremo 100/2006 del Ministerio

Secretaría General de la Presidencia; la herramienta entrega un listado de los problemas detectados para que el encargado del sitio web realice las correcciones correspondientes.

1.2 Metodología a utilizar para el desarrollo del software

Una metodología para el desarrollo de un proceso de software es un conjunto de métodos y técnicas, procedimientos, reglas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas informáticos. Por ello hay que tener mucho cuidado a la hora de seleccionar la metodología que llevará a cabo todo el desarrollo, debido a que un error en esta selección puede traer consigo clientes insatisfechos y desarrolladores más insatisfechos. (Samira, 2008)

1.2.1 Programación Extrema (XP)

Programación extrema, del inglés Extreme Programming (XP), es una de las conocidas metodologías de desarrollo de software ágiles. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. (Cortizo Pérez, y otros, 2004)

Esta metodología ha sido diseñada para solucionar el eterno problema del desarrollo de software por encargo: entregar el resultado que el cliente necesita a tiempo.

Características de XP:

- *Pruebas Unitarias:* Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- *Re fabricación:* Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- *Programación en pares:* Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo fundamental de XP es:

- **La comunicación** entre los usuarios y los desarrolladores, la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

- **Críticas a XP**

- No escalable y con altos riesgos si existen fallas en la arquitectura.
- Altos riesgos si no hay capacidad/estabilidad en las personas.
- La programación de a pares es un intento por solventar la falta de análisis.
- Puede caer en el modelo de codificar y probar.
- Vagas nociones de aseguramiento de la calidad.(Cortizo Pérez, y otros, 2004)

1.2.2 Plataforma de Solución de Microsoft (MSF)

MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. (MSF) Es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información. (Torres, 2006)

Características de MSF:(Torres, 2006)

- *Adaptable:* es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- *Escalable:* puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- *Flexible:* es utilizada en el ambiente de desarrollo de cualquier cliente.
- *Tecnología Agnóstica:* porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

Ventajas del método MSF

- Tiene facilidad de soporte y mantenimiento.
- Efectúa trabajo en equipo y la colaboración.

- Se puede utilizar para proyectos de cualquier magnitud.
- SF ayuda a implantar soluciones en base a la tecnología.

Desventajas del método MSF

- Este modelo utiliza demasiada documentación en sus fases.
- El análisis de riesgo suele llevar mucho tiempo frenando el avance del proyecto.
- Al ser un modelo de Microsoft implica que se tiene que utilizar herramientas solo de Microsoft.

1.2.3 El Proceso Unificado de Desarrollo de Software (RUP)

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, es un producto de Rational Software Corporation (IBM). (Jacobson, y otros, 2000)

Tiene como objetivo asignar tareas y responsabilidades para producir software de alta calidad, buscando satisfacer las necesidades de los clientes, ajustándose al presupuesto y a los tiempos estimados. RUP ofrece diferentes subprocesos, que brindan un marco de trabajo adaptable y extensible a las necesidades de cada organización. Se caracteriza por ser:(Jacobson, y otros, 2000)

- Iterativo e incremental: Cada fase se desarrolla en iteraciones. Se divide el trabajo en partes más pequeñas o mini-proyectos, donde cada uno es una iteración que resulta en un incremento.
- Centrado en la arquitectura: La arquitectura describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Guiado por los casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos.

Cada una de estas iteraciones debe ser clasificada y ordenada según su prioridad, y se convierte en un grupo de entregables al cliente. Trae como beneficio la retroalimentación en cada iteración.

RUP tiene bien definido sus procesos orientados a objetos, puede ser menos pesado si se es capaz de aceptar y adaptar a las condiciones esperadas de cada proyecto. Además de traer consigo un grupo de beneficios como la estandarización, facilita el entendimiento por parte de los usuarios del software, que no necesariamente deben tener conocimientos previos sobre el tema y el desarrollo del producto a distancia de

los clientes, utiliza el Lenguaje de Unificado de Modelado (UML) para describir todo el proceso.(Jacobson, y otros, 2000)

RUP define nueve disciplinas a realizar en cada fase del proyecto:

- Modelado del negocio
- Análisis de requisitos
- Análisis y diseño
- Implementación
- Test
- Distribución
- Administración del proyecto
- Gestión de configuración y cambios Gestión del proyecto.
- Ambiente

1.2.4 Justificación de la selección de la Metodología de Desarrollo a utilizar.

Para la selección de la metodología a utilizar, se realizó un estudio de algunas de las existentes en el mundo teniendo en cuenta, características, clasificaciones (Ágil o Tradicional) beneficios y considerando que se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar un software con calidad y a tiempo se decide RUP como proceso rector porque el objetivo del presente trabajo diploma es realizar el análisis y diseño , lo cual es un flujo de trabajo desarrollado por RUP para lograr una mejor calidad del software, además abarca todas las prácticas de la industria, configurable para proyectos pequeños, permitiendo seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto.

1.3 Lenguaje de Modelado

Hoy en día para el desarrollo de un software es necesario contar con un plan bien organizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar

cambios si no se han captado claramente sus necesidades. Para ello todo el proceso de software debe representarse de manera sencilla, que permita a los usuarios entender su estructura, contenido y organización. La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. Jacobson plantea que los objetivos de los mismos son:(Jacobson, y otros, 2000)

- Captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento.
- Pensar en el diseño de un sistema.
- Capturar decisiones del diseño a partir de los requisitos.
- Generar productos aprovechables para el trabajo.
- Organizar, encontrar, filtrar, recuperar, examinar, y corregir la información en grandes sistemas. Explorar económicamente múltiples soluciones.
- Domesticar los sistemas complejos.

Se utiliza el Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language), porque es el lenguaje utilizado por RUP para describir el proceso de desarrollo del software, además es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de mismo.

1.4 Herramientas Case

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), son aplicaciones que facilitan el desarrollo de software, reduciendo el esfuerzo, el costo y el tiempo, además de estructurar la documentación asociada a los artefactos generados. (Kendall, 1997)

Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Además permiten aumentar las capacidades y habilidades de los analistas, elementos importantes del proceso de desarrollo, pero no las reemplazan.

En el mundo informático se han creado varias herramientas para el desarrollo de la ingeniería de software con el objetivo de desarrollar programas, a continuación se mencionan algunas de las herramientas CASE más utilizadas en la actualidad con sus principales características.

1.4.1 Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor costo, además de permitir el dibujo de todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales con demostraciones interactivas y proyectos. (Boost Productivity with Innovative and Intuitive Technologies, 2007)

Seguidamente aparecen las razones por las cuales una organización debe elegir VPUML:

- Permite el modelado de base de datos.
- Permite la confección de los prototipos de interfaz.
- Integración perfecta con los principales IDEs que soportan Java, como Microsoft Visual Studio, JBuilder, Eclipse, NetBeans.
- Completa integración con Microsoft Office.
- Herramientas UML más fáciles de usar, ofrece una edición de líneas para diagramas UML, puede crear y especificar un modelo de elementos sin cuadro de diálogos, entre otras funcionalidades.
- Diseño centrado en casos de uso que genera un software de mayor calidad.
- Soporta múltiples plataformas: no importa el Sistema Operativo que este en uso.
- Análisis textual para la identificación de clases candidatas. El análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas.
- Conversión instantánea de código fuente, archivos ejecutables y binarios en modelos: permite invertir programas de código fuente, archivos ejecutables, binarios y modelos UML de inmediato. Además de

idiomas y formatos que incluyen XML, esquema XML, .NET dll o exe, Java de fuente/class/jar, origen de archivo C++ y archivos fuente CORBA IDL.

- Diseño automático de diagramas.

1.4.2 Rational Rose Enterprise Edition 2003

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases entregables. (RATIONAL)

Aplicación para desarrollo de arquitectura base del Software. Ideal para Analistas que utilizan el Proceso Unificado de Desarrollo con herramientas UML (Lenguaje Unificado de Modelado). Rational Rose es una herramienta de desarrollo basada en modelos que se integra con las bases de datos y los IDE de las principales plataformas del sector. IBM Rational Rose Enterprise es uno de los productos más completos de la familia Rational Rose. Todos los productos de Rational Rose dan soporte a Unified Modeling Language (UML), pero no son compatibles con las mismas tecnologías de implementación. Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Al igual que todos los productos de Rational Rose, ofrece un lenguaje de modelado común que agiliza la creación del software.

Dentro de las funcionalidades que soporta el Rational Rose están:

- Diagramas UML.
- Desarrollo Orientado al Modelado.
- La generación de código ADA, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Capacidad de crear definiciones de tipo de documento XML para el uso en la aplicación.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.
- Se integra con herramientas Integrated Development Environment (IDE) como:
- Borland JBuilder versiones 7.0 a 10.0

- Sun Forte for Java Community y Enterprise Editions
- Microsoft Visual Studio 6
- Microsoft Visual Studio 2003
- Microsoft Visual Studio 2005
- Wind River Tornado
- Green Hills MULTI.

1.4.3 Enterprise Architect

Es una herramienta CASE orientada a objetos que provee su alcance para el desarrollo de sistemas, administración de proyectos y análisis de negocios. Maneja totalmente el ciclo de vida de desarrollo de software, utilizando el UML como leguaje de modelado. Facilita y soporta el levantamiento de requerimientos, el análisis y diseño, las pruebas y mantenimiento del software en desarrollo. Soporta un impresionante rango de lenguajes de desarrollo, incluyendo ActionScript, C, C++, C# y VB.NET, Java, Visual Basic 6, Python, PHP, XSD, WSDL y otros más. Gestiona la ingeniería de código, normal e inversa, además de una efectiva documentación compatible con Microsoft Word.(SPARX System Pty L, 2000-2007)

Dentro de las funcionalidades que soporta el Enterprise Architect están:

- Diagramas UML.
- CU, Modelos Lógico, Dinámico y Físico.
- Extensiones personalizadas para modelado de procesos.
- Documentación de alta calidad compatible con MS Word.
- Modelado de Datos.
- Ingeniería directa de Base de Datos a DDL e ingeniería inversa de Base de Datos desde ODBC.
- Soporte de pruebas.
- Multi-usuario, con sistema de seguridad.

Se integra a través de plug-ins con herramientas como:

- Eclipse.
- Visual Estudio.Net.

1.4.4 Justificación de la selección de la herramienta CASE

Partiendo del estudio realizado a las principales herramientas CASE existentes, se decidió utilizar Visual Paradigm. Se hace esta elección, principalmente, por ser una herramienta multiplataforma, permite exportar documentos, es robusta y de fácil uso, cuenta con un entorno de creación de diagramas para UML. Permite dibujar todos los tipos de diagramas de clases, crear código o desde diagramas y generar documentación, además en la universidad se posee un amplio conocimiento acerca de la misma.

1.5 Lenguajes de Programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. (Wilson, 1993)

Dentro de los lenguajes de programación más usados en el desarrollo de aplicaciones web se pueden encontrar: PHP, Java Script, C#, PERL, entre otros.

1.5.1 PHP

Preprocesador de Hipertexto o PHP (por sus siglas en inglés) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones web, que es interpretado del lado del servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema, es cómodo obtener documentación para darle solución de forma rápida y sin costo alguno. ((Grupo de documentación de PHP.. , 2001).

Algunas de sus características:

- Seguridad: permite la protección contra diversos ataques a través de diferentes niveles de seguridad.
- Estabilidad: Posee un sofisticado manejo de variables que lo hacen muy robusto y estable.

- Multiplataforma: Puede ejecutarse en cualquier plataforma (Linux, Unix, Solaris y Mac OS, Windows, etc).

- Velocidad: Alta velocidad de ejecución sin introducir demoras en la máquina, bajo consumo de recursos y muy buena integración con Apache.

Ventajas:

- Posee una de las comunidades más grandes en Internet, por lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.

- Al ser Opens urce, el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no está forzado a pagar actualizaciones anuales para tener una versión que funcione.

Se ha decidido PHP como lenguaje de programación, debido que se utilizara el CMS joomla y están desarrollados en este lenguaje pues en cuanto a la seguridad, es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor web en forma de módulo o ejecutado como un binario CGI separado, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Estas propiedades hacen que cualquier cosa que sea ejecutada en un servidor web sea insegura por naturaleza y PHP está diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI que Perl o C.

1.6 Sistemas de Gestión de Contenidos

Un sistema de gestión de contenidos (en inglés Content Management System, abreviado CMS) es un programa que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas web. Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio, permitiendo manejar de manera independiente el contenido y el diseño. Genera páginas dinámicas interactuando con el servidor para generar la página web a petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor. Así, es posible manejar el contenido y proporcionarle en cualquier momento un diseño distinto, sin tener que formatear el contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Esto permite gestionar, bajo un buen formato, la información del servidor, reduciendo el tamaño de las páginas para descarga y minimizando el costo de gestión del portal con respecto a una página estática, en la que cada cambio de

diseño debe ser realizado en todas las páginas, de la misma forma que cada vez que se agrega contenido tiene que dar formato a una nueva página HTML y subirla al servidor.

Realizar el portal del Observatorio de Gobierno Electrónico para el país con la ayuda de un Sistema Gestor de Contenidos reportaría grandes beneficios tanto a los desarrolladores como a los usuarios finales del sitio. Esto facilita el acceso a la publicación de contenidos a un rango mayor de usuarios. Permite que sin conocimientos de programación, ni maquetación, cualquier desarrollador pueda indexar contenido en el portal. Además permite la gestión dinámica de usuarios y permisos, la colaboración de varios implicados en el mismo trabajo y la interacción mediante herramientas de comunicación. Además, los gestores de contenidos garantizan que la aplicación a construir sea escalable y tenga una interfaz amigable y sencilla.

Tipos de Gestores de Contenidos (CMS) (Franco, 2008)

Los gestores de contenido se pueden segmentar según diferentes criterios:

Según el lenguaje de programación empleado: Active Server Pages, Java, PHP, Ruby on Rails, Python.

Según la propiedad del código:

Código abierto; permite que se desarrolle sobre el código.

Código propietario; sólo su desarrollador puede desarrollar la aplicación.

Según el tipo de uso o funcionalidades:

Genéricos*:* Ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Pueden servir para construir soluciones de gestión de contenidos, de comercio electrónico, blogs, portales, etc. Ejemplos: Zope, OpenCMS, Typo3, Apache Lenya.

Foros**:** sitio que permite la discusión en línea donde los usuarios pueden reunirse y discutir temas en los que están interesados. Ejemplos: phpBB, MyBB.

Blogs**:** Publicación de noticias o artículos en orden cronológico con espacio para comentarios y discusión. Ejemplos: Wordpress, Typo.

Wikis**:** Sitio web dónde todos los usuarios pueden colaborar en los artículos, aportando información o reescribiéndola. También permite espacio para discusiones. Indicado para material que irá evolucionando con el tiempo. Ejemplos: Mediawiki, Tikiwiki.

eCommerce: Son Sitios web para comercio electrónico.

Portal**:** Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad. Ejemplos: PHPNuke, Drupal, Plone.

Galería**:** Permite administrar y generar automáticamente un portal o sitio web que muestra contenido audiovisual, generalmente, imágenes. Ejemplo: Gallery.

e-Learning**:** Sirve para la enseñanza de conocimientos. Los usuarios son los profesores y estudiantes, existen aulas virtuales donde se ponen a disposición el material del curso. La publicación de un contenido por un profesor es la puesta a disposición de los estudiantes, en un aula virtual, de ese contenido. Ejemplo: Moodle.

Publicaciones digitales**:** son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc. Ejemplo: ePrints.

Se puede hacer una división de los CMS según el tipo de licencia escogido. Por una parte están los CMS comercializados por empresas que consideran mantener el código fuente en propiedad. Por la otra están los de código fuente abierto, desarrollados por individuos, grupos o empresas que permiten el acceso libre y la modificación del código fuente. Se podría considerar que la herramienta comercial será más estable y coherente al estar desarrollada por un mismo grupo. En la práctica esta ventaja no es tan grande, pues los CMS de código abierto también están coordinados por un único grupo o por empresas, de forma similar a los comerciales. Utilizar una herramienta de gestión de contenidos de código abierto tiene otras ventajas que hace decidirse a la mayoría de usuarios: Habitualmente todo el software de código abierto es de acceso libre, es decir, sin ningún costo en licencias. Sólo en casos aislados se hacen distinciones entre empresas y entidades sin ánimo de lucro o particulares. En comparación, los productos comerciales pueden llegar a tener un costo que sólo una gran empresa puede asumir. En cuanto al soporte, los CMS comerciales acostumbran a dar soporte profesional, con un costo elevado en muchos casos, mientras que los de código abierto se basan más en las comunidades de usuarios que comparten información y solución a los problemas.

En el mercado hay CMS de calidad tanto comerciales como de código abierto. Muchos CMS de código abierto están poco elaborados aunque en plena evolución. En definitiva, un buen CMS de código abierto es mucho más económico que su homólogo comercial, con la ventaja de disponer de todo el código fuente y de una extensa comunidad de usuarios.

En resumen, los CMS basados en código abierto son los más conocidos, por su facilidad y bajo costo en la instalación. Es suficiente en la mayoría de los casos con un servidor Apache que pueda leer código PHP y una base de datos MySQL**.** A continuación aparece una relación de los CMS de código abierto más utilizados.

1.6.1 Sistema de Gestión de Contenidos WordPress

Wordpress es un Sistema de Gestión de Contenidos de gran sencillez de uso hasta el punto donde solo puede usarse para publicar artículos, por lo que es muy utilizado para la creación de Blog. Aclarar que Wordpress (núcleo) no es un CMS propiamente dicho, aunque puede evolucionar hasta llegar a serlo, y también puede ser extensible utilizando plugins adicionales. Entre las funcionalidades que brinda el sistema está la de generar un archivo cronológico automáticamente, y posee un calendario, en los cuales es posible navegar y buscar información publicada en cualquier día, mes o año. Para su desarrollo utiliza PHP y MySQL y es uno de los CMS más populares junto a Joomla y Drupal. Debido a que los blogs son los sistemas más susceptibles de recibir Spam2, Wordpress posee un plugin llamado "Akismet" que identifica y frena la mayoría de los comentarios y trackbacks que son spam. (Muras, 2009)

Wordpress presenta algunas desventajas, por ejemplo: no puede modificar el código de su sitio fácilmente. Dependiendo de la cantidad y frecuencia de las publicaciones la administración de Blog puede dificultarse, si no se hace revisión periódica. Por ser un espacio de acceso público pueden recibirse comentarios no deseados que no se encuentren relacionados con las temáticas. El acceso a Blog debe hacerse, necesariamente utilizando un navegador de Internet. Si no se conoce con certeza la dirección del Blog, la búsqueda se torna difícil. Carece de muchas de las funciones de redes sociales, comercio electrónico, foros, wikis, etc.

1.6.2 Sistema de Gestión de Contenidos Drupal

Drupal es un sistema de gestión de contenido modular y muy configurable. Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Se destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la Web, y un énfasis especial en la usabilidad y consistencia de todo el sistema. El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hacen que sea adecuado para realizar diferentes tipos de sitio web. El mismo ha sido diseñado con una arquitectura modular, y gracias a esta se puede agregar nuevas funciones ya sea la creación o el incremento de funcionalidades a un modulo sin afectar los ya creados. Estos módulos son los encargados del funcionamiento del sistema y separan la interfaz gráfica de la información. Este gestor de contenido posee una capa de abstracción de base de datos, implementada y soportada para MySQL y PostgresSQL, aunque puede añadir soporte para varias bases de datos. Está

estructurado por temas, los cuales se pueden descargar de Internet o simplemente crearlos como plantillas en PHP, HTML y CSS. La desventaja que presenta Drupal es la alta curva de aprendizaje y la gran cantidad de módulos, hace difícil la selección del más eficiente

1.6.3 Sistema de Gestión de Contenidos Joomla

Es uno de los sistemas gestores de contenidos de código abierto más popular que permite crear sitios web de alta interactividad, profesionalidad y eficiencia. La administración de Joomla está enteramente basada en la gestión online de contenidos. Se dice "gestión online" porque todas las acciones que realizan los administradores de sitios en Joomla, ya sea para modificar, agregar, o eliminar contenidos se realizan exclusivamente mediante un navegador web (browser) conectado a Internet, es decir, a través del protocolo HTTP (Protocolo de Transferencia de Hipertexto).

Tiene una gran comunidad de usuarios y también toda la documentación para crear diferentes aplicaciones. Además, es posible realizar casi cualquier sitio con muy pocos conocimientos: comercio electrónico, revistas online, intranets, redes sociales, etc. Realiza un gran trabajo gestionando el contenido necesario para que un Sitio Web funcione. Para muchas personas, el verdadero potencial de dicho CMS reside en la arquitectura de la aplicación, que posibilita que miles de desarrolladores en el mundo puedan crear potentes add-ons y extensiones disponibles como : gestores de documentos, generadores de formularios dinámicos, galerías de imágenes multimedia, motores de comercio y venta electrónica, directorios de empresas u organizaciones, software de foros y chats, calendarios, software para blogs, servicios de directorio, boletines de noticias, herramientas de registro de datos, sistemas de publicación de anuncios, servicios de suscripción. Está programado en lenguaje PHP y SQL. Utiliza bases de datos relacionales, más específicamente MySQL y al ser una aplicación Web, funciona obviamente en servidores de páginas web (HTTP Servers).

Características de publicación de páginas web en Joomla (Joomlaos.net., 2009)**:**

Automatización en la publicación: Las páginas y documentos de Joomla pueden programarse con fecha de publicación y fecha de caducidad.

Archivo e historial: Las páginas viejas o publicaciones que hayan perdido vigencia pueden enviarse a un "archivo" de almacenamiento, sin necesidad de tener que borrarlas. Esto permite también dar la posibilidad a los navegantes de consultar artículos viejos o documentos anteriores en un historial.

Formatos de lectura: Cada documento es generado automáticamente por Joomla en formato **PDF** y en **XML**. **Envío por E-mail**: Los usuarios del sitio Joomla podrán enviar automáticamente a un amigo por correo cada documento publicado.

Valoración de contenidos: Los visitantes del sitio podrán votar por la calidad de lo publicado. **Comentarios**: Los usuarios podrán comentar sus opiniones o expresar sus inquietudes en la misma página de contenidos. **Ventajas de Joomla:** (Joomlaos.net., 2009)

Organización del sitio web: Joomla está preparado para organizar eficientemente los contenidos de su sitio en secciones y categorías, lo que facilita la navegabilidad para los usuarios y permite crear una estructura sólida, ordenada y sencilla para los administradores. Desde el panel administrador de Joomla usted podrá crear, editar y borrar las secciones y categorías de su sitio de la manera en que más le convenga.

Publicación de contenidos: Con Joomla CMS podrá crear páginas ilimitadas y editarlas desde un sencillo editor que permite formatear los textos con los estilos e imágenes deseados. Los contenidos son totalmente editables y modificables.

Escalabilidad e implementación de nuevas funcionalidades: Joomla ofrece la posibilidad de instalar, desinstalar, administrar componentes y módulos, que agregarán servicios de valor a los visitantes de su sitio web, por ejemplo: galerías de imágenes, foros, boletines, clasificados, etc.

Administración de usuarios: Joomla le permite almacenar datos de usuarios registrados y también la posibilidad de enviar e-mails masivos a todos los usuarios. La gestión de usuarios es jerárquica, y los distintos grupos de usuarios poseen diferentes niveles de permisos dentro de la gestión y administración del sitio.

Diseño y aspecto estético del sitio: Es posible cambiar todo el aspecto del sitio web tan solo con un par de clic, gracias al sistema de plantillas que utiliza Joomla.

Navegación y menú: Totalmente editables desde el panel administrador de Joomla.

Administrador de imágenes: posee una utilidad para subir imágenes al servidor y usarlas en todo el sitio.

Disposición de módulos modificable: la posición de módulos puede acomodarse como se prefiera.

Encuestas: posee un sistema de votaciones y encuestas dinámicas con resultados en barras porcentuales.

Feed de noticias: trae incorporado un sistema de sindicación de noticias por RSS/XMS de generación automática.

Publicidad: Es posible hacer publicidad en el sitio usando el Administrador de Banners.

Estadísticas de visitas: con información de navegador, sistemas operativos y detalles de los documentos (páginas) más vistos.

1.6.4 Consideraciones de los Sistemas de Gestión de Contenidos

El CMS que se vaya a seleccionar tiene que ser de código fuente abierto (o libre), fiable, robusto y permitir la escalabilidad para adecuarse a futuras necesidades con módulos. La herramienta tiene que tener soporte tanto por parte de los creadores como por otros desarrolladores, tiene que ser fácil de utilizar y aprender. Una vez aclarado los requerimientos que tendría que poder satisfacer el CMS a escoger, se concluye que: WordPress no es el sistema adecuado para usar para el desarrollo de la solución informática, pues sus funcionalidades están encaminadas a la construcción de blogs además de presentar varias desventajas, las cuales fueron expuestas anteriormente. Para desarrollar la aplicación en el menor tiempo posible lo ideal es usar Joomla, aunque Drupal brinda tantos beneficios como Joomla, la decisión de rechazar a Drupal está basada en que Joomla es más fácil de configurar y poner en marcha que su competencia. El panel de control de la administración de Drupal es malo; la separación entre el "front-end y el backend" es débil y confusa, Joomla es mucho mejor. En cuanto a las plantillas, Joomla gana por un amplio margen, Drupal todavía tiene una enorme rampa que subir para conseguir que un Web de noticias, una revista, etc. se vea atractiva, característica esta, indispensable para el sistema en construcción.

1.7 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado o IDE (acrónimo en inglés de Integrated Development Environment), es un compilador, editor de código, depurador y constructor de interfaz gráfica .Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes que proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

1.7.1 Entorno de Desarrollo Integrado NetBeans 6.8

En su núcleo, NetBeans IDE es una herramienta de desarrollo Java, escrita puramente sobre la base de la tecnología Java. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Este enfoque de bienes comunes creativos ha permitido una mayor capacidad de uso, con cada nueva versión, y ha proporcionado a los desarrolladores mayor flexibilidad, al modificar el IDE, si así lo desean.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. 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. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación.

NetBeans es un Entorno de Desarrollo Integrado de código abierto y gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. NetBeans IDE es fácil de instalar y listo para usar, se puede ejecutar tanto en Windows, Linux, Mac OS X como en Solaris. (Sun Microsystems, 2009)

Entre las características de la plataforma están: (Dadiskod, 2010)

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.
- Framework basado en asistentes (diálogo paso a paso).

NetBeans IDE 6.8 incluye nuevas funcionalidades: (Sun Microsystems, 2009)

Mayor soporte PHP: amplía el soporte de lenguajes dinámicos con soporte para PHP 5.3 y el Framework Symfony.

Mayor integración con Proyecto Kenai: un entorno colaborativo para promover el desarrollo de proyectos de código abierto y libre en la red, ofrece soporte para JIRA5, mensajería instantánea y seguimiento de incidencias mejorados.

Mejor parametrización C/C ++: Parametrizar y afinar aplicaciones C/C++ con el nuevo indicador Microstate Accounting, monitor de actividad I/O.

1.7.2 Entorno de Desarrollo Integrado Zend Studio for Eclipse 6.0

Zend Studio for Eclipse 6.0 es un IDE que soporta varios lenguajes aunque fue construido especialmente para PHP. Los expertos en PHP consideran a Zend Studio como el entorno más maduro y con más características útiles. Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez, versiones del producto para Windows, Linux y MacOS. Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Además, permite hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (Álvarez, 2003)

1.7.3 Consideraciones de los Entornos de Desarrollo Integrado

Luego de haber analizado algunos de los entornos de desarrollo de la programación web se ha concluido que el más conveniente para utilizar para el desarrollo de la solución informática es el NetBeans IDE 6.9**,** ya que es una herramienta para programadores pensada para 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 Entorno de Desarrollo Integrado (IDE) NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso, adquirirlo reportaría muchísimos beneficios

debido a que se puede integrar con framework de PHP como Symfony, el editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP, brinda un entorno amigable para ejecutar comandos de Symfony, es posible crear test con PHP Unit, para diferentes funciones, luego realizar la comprobación y ver todos los resultados. En las propiedades PHP Unit puede definir una configuración personalizada de archivos XML, un archivo de arranque para las opciones de línea de comandos, una serie de pruebas a medida, o puede que el IDE genere el código esqueleto para usted. A pesar de los grandes beneficios que presenta Zend Studio se descarta la posibilidad de usarlo por ser de código propietario.

1.7.4 Framework

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Existen varios framework tales como: CodeIgniter, ZooP Framework, WACT, Seagull Framework, CakePHP, Zend Framework, Symfony, Prado, PHP on TRAX y eZComponents.

Symfony

Se selecciona Symfony, ya que es un framework que es diseñado para optimizar el desarrollo de las aplicaciones web, ha sido creado para obtener el máximo rendimiento de PHP, permite controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS y CSRF, se publica bajo licencia MIT, con la que puedes desarrollar aplicaciones web comerciales, gratuitas, es infinitamente escalable si se disponen de los recursos necesarios. Es multiplataforma y sigue una política de tipo LTS (*longtermsupport*), por la que las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de errores.

1.8 Sistemas Gestores de Base de Datos

Un sistema gestor de bases de datos (SGBD) es un conjunto de programas transparentes al usuario final que se encarga de la integridad y seguridad de los datos así como de la interacción con el Sistema Operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales, controlando todas las operaciones que el usuario hace contra la base de datos.

El gestor almacena una descripción de datos en lo que le llaman: “diccionario de datos”, así como los usuarios permitidos y los permisos. Asimismo, tiene que haber un usuario administrador que se encargue de centralizar todas estas tareas. En cuanto al diccionario de datos, no es más que una base de datos donde se guardan todas las propiedades de la base de datos, descripción de la estructura, relaciones entre los datos, etc.

Los sistemas gestores de bases de datos deben permitir definir (especificar tipos, estructuras y restricciones de datos), construir (guardar los datos en algún medio controlado por el mismo SGBD) y manipular (realizar consultas, actualizarla, generar informes) a una base de datos. Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

1.8.1. Sistema Gestor de Base Datos Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es considerado como uno de los sistemas de bases de datos más completos, destacando su: Soporte de transacciones, Estabilidad, Escalabilidad, Soporte multiplataforma. Oracle Corporation es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión. Cuenta además, con herramientas propias de desarrollo para realizar potentes aplicaciones, como Oracle Designer y Oracle JDeveloper.

Ventajas:

- Puede ejecutarse en todas las plataformas, desde una PC hasta un supercomputador.
- Oracle soporta todas las funciones que se esperan de un servidor "serio": un lenguaje de diseño de bases de datos muy completo (PL/SQL) que permite implementar diseños “activos", con triggers y procedimientos almacenados, con una integridad referencial declarativa bastante potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- Oracle es la base de datos con mas orientación hacía INTERNET.

1.8.2. Sistema Gestor de Base Datos PostgreSQL

Es un sistema gestor de bases de datos relacionales, soporta casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl, Tcl yPython) .Es considerado como una de las alternativas de sistema de base de datos de código abierto. Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, pero no es un sistema de gestión de bases de datos puramente orientado a objetos.Tiene como ventajas que posee Instalación ilimitada, soporte, ahorros considerables en costos de operaciones, estabilidad y confidencialidad legendariases extensible, multiplataforma, diseñado para ambientes de alto volumen, además de poseer herramientas graficas para gestión de BD de diseño y administración de BD.

1.8.3 Sistema Gestor de Base Datos MySQL

MySQL AB es una compañía de software fundada por David Axmark, Allan Larsson y Michael Widenius en 1995, creadora del sistema administrador de bases de datos relacionales MySQL, y una de las más grandes empresas de software libre del mundo. Desde el 16 de enero de 2008 es una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009.

La base de datos de código abierto MySQL es la más popular del mundo, debido a su gran rendimiento, alta fiabilidad, facilidad de uso y ahorro de costos, pues se distribuye bajo doble licencia. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Este esquema de licenciamiento dual posibilita que la compañía, además de la venta de licencias privativas, ofrezca soporte y servicios.

MySQL 5.1 es el sistema de bases de datos enfocadas a servidores web y es una de las versiones más estables y utilizadas. Es una base de datos relacional, multihilo y multiusuario que se ha estandarizado al ser el soporte de la gran mayoría de CMS junto con el conocido lenguaje PHP.

A pesar de que MySQL 5 es un proyecto básicamente Open Source, la propiedad real es de Sun Microsystems, a su vez propiedad de Oracle. Sin embargo, el núcleo de MySQL 5 es libre y puede ser utilizado en cualquier proyecto sin ningún problema. Estas empresas ofrecen soporte y fomentan el desarrollo de este estupendo sistema de gestión de bases de datos (SGDB).

Características de MySQL (Saha, 2008)**:**

Velocidad: Arquitectura Multi-hilos, múltiples clientes tienen acceso concurrente. Cachea el resultado de las consultas comunes. **Facilidad de uso:** Interfaz de línea de comandos y herramientas gráficas – escritorios, basadas en Web.

Soporte Multi-usuario: Múltiples usuarios tienen acceso concurrente a una o más bases de datos simultáneamente. Sistema de privilegios de usuarios potente y flexible. Esquemas de autenticación basados en usuario – máquina.

Portabilidad: Funciona en diferentes plataformas: Linux, Solaris, Windows, etc.

Amplio Soporte de aplicaciones: Base de datos de aplicaciones para Escritorio y la Web. APIs (Application Programming Interface, o Interfaz de Programación de Aplicaciones) para: C/C++, Java, PHP, Perl, Python, Ruby.

Programas almacenados: Procedimientos, funciones almacenadas y disparadores (triggers).

Motor de almacenamiento: Escribe los datos en almacenes persistente. Motores de almacenamiento conectable, permite que el mismo motor sea cargado y/o cambiado dinámicamente en tiempo de ejecución. Algunos de los motores que utiliza son: HEAP, MyISAM, CSV, Falcon, Cluster, y su propio motor. **Arquitectura interna:** Sistema de asignación de memoria basado en hilos. Tablas temporales o tablas virtuales, formadas por consultas SQL son implementadas en tablas hash en memoria.

1.8.4. Justificación del Gestor de Base de Datos a utilizar en el Sistema.

A partir de los elementos expuestos anteriormente se propone usar para la gestión con la Base de Datos el gestor MySQL, principalmente por su simpleza, rapidez y robustez. Soporta una gran cantidad de datos, desde código abierto y facilita la exportación e importación de datos. Posee gestión de usuarios y contraseñas, manteniendo un alto nivel de seguridad en los datos y es capaz de soportar gran cantidad de tipos de datos para las columnas .Debido a que el CMS Joomla, anteriormente seleccionado, está programado en SQL por lo que utiliza bases de datos relacionales, específicamente MySQL. Al mismo tiempo MySQL 5.1 Generalmente Disponible está disponible en tres modelos para atender a las necesidades exclusivas de los distintos usuarios. Seleccionando MySQL Community Server; la versión de código abierto

disponible gratuitamente. Por estar licenciado bajo el GPL que no requiere soporte comercial o servicios adicionales el ahorro en costes sería notable.

1.9 Servidor Web

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los clientes como por ejemplo los navegadores web, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle: (Alfonso Navarro Albey, 2010)

Espera peticiones en el puerto TCP indicado, luego recibe una petición posteriormente busca el recurso, lo envía utilizando la misma conexión por la que recibió petición y finalmente vuelve a recibir la petición.

Como ejemplos de servidores web pueden encontrarse:

Internet Information Server (IIS) , Apache Web Server, AOLServer, Hawkeye , JavaServer , Xitami, entre otros. (Alfonso Navarro Albey, 2010)

1.9.1 Servidor Web APACHE.

Dentro de los ejemplos mencionados anteriormente se tomará el Apache Web Server para el empleo y desarrollo del sistema pues es el servidor web por excelencia para aplicaciones desarrolladas en PHP, la facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, open-source y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar las sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda (Ciberaula, 2009)

Apache presenta ventajas tales como: (Alfonso Navarro Albey, 2010)Actualmente existen muchos módulos que son adaptables a él, y están ahí para que sean instalados cuando se necesiten, proporciona el código fuente completo y viene con una licencia sin restricciones, es altamente configurable y extensible, trabaja con gran cantidad de lenguajes como PHP, Perl y otros lenguajes script además funciona sobre muchas plataformas como: se ejecuta en Windows, NetWare 5.x y superior, OS/2, y la mayoría de las versiones de Unix, así como varios sistemas operativos. Posee una alta aceptación en la red debido a esto desde 1996 es

el servidor HTTP más usado, llegando a ser el servidor empleado por el 70% de los sitios web en el mundo. Por las ventajas antes expuestas se opta por el mismo. Luego de seguir un estudio del arte se llegó a la conclusión que era conveniente seguir utilizando este tipo de servidor por las facilidades que brinda.

1.10 Conclusiones Parciales

En este capítulo se definieron los principales conceptos y aspectos relacionados con los observatorios. Se analizaron varios tipos de observatorios donde se vieron importantes aspectos a tener en consideración a la hora de elegir cual cumple con las necesidades de esta investigación. Finalmente se propuso la tecnología y las herramientas con las que se desarrollará el portal del observatorio. Después de realizado el estudio del estado del arte y analizar los aspectos importantes a tener en cuenta, cuando se va a desarrollar un observatorio internacional, se concluye que: Sera desarrollado un portal Web para el observatorio internacional de gobierno electrónico que servirá de espacio de colaboración multidisciplinaria a todos los implicados en el desarrollo del gobierno electrónico en el país, contribuyendo a informatizar la gestión del gobierno en el país.

El sistema se realizará con la tecnología siguiente:

1. La metodología mediante la cual se guiará todo el proceso de desarrollo será RUP.
2. Para la representación gráfica de cada uno de los procesos se utilizará UML.
3. La herramienta case para el diagramado que se utilizará será Visual Paradigm.
4. El lenguaje de programación mediante el cual se desarrollará el producto es PHP.
5. Como sistema gestor de contenido se utilizará Joomla.
6. El Entorno de desarrollo Integrado a utilizarse es el Net Beans IDE 6.8.
7. EL framework de programación que se va a utilizar es Symfony
8. EL gestor de base de datos a utilizar en el Sistema es MySQL.
9. El servidor web a utilizar es el APACHE.

Capítulo 2: Modelo, Propuesta Solución

2.1 Introducción

En este capítulo se hace una descripción de la propuesta a desarrollar. Se comienza haciendo un análisis de los requisitos que se necesitan para la construcción de un observatorio de información. Luego se describe el producto que se desea desarrollar, especificando los principales conceptos a tratar y las principales responsabilidades de los módulos por los que está compuesto el producto.

Además se identifican los actores que intervienen en la aplicación y los principales casos de uso. Se hace una descripción de los casos de uso, donde se describe todo el flujo de actividades y las características de cada uno de ellos. Se muestran los principales diagramas que se diseñaron.

2.2 Descripción de Plantillas tecnológicas

La plantilla tecnológica es el documento base del funcionamiento del observatorio. La misma cuenta con una descripción general de una aplicación en específico. La capa más importante de información, son las aplicaciones, que están agrupadas en categorías y subcategorias definidas por el observatorio. La información referente a cada una de las aplicaciones es lo que se almacena en la plantilla tecnológica, la misma se desglosa en un grupo de parámetros que serán el punto de partida para la posterior comparación.

2.3 Técnicas y métodos empleados en la captura de requisitos

Un requerimiento define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Además son condiciones o capacidades que necesita un usuario para resolver un problema o lograr un objetivo. Para ello, durante dicha etapa los analistas se valen de diferentes técnicas de capturas de requisitos que es “la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema”. (Díez, 2001) Algunas de las técnicas utilizadas en este trabajo son:

- **Entrevistas**: Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización. En este caso se han concebido 3 pasos para realizarla: preparación de la entrevista (realizar un escrito de posibles preguntas, establecer momento de la entrevista, entre otros.), realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

- **Tormenta de ideas**: Es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre (S. Raghavan, 2004). Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas donde un integrante del grupo debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.
- **Análisis de soluciones existentes**: Esta técnica consiste en analizar distintos sistemas ya desarrollados que tengan características congruentes con el sistema a ser construido como es el caso del Observatorio de gobierno electrónico de Chile, Perú, entre ostros sistemas. Por un lado, se pueden analizar las interfaces de usuario, observando el tipo de información que se maneja y cómo es manejada. Además de poder analizar las funciones que presenta muchas de las cuales podrían ser muy útiles en sistema a desarrollar, arquitectura, entre otros. Esto puede ser útil para descubrir información importante a tener en cuenta o para comprender la información obtenida en otros métodos de captura.
- **Comparación de terminología y glosario de términos**: Uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello, se ha creado un Glosario de términos identificando el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste). (Calgary, 2001)
- **Estudio de la documentación**: La utilización de esta técnica depende de la información que se tenga almacenada acerca del software que se desee construir. Las entidades guardan información acerca de sus procesos, los modelos o informes necesarios para su desarrollo o para rendir cuenta a los organismos superiores. Con esta técnica los analistas se tratan de determinar posibles requerimientos sobre la base de analizar la documentación utilizada por la empresa. Se debe recolectar cualquier formulario o documento que sea utilizado para registrar o enviar información.

Existen más técnicas para la captura de requisitos, incluso también es común encontrar alternativas que combinen varias de estas técnicas. Sin embargo, las presentadas anteriormente han sido suficientes para la captura de requisitos para el desarrollo del portal para el observatorio internacional de gobierno electrónico en Cuba.

2.4 Modelo Dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Define un modelo de clases común para todos los implicados en el desarrollo, representadas en objetos del dominio, sirve como interlocutor entre clientes y desarrolladores. El propósito fundamental de este modelo es generar una terminología común y sentar las bases del entendimiento del desarrollo y no para definir el sistema completo. Cualquiera sea la solución de casos de uso que se haya elegido, los conceptos e ideas propias del dominio del problema son las mismas; un mismo modelo de dominio contempla cualquiera de las soluciones analizadas. El modelo de dominio es global, es decir se realiza para todos los casos de uso y no para uno en particular.

Debido al bajo nivel de estructuración que presenta el negocio que se está estudiando y que está altamente centrado en tecnologías informáticas, se propone un modelo del dominio debido a que permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Para capturar satisfactoriamente los requisitos y poder construir un sistema correcto se necesita tener un firme conocimiento del funcionamiento del objeto de estudio. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema.

Se realizará la descripción del modelo de dominio mediante un diagrama de clases UML donde se especifican las principales clases conceptuales que puedan intervenir en el sistema, estos representaran los objetos que existen o eventos que suceden en el entorno en el que se trabajará el mismo. Se procederá a explicar los principales conceptos que se identifican en el modelo:

Administrador: Esta clase se refiere a las personas que son administradores de portal. Las principales responsabilidades de estas personas consiste en gestionar Información y asignar rol a los Usuarios .

Información: Esta clase se encuentran las noticias, reportes, encuestas e informaciones que van a ser visualizadas por el portal.

Observatorio: Esta clase están las planillas, una planilla representa un documento donde se almacenan las características y los datos referentes a la aplicación en especifico .En cada planilla aparece una descripción de las secciones de la aplicación.

Secciones: Esta clase se refiere a las secciones que ha de tener la aplicación, en ella se almacenan todas las leyes, normas, estándares y tecnologías que van a ser utilizada en la aplicación.

Subcategoría: Esta clase se refiere a categorías que pueden o no aparecer dentro de las secciones, ejemplo una categoría podría ser legislación, y una subcategoría seria documentos complementarios.

Cliente: Esta clase identifica a todos los que acceden al observatorio, en busca de información referente al gobierno.

Noticia de cambio: Son todas las informaciones de los cambios que ocurran en el observatorio internacional de gobierno electrónico .Cada actualización, eliminación o adición de información constituirá una noticia.

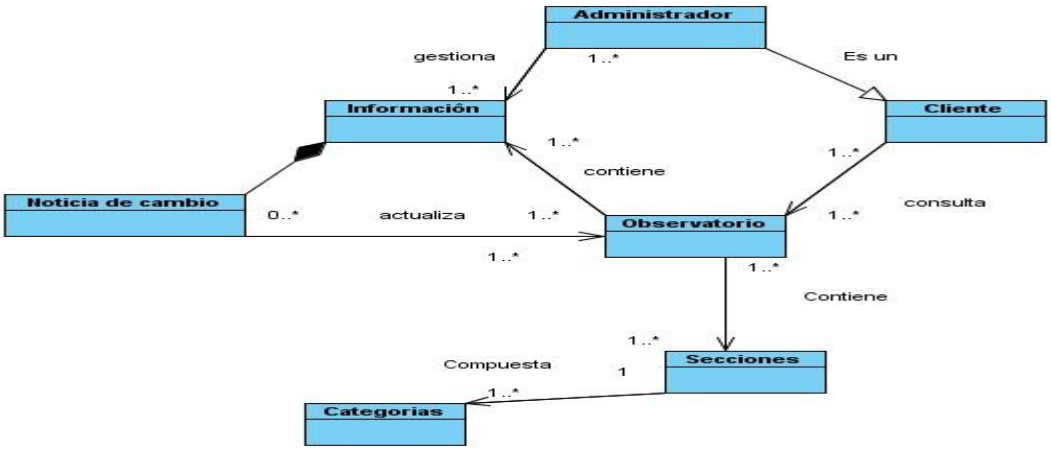


Figura 2.1: Modelo de Dominio

En función de garantizar un mecanismo que permita servir de espacio de colaboración a todos los implicados en el desarrollo del gobierno en el país, se requiere la definición de un observatorio de gobierno electrónico(OGE) que se enfoque en tres aéreas fundamentales: Tecnológica , Socio_Juridica, y Normativa .El análisis y diseño del OGE , será una base sobre la cual un conjunto de usuarios bien definidos , se interrelacionarán, compartirán, publicaran información de forma controlada y sostenida; el mismo estará integrado por un administrador que es el encargado de gestionar las fuentes de información , un grupo de especialistas encargados de la recopilación de datos para estudiar las nuevas tendencias relacionadas con el tema de gobierno electrónico en el mundo , mediante encuestas , los mismo son los encargados de generar los reportes ,y los usuarios los cuales interactuaran con el observatorio en busca de información.

2.5 Especificación de los Requisitos de Software

En este flujo se establece lo que tiene que hacer exactamente el sistema que se va a construir. En esta línea los requisitos son el contrato que se debe cumplir, de manera tal que los usuarios finales comprendan y acepten los requisitos que se especifiquen. Se dividen en dos grupos: los requisitos funcionales y los requisitos no funcionales.

2.5.1 Requerimientos funcionales

Después de analizar un grupo de ejemplos, expuestos en el capítulo 1, los cuales nos introdujeron en el mundo de los observatorios de gobierno electrónicos, se hace evidente que la mejor opción de acuerdo a las necesidades expuestas, es desarrollar un observatorio internacional que permita lo siguiente:

RF.1 Autenticar usuario: El sistema debe dar la opción al usuario de autenticarse, mediante la validación de los datos, previamente gestionados.

RF.2 Crear reportes: El sistema debe permitir generar reportes de las estadísticas relacionadas con el monitoreo del flujo de visitas en cada fuente de información utilizada.

RF.3 Modificar Reportes: El sistema debe dar la opción de modificar reportes.

RF.4 Eliminar Reportes: El sistema debe dar la opción de eliminar reportes.

RF.5 Consultar Reportes: El sistema debe dar la opción de consultar reportes.

RF.6 Crear Encuesta: El sistema debe tener la capacidad de crear encuestas.

RF.7 Modificar Encuestas: El sistema debe tener la capacidad de modificar una encuesta.

RF.8 Eliminar Encuestas: El sistema debe tener la capacidad de eliminar una encuesta.

RF.9 Consultar Encuestas: El sistema debe tener la capacidad de consultar una encuesta.

RF.10 Crear usuario: El sistema debe dar la opción de adicionar un usuario según el rol que los mismos posean.

RF.11 Modificar Usuario: El sistema debe dar la opción de modificar sus datos (nombre, apellidos, usuario, contraseña).

RF.12 Eliminar Usuario: El sistema debe dar la opción de adicionar un usuario según el rol que los mismos posean.

RF.13 Consultar Usuario: El sistema debe dar la opción de consultar un usuario según el rol que los mismos posean.

RF.14. **Consultar fuentes de información**: El sistema a desarrollar debe brindar la posibilidad a los usuarios que visiten el portal de realizar consultas activamente a las fuentes de información.

RF.15. **Insertar noticias:** El sistema debe dar la opción al usuario de insertar noticias referentes al tema de gobierno electrónico a nivel mundial.

RF.16.**Eliminar Noticias:** El sistema debe dar la opción al usuario de eliminar noticias.

RF.17 Consultar Noticias: El sistema debe dar la opción al usuario de consultar noticias.

RF.18 Realizar encuesta: EL sistema debe dar la opción al usuario de realizar una encuesta, mediante la validación de los datos de las encuestas previamente gestionados**.**

RF.19 Insertar Rol: El sistema debe dar la opción al administrador de insertar u rol.

RF.20 Modificar Rol: El sistema debe dar la opción al administrador de modificar rol.

RF.21 Eliminar Rol: El sistema debe dar la opción al administrador de modificar rol.

RF.22 Consultar Rol: El sistema debe dar la opción al administrador de modificar ro.l

RF.23 Insertar fuentes de información: El sistema debe ser capaz de permitir al administrador insertar fuentes de información.

RF 24.Modificar Fuentes de información: El sistema debe dar la opción al administrador de modificar datos de la Fuente.(autor, ano publicación , editorial).

RF.25. Eliminar Fuente de información: El sistema debe dar la opción al administrador de eliminar una Fuente de información.

2.5.2 Requerimientos No funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se muestran los requerimientos no funcionales:

Software:

RNF1: Navegador Internet Explorer v6.0 o superior, Mozilla Firefox v2.x.

RNF2: Servidor Web Apache v2.x o superior con módulo PHP 5.2.3 disponible, este deber estar configurado con las extensiones, “soap”, “ldap”.

RNF3: Como sistema gestor de bases datos: MySQL 5.1.

Hardware:

RNF4: Para los servidores tanto Web como SGBD: Pentium IV o superior con 256MB de RAM o más.

RNF5: Capacidad de Disco Duro mayor de 10 GB.

***Restricciones para el diseño e implementación*:**

RNF6: Utilizar como lenguaje al lado del servidor PHP v5.2.3 o superior y del lado del cliente HTML y Java Script.

RNF7: Para la programación PHP se recomienda el IDE: NetBeans.

RNF8: El código deberá ser reutilizable.

***Apariencia o Interfaz externa*:**

RNF9: Diseñado para la resolución de 1024x768, aunque deberá verse en 800x600 o cualquier resolución superior a esta.

RNF10: La interfaz deber ser agradable para el usuario, que combine bien los colores, tipo de letra y tamaño.

Seguridad:

RNF11: Contar con un sistema de permisos para el acceso a la información.

RNF12: El sistema debe estar disponible las 24 horas del día (Disponibilidad).

RNF13: Se realizarán salvas mensuales del sistema sobre cintas magnéticas

Usabilidad:

RNF14: El sistema debe permitir acceso a los usuarios.

Rendimiento:

RNF15: El sistema deberá ser capaz de gestionar toda la información y dar respuestas a las solicitudes lo más rápido posible.

RNF16 Compatibilidad con diferentes navegadores.

2.6 Modelado del Sistema

El modelo del sistema está basado en las funcionalidades que este debe cumplir y sirve como acuerdo entre clientes y desarrolladores. Este artefacto contiene actores, casos de uso y sus relaciones y sirve como entrada fundamental para el análisis, diseño y pruebas.

2.6.1 Actores del Sistema

Cada trabajador del negocio es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Actores del Sistema

Actor	**Descripción**
Usuario	Es una generalización del actor del sistema representa a todos las personas, que pueden acceder al sistema libremente sin necesidad de registrarse pudiendo visualizar toda la información existente en el sitio.
Publicador	Es una especificación del actor general del sistema “usuario” por tanto además de tener los permisos que tiene dicho actor es el encargado de mostrar el contenido de las fuente de información. También tiene como objetivo insertar o eliminar noticias en el sitio.
Administrador	Es una especificación del actor general del sistema “publicador” por tanto además de tener los permisos que tiene dicho actor es el encargado de gestionar las

	fuentes de información así como gestionar un usuario.
Grupo de Especialista	Es una especificación del actor general del sistema "publicador" por tanto además de tener los permisos que tiene dicho actor es el encargado de generar los reportes, gestionar las encuestas.

Tabla 1.1: Actores del Sistema.

2.6.2 Descripción de los Casos de Uso del Sistema

Un caso de uso del sistema (CUS) es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso (CU) proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. En otras palabras, un CU es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los CU son entidades esenciales a la hora de modelar el diagrama de casos de uso de un sistema. Estos diagramas representan gráficamente a los procesos y su interacción con los actores, especificando la comunicación y el comportamiento del sistema mediante la interacción con los usuarios y/u otros sistemas. Se utilizan para ilustrar los requerimientos del sistema, pues representa las funcionalidades que ofrece el mismo en lo que se refiere a su interacción externa. Los elementos que pueden aparecer en un diagrama de casos de uso son: actores, casos de uso y relaciones entre casos de uso.

A continuación se brinda una breve descripción de un caso de uso crítico del sistema: "Gestionar Usuario de Información". También se expone el diagrama de casos de uso del sistema para ofrecer una visión general de las principales funcionalidades que serán implementadas en el mismo, Para conocer las descripciones de los demás casos de uso consultar anexo 1.

Tabla 2.2: Descripción del CU "Gestionar Usuario**"**

Caso de Uso:	Gestionar Usuario.
Actores:	Administrador.
Resumen:	El caso consiste en que el administrador podrá crear, modificar, eliminar y consultar usuarios en la aplicación.
Precondiciones:	El Administrador debe estar autentificado en la aplicación.

Referencias	**RF 4, RF 1**
Prioridad	Alta
Nivel	Administrador
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el Administrador desea Crear, Modificar, Eliminar o Consultar un usuario.	2. La aplicación muestra la interfaz **"Gestionar Usuario"** con los siguientes datos: • **Crear Usuario.** • **Modificar Usuario.** • **Eliminar Usuario.** • **Consultar Usuario.**
3. Si el Administrador selecciona la opción Crear Usuario, ir a la sección **"Crear Usuario"**. 4. Si el Administrador selecciona la opción Modificar Usuario, ir a la sección **"Modificar Usuario"**. 5. Si el Administrador selecciona la opción Consultar Usuario, ir a la sección **"Consultar Usuario"**. 6. Si el Administrador selecciona la opción Eliminar Usuario, ir a la sección **"Eliminar Usuario"**.	
Sección "Crear Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Crear Usuario"** con los siguientes datos: • **Usuario.** • **Contraseña.**

	• **Rol.** • **Nombre.** • **Apellidos.** Y los botones **"Crear"** y **"Cancelar"**.
2. El Administrador Inserta los datos solicitado por la interfaz. 3. El Administrador da clic en el botón **"Crear"**. 4. Si el Administrador desea cancelar la acción, ir a la sección **"Cancelar".**	5. La aplicación valida que el usuario no existe ya en la base de datos, y muestra un mensaje **"Se ha creado un nuevo usuario"**. 6. En caso que muestre el siguiente mensaje **"El usuario ya existe en la base de datos"** ir a la sección **"Usuario Existente".** 7. En caso que los datos estén incompletos la aplicación muestra un mensaje **"Existen campos incompletos o vacíos. Por favor, complete los datos."**, ir a la sección **"Datos Incompletos"**.
8. El Administrador da clic en **"Aceptar"**. Termina el caso de uso.	
Prototipo de Interfaz	
Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y los botones **"Buscar"** y **"Cancelar"**.
2. El Administrador introduce el dato solicitado por la interfaz, y da clic en el botón **"Buscar"**.	3. La aplicación muestra la interfaz **"Modificar Usuario"** con los siguientes datos: • **Usuario.**

	• **Nombre(s).** • **Apellidos.** • **Rol.** • **Contraseña.** Y los botones **"Modificar"** y **"Cancelar"**. 4. En caso que no se inserte algún dato solicitado por la aplicación se muestra el mensaje **"Campos incompletos. No se introdujo ningún dato."**, ir a la sección **"Datos Incompletos"**.
5. El Administrador da clic en el botón **"Modificar"**. 6. Si el Administrador desea cancelar la acción, ir a la sección **"Cancelar".**	7. Si la acción fue correcta la aplicación muestra un mensaje **"El usuario ha sido modificado correctamente."**. 8. En caso que no se inserte algún dato solicitado por la aplicación se muestra el mensaje **"No se introdujo ningún dato."**, ir a la sección **"Datos Incompletos"**.
9. El Administrador da clic en el botón **"Aceptar"**. Termina el caso de uso.	
Prototipo de Interfaz	
Sección "Consultar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y los botones **"Buscar"** y **"Cancelar"**.
2. El Administrador Inserta el dato solicitado por la interfaz, y da clic en el botón	3. La aplicación muestra la interfaz **"Consultar Usuario"** con los siguientes

"Buscar".	datos: • **Usuario.** • **Nombre(s).** • **Apellidos.** • **Rol.** Y el Botón **"Aceptar"**. 4. En caso que no se inserte algún dato solicitado por la aplicación se muestra el mensaje **"No se introdujo ningún dato."**, ir a la sección **"Datos Incompletos"**.
5. El Administrador da clic en el botón **"Aceptar".** Termina el caso de uso.	
Prototipo de Interfaz	
Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y los botones **"Buscar"** y **"Cancelar"**.
2. El Administrador Inserta el dato solicitado por la interfaz, y da clic en el botón **"Buscar".**	3. La aplicación muestra la interfaz **"Eliminar Usuario"** con los siguientes datos: • **Usuario.** Y el Botón **"Eliminar"**. 4. En caso que no se inserte algún dato solicitado por la aplicación se muestra el mensaje **"No se introdujo ningún**

	dato.”, ir a la sección **“Datos Incompletos”**.
5. El Administrador selecciona el usuario que desea eliminar, y da clic en el botón **“Eliminar”.**	6. La aplicación muestra un mensaje **“Desea eliminar el usuario seleccionado.”**, y los botones **“Aceptar”** y **“Cancelar”**.
7. El Administrador da clic en el botón **“Aceptar”.** 8. Si el Administrador selecciona cancelar la acción, ir a la sección cancelar.	9. La aplicación muestra un mensaje **“El usuario a sido eliminado correctamente.”**
10. El Administrador da clic en el botón **“Aceptar”.** Termina el caso de uso.	
Prototipo de Interfaz	
Flujo Alternos	
Sección “Datos Incompletos”	
Acción del Actor	**Respuesta del Sistema**
1. El usuario oprime el botón **“Aceptar”**.	
Prototipo de Interfaz	
Sección “Cancelar”	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador de clic en el botón **“Cancelar”**.	
Prototipo de Interfaz	
Sección “Usuario Existente”	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador de clic en el botón	2. Ir a la Respuesta del Sistema # 4.

"Aceptar".	
Prototipo de Interfaz	
Pos condiciones	El usuario debe quedar autentificado.

2.6.3. Diagrama de Caso de Uso del Sistema

Los **diagramas de casos de uso** documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar

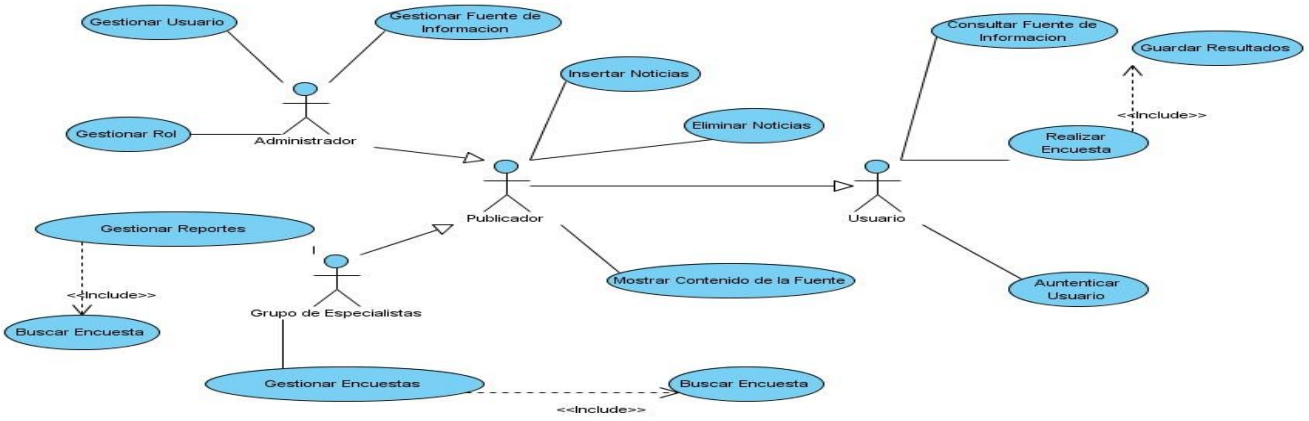


Figura 2.2: Diagrama de casos de uso del sistema

2.7 Modelo de Análisis

"Durante el análisis, se analizaron los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura". (Jacobson y otros, 2000)

Aunque en el modelo del análisis hay un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema, pues el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución. El modelo del análisis es un artefacto que usualmente se genera para entender

claramente los requisitos y realizar mejor el diseño. Es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño y puede contener: las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

2.7.1 Diagramas de Clases del Análisis

En el diagrama de clases de análisis se representa los conceptos fundamentales que comprende el dominio del problema. Estos diagramas muestran que clases participan en las realizaciones de los distintos casos de uso, representan las definiciones y relaciones entre ellas. Las clases del análisis se clasifican en Interfaz, de Control o Entidad.

Clase Interfaz: Modela la interacción entre el sistema y sus actores.

Clase Control: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

Clase Entidad: Modelan información que posee larga vida y que es a menudo persistente.

La siguiente imagen muestra el diagrama de clases del análisis correspondiente al caso de uso Gestionar Usuario, a través de la misma se representa los actores y su interacción con el sistema Para consultar los diagramas de clases del análisis de los demás casos de uso consultar Anexo 2 “Modelo del Análisis”.

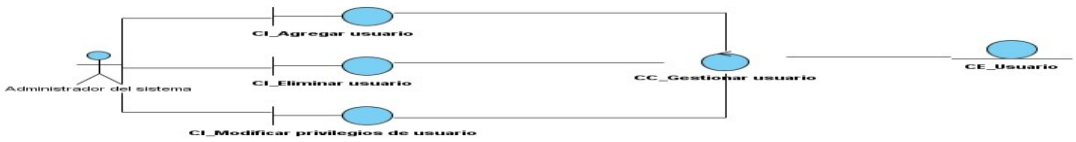


Figura 2.3: Diagrama de Clases del Análisis” Gestionar Usuario

2.7.2 Patrones del análisis

Los patrones son soluciones simples compuestos por una pareja problema/solución, fundamentadas en la experiencia para problemas específicos y comunes, se ha demostrado que funcionan, pueden emplearse en diferentes contextos. Algunos de los utilizados son:

Concordancia: toma una subsecuencia de acciones que estén en diferentes partes del flujo de casos de uso (CU) y la define por separado.

Inclusión Concreta: es de estructura. Consiste en dos CU y una relación de inclusión entre el CU base y el CU incluido. Este último puede ser instanciado por sí solo. El CU base puede ser concreto o abstracto. Se utiliza cuando un flujo de datos puede ser incluido en el flujo de otro CU y también puede ejecutarse por sí solo. . Ejemplo del mismo en el análisis de la solución es el que se evidencia entre los casos de Gestionar Reportes y Buscar Reportes, el cual sin importar la funcionalidad a realizar por el caso base, el CU incluido se ejecuta por sí solo.

CRUD Completo: es un caso de uso llamado Información de CRUD o Administrar Información, que modela las diferentes operaciones que pueden realizarse en un pedazo de información de un cierto tipo, como crear, leer, actualizar, y eliminar. Debe usarse cuando todos los flujos contribuyen al mismo valor del negocio y estos son cortos y simples. Ejemplo el gestionar Usuario.

CRUD Parcial: patrón alternativo que modela uno de las alternativas del CU como un CU separado. Es preferible cuando uno de las alternativas del CU es más significativa, grande, o mucho más compleja que las otras alternativas. Ejemplo en la aplicación mediante el CU gestionar fuente de información donde, el administrador es el encargado de crear, modificar y eliminar dicha fuente, pero todos los usuarios pueden consultarlo.

Múltiples actores: captura la concordancia entre actores manteniendo roles separados.

Roles comunes: puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del CU, solo exista una entidad externa interactuando con cada una de las instancias del CU , ejemplo es entre los actores administrador y el grupo de especialistas .

Estos patrones brindan la posibilidad de realizar un mejor y más entendible modelado del sistema. Esto es de gran importancia, pues este modelo es una entrada fundamental para realizar el diseño del sistema.

2.8 Modelo de Diseño

"El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el Modelo de Diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación". (Jacobson y otros, 2000)

El modelo de diseño está muy próximo al de implementación, obviamente para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. En el diseño se modela el sistema y se obtiene su

forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada fundamental en el diseño es el resultado del análisis, que proporciona una comprensión detallada de los requisitos. Además, impone una estructura del sistema la cual hay que conservar para proporcionarle forma al mismo. Este modelo puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.

2.8.1 Diagramas de Clases del Diseño

Los diagramas de clases del diseño son los diagramas principales en el flujo de trabajo análisis y diseño para obtener un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones. En la figura siguiente se expone el diagrama de clases del diseño para un caso de uso: "Gestionar Usuario". En el diagrama se representa las clases que intervienen dentro del caso de uso así como las relaciones entre estas. Para conocer los diagramas de clases del diseño correspondientes al resto los casos de uso del sistema consultar anexo 3 "Modelo de Diseño".

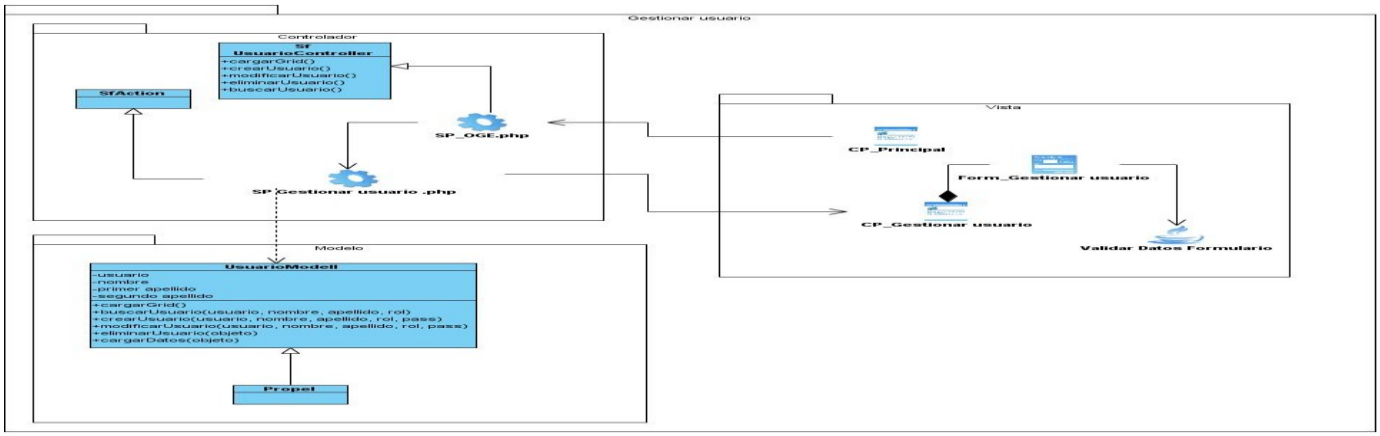


Figura 2.4: Diagrama de Clases del Diseño CU Gestionar usuario.

2.8.2 Diagramas de Interacción del Diseño

Los diagramas de interacción del diseño muestran las relaciones entre actores y las clases del diseño que participan en un caso de uso, además de los mensajes que se pasan entre ellos para lograr un objetivo. Estos

diagramas, al igual que los de interacción del análisis, se expresan de dos formas: diagramas de secuencia y diagramas de colaboración. A continuación se ofrecen la imagen del diagrama de secuencia del diseño para el caso de uso “Gestionar Usuario”. Para chequear los diagramas de secuencia del diseño correspondientes al resto de los casos de uso consultar el anexo 3 “Modelo de Diseño”.

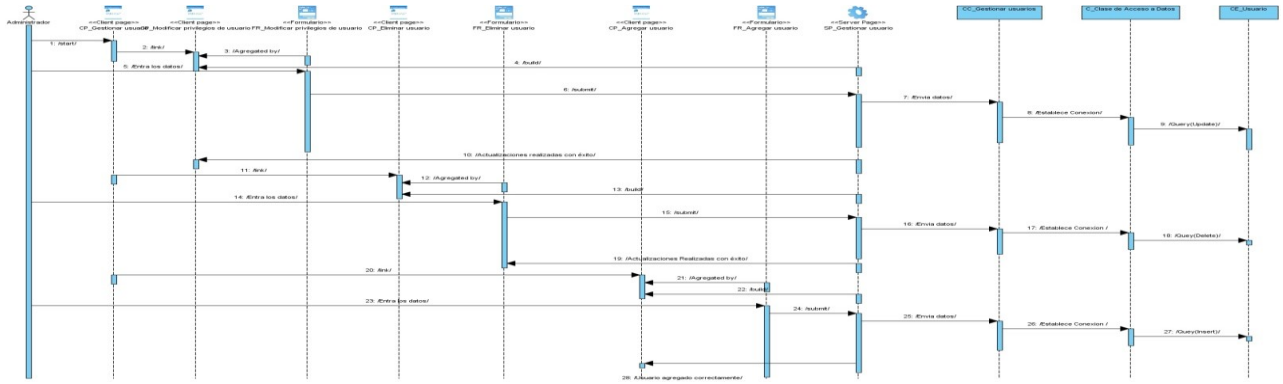


Figura 2.4: Diagrama de Secuencia del Diseño CU Gestionar usuario

2.9 Arquitectura definida para el Sistema

“La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.” (Kruchten, 1995)

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema informático. Lo usual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Precisamente, las arquitecturas más universales son:

Monolítica: Donde el software se estructura en grupos funcionales muy acoplados.

Cliente-servidor: Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

Arquitectura de tres niveles: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia).

Los sitios web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas que permitan dar con la arquitectura óptima que facilite la construcción de los mismos. El usuario interacciona con las aplicaciones web a través del navegador. Como consecuencia de la actividad del usuario, se envían peticiones al servidor, donde se aloja la aplicación y que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta al navegador que la presenta al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador, que presenta la interfaz al usuario; la aplicación, que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste y la base de datos, donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas.

Capa de Presentación*:* Esta capa es la que ve el usuario, presenta el sistema al usuario. Le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz de gráfica y debe tener la característica de ser amigable para el usuario por lo que generalmente se presenta como formularios.
Capa de Negocio: Es en esta capa donde, se describen las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso la lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
Capa de Datos: Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

2.8.3 Patrones de Diseño Utilizados

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles". (Booch, Grady)

Los patrones de diseño proponen soluciones simples y exitosas a problemas habituales que se pueden presentar durante el diseño, basadas en la experiencia. Un patrón de diseño soluciona problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

Symfony previamente seleccionado como framework de desarrollo está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles:

El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. La Vista transforma el modelo en una página web que permite al usuario interactuar con ella. El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo gestor de bases de datos utilizado por la aplicación. (Potencier, 2009)

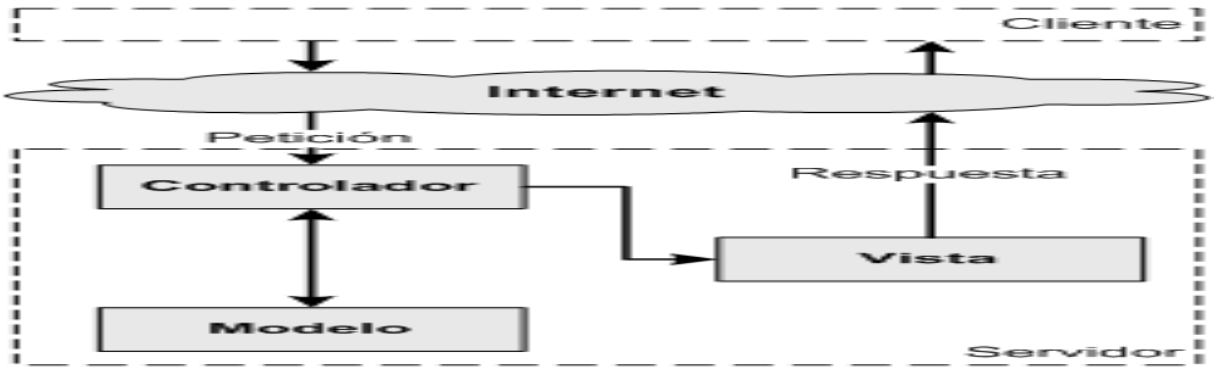


Figura 2.5: Arquitectura MVC

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las

aplicaciones. Los diseñadores web normalmente trabajan con las plantillas (que son la presentación de los datos de la acción que se está ejecutando) y con el layout (que contiene el código HTML común a todas las páginas). Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP. El contenido de las plantillas se integra en el layout, o si se mira desde otro punto de vista, el layout decora las plantillas. Este comportamiento es una implementación del patrón de diseño llamado "decorador".

Ilustración 1.6: Patrón Decorador

Los patrones de diseño tienen como características: (Larman, 2004)

- *Son soluciones concretas:* proponen soluciones a problemas concretos, no son teorías genéricas.
- *Son soluciones técnicas:* indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- *Se utilizan en situaciones frecuentes:* ya que se basan en la experiencia acumulada la resolver problemas reiterativos.
- *Favorecen la reutilización de código:* ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

  A continuación se muestran los patrones de diseño utilizados para el desarrollo de dicho portal.

Patrones GRASP: Ios patrones de asignación de responsabilidades (General Responsibility Assignment Software Patterns, GRAPS) describen los principios fundamentales de la asignación de responsabilidades a objetos. Constituyen el fundamento de cómo se va a diseñar el sistema finalmente. Es importante que el diseñador de software domine y aplique estos conocimientos durante la realización de un diagrama de interacción. (Larman, 2004)

Experto:

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Este patrón se usa más que cualquier otro al asignar responsabilidades. El patrón Experto ofrece una analogía con el mundo real.

Creador:

Solución: Asignarle a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:

- B agrega los objetos de A.
- B contiene los objetos de A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así B es un Experto respecto a la creación de A).

Problema: ¿Quién debería ser el responsable de crear una nueva instancia de alguna clase?

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento.

Bajo acoplamiento:

Solución: Asignar una responsabilidad para mantener bajo acoplamiento.

Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

El Bajo acoplamiento es un principio que debemos recordar durante las decisiones del diseño: es la meta principal que es preciso tener siempre presente.

Alta cohesión:

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Problema: ¿Cómo mantener la complejidad dentro de límites manejables?

Alta cohesión es un principio de debemos tener presente en todas las decisiones de diseño: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.

Controlador:

Solución: Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El sistema global.
- La empresa u organización global.
- Algo activo en el mundo real que pueda participar en la tarea.
- Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Patrones GOF: estos patrones surgen a raíz del trabajo de un grupo de autores conocido como el Gang of Four (GoF). Ellos recopilaron y documentaron un grupo de patrones de diseño aplicados usualmente por diseñadores de software orientado a objetos.

El grupo GoF agrupó los patrones en tres grandes categorías de acuerdo a su propósito. Este criterio describe la función que el patrón cumple. Los patrones de diseño pueden tener propósito creacional, estructural o de comportamiento: (Larman, 2004)

Patrones Creacionales: Se encargan de las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Fábrica abstracta *(Abstract Factory):* permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. **Instancia única** (Singleton): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones Estructurales: Describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

Fachada (Facade): El patrón fachada trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace de pantalla o fachada.

Decorator: añade funcionalidad a una clase dinámicamente.

Patrones de Comportamiento: Nos ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Iterador *(Iterator*): permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

Mediador *(Mediator):* define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.

2.8.4 Modelo de Datos

Un Modelo de datos es un conjunto de conceptos, reglas y convenciones para describir distintos aspectos del negocio del sistema. Requiere conocimiento, normas y estándares que aseguren la correcta descripción e interpretación de dichas técnicas. Su objetivo es producir cierta descripción estructurada de la organización y el negocio del cliente para permitir construir un sistema. Existen varias clasificaciones de los modelos de datos, dentro de los cuales podemos mencionar los modelos de datos conceptuales, los de datos lógicos y los de datos físicos.

Diagrama de Clases Persistentes.

Un diagrama de clases muestra un conjunto de clases, interfaces, y colaboraciones y sus relaciones. Gráficamente un diagrama de clase es una colección de vértices y arcos. Un diagrama de clase persistente es justo un tipo de diagrama que muestra un conjunto de objetos deben ser almacenados en algún repositorio como una base de datos relacional.

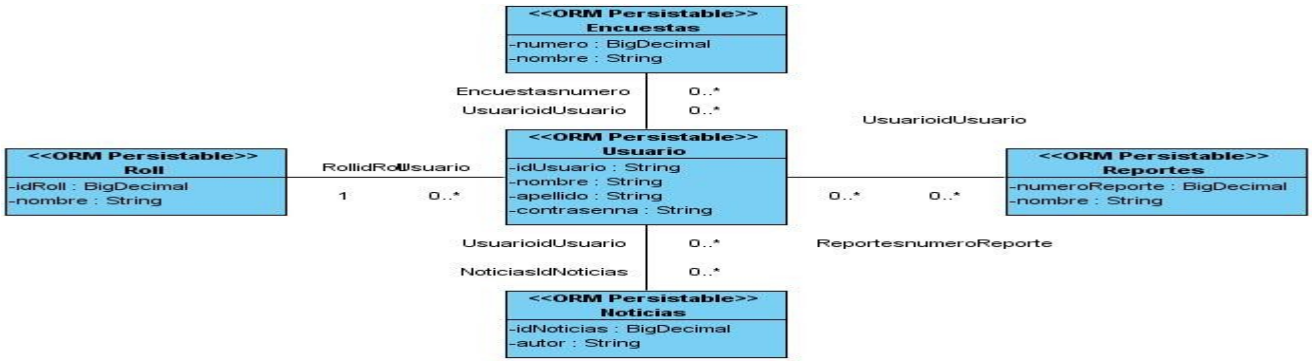
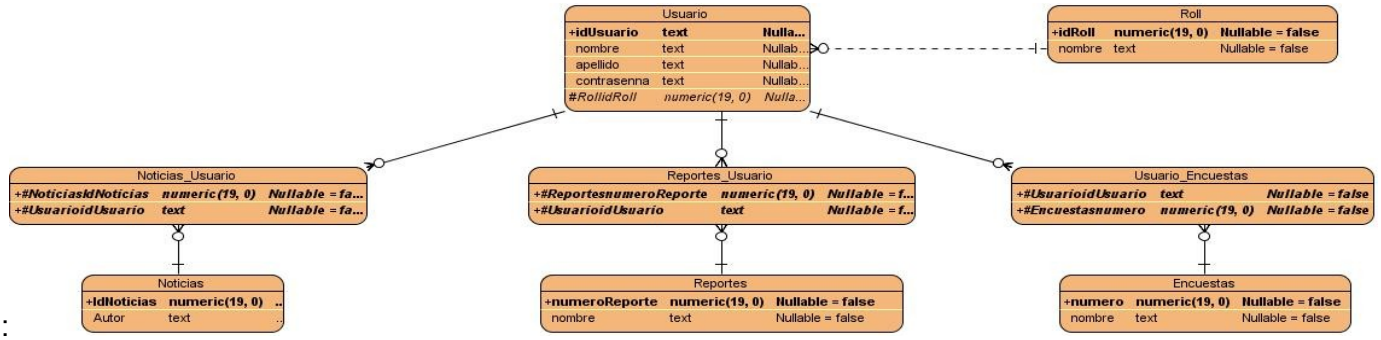


Figura 2. Diagrama Clases Persistentes del observatorio

Diagrama de entidad relación

Al modelar un sistema, la técnica del modelado Entidad - Relación mediante diagramas o modelos Entidad - Relación (denominado por sus siglas, E-R “Entity relationship”, o, "DER" Diagrama de Entidad Relación) brindan comodidades, ya que estos constituyen una representación conceptual de la información, mediante la cual se pretende “visualizar” los objetos que pertenecen a la Base de Datos como entidades (se corresponde al concepto de objeto de la Programación Orientada a Objetos) las cuales tienen atributos y se vinculan mediante relaciones. Además, estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. Seguidamente se muestran los DER pertenecientes a los procesos Realizar Inspección Sanitaria e Interconsulta, así como la descripción de la taba de la base de datos:



:

Figura 2.8 Diagrama Entidad Relación del observatorio.

En el anexo 4 se pueden observar las descripciones de las tablas de las base de datos.

2.9 Conclusiones Parciales

En este capítulo se llegó a las siguientes conclusiones:

- La utilización de técnicas para la identificación de requisitos, permitió obtener resultados satisfactorios en la identificación de las funcionalidades con las que deberá contar el sistema.

- Se logró un refinamiento de los requisitos capturados, se identificaron los actores encargados de interactuar con el sistema, se generó y especificó el diagrama de casos de uso del sistema.

- Los artefactos creados en Visual Paradigm y UML como lenguaje de modelado, propició un mayor entendimiento entre los involucrados.

- La aplicación de patrones, tanto de casos de uso como de diseño, posibilitó obtener artefactos aceptables y evitó que se cometieran errores tradicionales.

- La construcción del modelo del sistema y diseño permitió obtener los elementos que deberán ser implementados.

- En este capítulo se han mostrado las principales funcionalidades que podría tener el sistema a través de una serie de modelos, que garantizan un mayor entendimiento del mismo.

Capítulo 3: Validación de los resultados

3.1 Introducción

Una métrica de calidad es una definición operativa que describe, en términos muy específicos, un atributo del producto o del proyecto, y la manera en que el proceso de control de calidad lo medirá. Una medición es un valor real. La tolerancia define la variación permisible de las métricas. Las métricas de calidad se emplean en los procesos de aseguramiento de la calidad y de control de calidad. (Institute, 2000).

Como todas las métricas de desarrollo de software los objetivos principales de las métricas orientadas a objetos se derivan del software convencional: comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto.

En el presente capítulo se hace un análisis de los resultados obtenidos durante el desarrollo de la solución propuesta. Se aplican métricas para medir la calidad de la funcionalidad del diagrama de casos de uso del sistema y para la validación de los artefactos del modelo del diseño se aplican métricas a nivel del tamaño de clases.

3.2 Validación de los Requisitos

La identificación de los requisitos y la especificación de los de los casos de uso constituyen los principales artefactos obtenidos durante el desarrollo de la ingeniería de requisitos del portal para el Observatorio Internacional de Gobierno Electrónico en Cuba. Con el objetivo de garantizar su factibilidad fueron sometidos a varias iteraciones de revisiones por parte del proyecto calidad de la facultad. Se aplicaron listas de chequeo para los requisitos y para la descripción de los casos de uso. Las no conformidades encontradas durante las iteraciones de la revisión fueron solucionadas por el equipo de análisis. Además se hicieron varias revisiones por parte de los usuarios, emitiendo posteriormente una Carta de Aceptación como constancia de las revisiones.

3.3 Métricas para la Calidad de la Especificación de los Requisitos de Software

Los requisitos, una vez definidos, necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de estos define realmente las funcionalidades que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Con el objetivo de medir la satisfacción del cliente se aplicaron un conjunto de métricas para la calidad de la Especificación de los Requisitos de Software. A continuación se muestra como se realizó este proceso.

Inicialmente fue necesario calcular el número total de requisitos, para así poder aplicar las métricas que hacen uso de este. (Overmyer, 1993)

Rf: Número de requisitos funcionales.

Rnf: Número de requisitos no funcionales.

Rt: Total de requisitos.

Rt = RF +Rnf

Rt = 25 + 19 = 44

- **Especificidad**

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos se empleó la métrica basada en la consistencia de la interpretación de los revisores para cada requisito. Cuanto más cerca de 1 esté el valor de Q1 (grado de especificidad de los requisitos), mayor será la consistencia de la interpretación de los revisores para cada requisito y menor será la ambigüedad de la especificación de los requisitos. (Overmyer, 1993)

$R11$: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

$Q1 = R11 / Rt$

Q1=40/44 =0.91

- **Corrección**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos. Este valor se calcula como se muestra a continuación: (Overmyer, 1993)

Q2: Grado de validación de los requisitos.

Rc: Número de requisitos que se han validado como correctos.

Rnv: Número de requisitos que no se han validado como correctos todavía.

Q2 = Rc / (Rc + Rnv)

Q2 =44/ (44+0) = 1

- **Compleción**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de completitud en la definición de los requisitos. Este valor se calcula como se muestra a continuación (Overmyer, 1993):

Na**:** Número de requisitos completos.

Nb**:** Número de requisitos pobremente especificados**.**

Q3 = Na / (Na + Nb)

Q3 = 44/(44+0) = 1

- **Comprensión**

La comprensión de los requisitos se determinó a partir de la relación que se muestra a continuación. El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1. (Overmyer, 1993)

Rbc**:** Número de requisitos que todos los revisores entienden.

Q4 = Rbc / Rt

Q4 = 44 /44 = 1

- **Consistencia interna**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que no existen subconjuntos de requisitos contradictorios. El valor óptimo de esta métrica es el más cercano a 1. (Overmyer, 1993)

Nu: Número de requisitos especificados.

Nn: Número de requisitos en conflicto con otros requisitos en la especificación.

Q5 = (Nu - Nn) / Nu

Q5 = (44 - 0) / 44 = 1

- **Consistencia externa**

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que ninguno de los requisitos está en contradicción con lo expresado en documentos de nivel superior, o sea los requisitos del software no pueden contradecir los requisitos del sistema. El valor óptimo de esta métrica es el más cercano a 1.

Nec: Número de requisitos que son consistentes con otros documentos.

Q6 = Nec / Rt

Q6 = 44 / 44 = 1

- **Estabilidad**

Para medir la estabilidad de los requisitos de software, en el presente trabajo se aplicó la métrica propia para esto, la cual ofrece valores entre 0 y 1. El mejor valor de **E** es el más cercano a 1. La estabilidad de los requisitos se calcula de la siguiente forma:

E: Valor de la estabilidad de los requisitos. (Overmyer, 1993)

Rt: Total de requisitos.

Rm: Número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados.

E = (Rt - Rm) / Rt

E = (44-4)/ 44 = 0.90

Para las métricas especificidad, correctitud, comprensión, consistencia interna y externa se recomienda que los resultados estén más cercanos a 1 para ser clasificados como óptimos. El umbral para la métrica de compleción se recomienda que sea un peso aproximado a 0.7, aunque si este es 1 la métrica es clasificada como completa. En el caso de la estabilidad se clasifica en alta (0.90<=**E**<=1), en media (0.80<=**E**<0.90) y en baja (0.7<=**E**<0.80). Con los resultados obtenidos se demostró que el proceso de especificación de los requisitos cuenta con una alta calidad , al utilizar la métricas especificidad se obtuvo como resultado que la ambigüedad en los requisitos fue mínima debido a que el nivel de especificidad es alto ,en cuanto a las métricas de corrección, comprensión, consistencia interna y externa se obtuvieron resultados óptimos puesto a que el resultado de la aplicación de dichas métricas, garantizo un alto nivel de corrección ,completitud , comprensión , con el elevado nivel de consistencia se puede garantizar que no existe contradicción entre los requisitos ya sean internos como en el nivel superior, además ello demuestra que se hizo un buen trabajo durante el proceso de captura de requisitos y que hay entendimientos comunes entre clientes y equipo de desarrollo.

3.4 Métricas para la evaluación de la calidad en los diagramas de casos de usos.

Las métricas para la calidad de la funcionalidad del diagrama de casos de uso de sistema definen cuatro atributos (Overmyer, 1993):

- Completitud: permite determinar el grado en que se ha incluido de forma clara y concisa todos los elementos necesarios para la descripción del aspecto.
- Consistencia: permite definir el grado en que los elementos del artefacto representan en forma única y no contradictoria un aspecto del problema.

- Corrección: permite establecer el grado de adecuación del artefacto para satisfacer los requerimientos establecidos.
- Complejidad: permite medir el grado de claridad y rehúso del artefacto.

Estos atributos presentan un significado determinado de acuerdo con el tipo de artefacto y al nivel de abstracción que éste describe. Cada atributo se evalúa en términos de un conjunto de factores, los cuales tendrán asociados una métrica.

A continuación se muestra la tabla de las métricas correspondientes a cada uno de los atributos especificados anteriormente (Overmyer, 1993):

No.	Atributo.	Métricas.
1.	Completitud	Número de casos de uso que no tiene descripción resumida.
2.		Número de casos de uso que tienen requerimientos omitidos.
3.		Número de casos de uso que no poseen una descripción extendida.
4.		Número de casos de uso que tienen acciones del flujo de eventos no redactados en función del responsable.
5.		Número de casos de uso que no describen condiciones de excepción relevantes.
6.		Número de casos de uso que no han sido clasificados.
7.	Consistencia	Número de casos de uso que tienen un nombre incorrecto.
8.		Número de casos de uso que no representan una interacción observable por un actor.

9.		Número de casos de uso que tienen acciones del flujo de eventos asignados a un responsable que no le corresponde.
10.		Número de casos de uso no aceptados.
11.		Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos.
12.		Número de casos de uso que no tienen un usuario responsable.
13.	Corrección	Número de casos de uso en que los requerimientos representados no son comprensibles por el usuario.
14.		Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.
15.		Número de casos de uso que deben ser modificados para mejorar el proceso actual.
16.	Complejidad	Número de elementos del diagrama que requieren reubicación.

Tabla 3.1 Métricas de calidad de diagrama de casos de uso.

Resultados por métricas

Métrica	No de casos de uso analizados	Resultado	Porcentaje de error
1	14	Todos poseen una descripción resumida.	0%
2	14	Ninguno presenta requerimientos omitidos, al menos están relacionados con 1 requerimiento.	0%
3	14	No se detectó ningún caso de uso que no presentara su descripción extendida.	0%
4	14	Todos están redactados en función del responsable que le corresponde a cada uno.	0%
5	14	No se detectó ningún caso de uso que no describiera condiciones de excepción relevantes.	0%
6	14	Todos han sido clasificados según los diferentes tipos de prioridad.	0%

7	14	Ninguno presentó problemas en cuanto al nombre dado, ya que todos los nombres representan una expresión verbal en infinitivo describiendo alguna funcionalidad relevante y significativa.	0%
8	14	Todos representan correctamente la interacción con un actor.	0%
9	14	Ninguno presentó acciones del flujo de eventos que estuvieran asignados a un responsable que no le corresponde.	0%
10	14	Todos fueron aceptados.	0%
11	14	Todos poseen sus descripciones del flujo básico y de flujos alternos separadas.	0%
12	14	Todos poseen un usuario responsable.	0%
13	14	Todos los casos de uso en que los requerimientos fueron representados son comprensibles por el usuario.	0%
14	14	Se detectó un caso de uso que debe ser modificado para adecuarlo a la funcionalidad del sistema.	7.14%
15	14	Ninguno tuvo que ser modificados para mejorar el proceso actual.	0%
16	14	Ninguno de los elementos que conforman el diagrama de casos de uso requiere reubicación.	0%

Luego de haber aplicado las métricas al diagrama de casos de uso, se han graficado los resultados obtenidos, los cuales se representan a continuación.

Como resultado de aplicar esta métrica se pudo observar la calidad y funcionalidad del diagrama de casos de uso, obteniéndose un 100% tanto en completitud, consistencia, complejidad y correctitud, estos resultados se obtuvieron a través de 3 interacciones donde se fueron refinando dichos casos de uso.

3.5 Métricas orientadas a clases para evaluar el diseño

Las métricas dentro del desarrollo de sistemas implican cualquier medida o conjunto de medidas destinadas a conocer o estimar estadísticas de un software o un sistema de información, generalmente para realizar comparativas o para la planificación de proyectos de desarrollo.

Para la evaluación del diseño, se genera un documento donde se definen y aplican las métricas de Tamaño operacional de la clase y de Relación entre clases las cuales serán explicadas a continuación:

3.5.1 Tamaño operacional de la clase

El tamaño operacional de las clases está dado por el número de métodos asignados una clase. Mediante el cual se calcula el nivel de responsabilidad de los métodos, la complejidad de implementación de los mismos y su reutilización a fin de inspeccionar la efectividad del diseño, utilizando estos criterios como un conjunto de atributos que se ven afectados para lograr de forma estadística el tamaño de las operaciones realizables:

Tabla 3.2 Resultados del instrumento de evaluación de la métrica Tamaño Operacional de clase (TOC)

Tamaño operacional de clase (TOC)	
Atributos que afecta:	**Modo en que lo afecta:**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

Primeramente se listan los métodos de las clases que intervienen en el proceso.

Subsistema	Clase	Cantidad de Procedimientos
OGE	usuarioController	5
OGE	usuarioModel	6
OGE	rollController	5
OGE	rollModel	6
OGE	encuestaController	5
OGE	encuestaModell	6
OGE	noticiaController	4
OGE	noticiaModel	4
OGE	fuenteController	4
OGE	fuenteModel	5
OGE	consultarFuenteController	1
OGE	consultarFuenteModel	2
OGE	autenticarController	1
OGE	autenticarModel	1
OGE	realizarEncuestaController	3
OGE	realizarEncuestaModel	3
OGE	mostrarContenidoControler	2
OGE	mostrarContenidoModel	2

Tabla 3.3 Datos.

Se le halla el promedio de sus procedimientos con respecto a la cantidad de clases.

Total de clases: 18

Promedio de procedimientos: 3.61

Teniendo los valores anteriores se le aplican los siguientes criterios para obtener en un rango (alta, media o baja) de las estadísticas deseadas:

Tabla 3.4. Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

	Categoría	Criterios
Responsabilidad	Alta	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Baja	> 2* Prom.
Complejidad de implementación	Alta	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Baja	> 2* Prom.
Reutilización	Alta	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Baja	<= Prom.

Obteniendo el rango de responsabilidad, complejidad y reutilización de cada clase:

Tabla 3.5. Resultados

Clase	Responsabilidad	Complejidad	Reutilización
usuarioController	Media	Media	Media
usuarioModel	Media	Media	Media
rollController	Media	Media	Media
rollModel	Media	Media	Media
encuestaController	Media	Media	Media
encuestaModell	Media	Media	Media
noticiaController	Media	Media	Media
noticiaModel	Media	Media	Media
fuenteController	Media	Media	Media
fuenteModel	Media	Media	Media
consultarFuenteController	Baja	Baja	Alta
consultarFuenteModel	Baja	Baja	Alta

autenticarController	Baja	Baja	Alta
autenticarModel	Baja	Baja	Alta
realizarEncuestaController	Baja	Alta	Alta
realizarEncuestaModel	Baja	Alta	Alta
mostrarContenidoControler	Baja	Alta	Alta
mostrarContenidoModel	Baja	Alta	Alta

Estadísticamente se pueden analizar los promedios de los procedimientos por criterio de cantidad de procedimientos de la siguiente manera:

Tabla 3.6. Resultados por criterios de cantidad de procedimientos

Criterio	Cantidad de clases	Promedio
Entre 1 y 5 procedimientos	15	83,33333333
Entre 6 y 10 procedimientos	3	16,66666667
Entre 11 y 15 procedimientos	0	0
Entre 16 y 20 procedimientos	0	0
Entre 21 y 25 procedimientos	0	0
Más de 26 procedimientos	0	0
Total	18	100

Otra de las vías de análisis es de manera gráfica, donde es más fácil apreciar los porcientos que representa cada promedio de procedimiento

Analizando los resultados de manera independiente se pueden realizar las siguientes apreciaciones:

- Responsabilidad

Responsabilidad	Cantidad de clases	Promedio
Baja	8	44,44444444
Media	10	55,55555556
Alta	0	0

- Complejidad

Complejidad	Cantidad de clases	Promedio
Baja	8	44,44444444
Media	10	55,55555556
Alta	0	0

- Reutilización

Reutilización	Cantidad de clases	Promedio
Alta	8	44,44444444
Media	10	55,55555556
Baja	0	0

Según los valores obtenido por la métrica utilizada TOC La mayoría de las clases que conforman el sistema están dentro de las categorías de baja y media, para un 94 % del total, lo que demuestra la elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto por lo que la implementación se hace menos compleja, las pruebas son más fáciles de realizar y aumenta la responsabilidad de las clases. Por tanto se concluye que los resultados obtenidos según esta métrica son positivos.

3.5.2 Relaciones entre clases

Las relaciones entre las clases están dadas por el número de relaciones de uso que tenga una clase con otras. Mediante la cual se calcula el Acoplamiento, la Complejidad, la Reutilización y la Cantidad de pruebas a fin de inspeccionar la efectividad del diseño, utilizando estos criterios como un conjunto de atributos que se ven afectados para lograr de forma estadística el tamaño de las operaciones realizables:

Atributos que afecta:	**Modo en que lo afecta:**
Acoplamiento	El aumento del RC provoca un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	El aumento del RC provoca un aumento de la

	complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC provoca una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	El aumento del RC provoca un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 3.7. Métrica Relaciones entre Clases (RC)

Resultados del instrumento de evaluación de la métrica Relaciones entre clases (RC)

Primeramente se listan los procedimientos de las clases que intervienen en el proceso.

No	Subsistema	Clase	Cantidad de Relaciones de Uso
1	OGE	usuarioController	0
2	OGE	usuarioModel	4
3	OGE	rollController	0
4	OGE	rollModel	2
5	OGE	encuestaController	0
6	OGE	encuestaModell	2
7	OGE	noticiaController	0
8	OGE	noticiaModel	2
9	OGE	fuenteController	0
10	OGE	fuenteModel	2
11	OGE	consultarFuenteController	0
12	OGE	consultarFuenteModel	3
13	OGE	autenticarController	0
14	OGE	autenticarModel	1
15	OGE	realizarEncuestaController	0
16	OGE	realizarEncuestaModel	2
17	OGE	mostrarContenidoControler	0
18	OGE	mostrarContenidoModel	1

Tabla 3.7. Datos

Se le halla el promedio de sus asociaciones de uso con respecto a la cantidad de clases:

Total de clases 18

Promedio de asociaciones de uso 1.06

Teniendo los valores anteriores se le aplican los siguientes criterios para obtener en un rango (alta media o baja) de las estadísticas deseadas:

	Categoría	**Criterios**
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mant.	Bajo	<= Prom.
	Medio	Entre Prom. y 2*Prom.
	Alto	> 2*Prom.
Reutilización	Bajo	>2* Prom.
	Medio	Entre Prom. y 2*Prom.
	Alto	<= Prom.
Cantidad de pruebas	Bajo	<= Prom.
	Medio	Entre Prom. y 2*Prom.
	Alto	> 2*Prom.

Tabla 3.82. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Obteniendo el rango de Acoplamiento, Complejidad, Reutilización y Cantidad de pruebas de las relaciones:

Clase	**Acoplamiento**	**Complejidad Mant.**	**Reutilización**	**Cantidad de Pruebas**
usuarioController	Ninguno	Baja	Alta	Baja
usuarioModel	Alto	Alta	Media	Media
rollController	Ninguno	Baja	Alta	Baja
rollModel	Medio	Media	Alta	Baja

encuestaController	Ninguno	Baja	Alta	Baja
encuestaModell	Medio	Media	Alta	Baja
noticiaController	Ninguno	Baja	Alta	Baja
noticiaModel	Medio	Media	Alta	Baja
fuenteController	Ninguno	Baja	Alta	Baja
fuenteModel	Medio	Media	Alta	Baja
consultarFuenteController	Ninguno	Baja	Alta	Baja
consultarFuenteModel	Alto	Alta	Media	Media
autenticarController	Ninguno	Baja	Alta	Baja
autenticarModel	Bajo	Baja	Alta	Baja
realizarEncuestaController	Ninguno	Baja	Alta	Baja
realizarEncuestaModel	Medio	Media	Alta	Baja
mostrarContenidoControler	Ninguno	Baja	Alta	Baja
mostrarContenidoModel	Bajo	Alta	Alta	Baja

Tabla 3.9. Resultados.

Criterio	Categoria	Cantidad de clases	Promedio
0 dependencias	Muy Bueno	9	52,941176
1 dependencias	Bueno	2	11,764705
2 dependencias	Regular	5	29,411764
3 dependencias	Malo	1	5,8823529
> 3 dependencias	Muy Malo	1	5,8823529
Total		18	105,88235

Tabla 3.10. Resultados por criterios de dependencia

Otra de las vías de análisis es de manera gráfica, donde es más fácil apreciar los porcientos que representa cada promedio de dependencia:

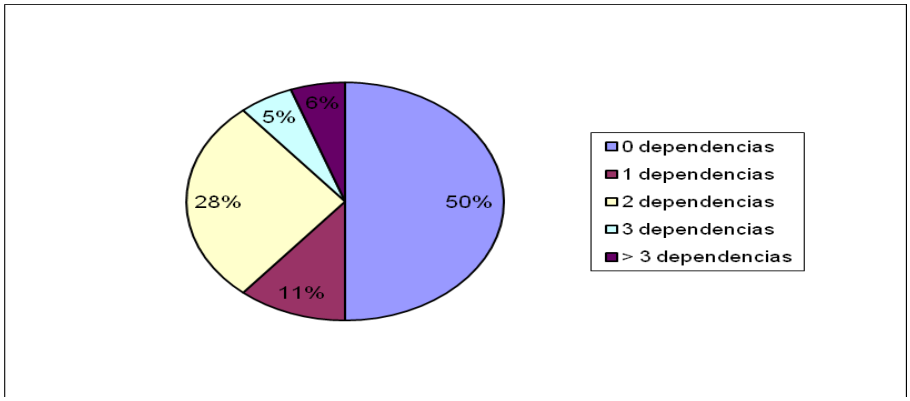


Figura 3.7. Representación en porciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Analizando los resultados de manera independiente se pueden realizar las siguientes apreciaciones:

- Acoplamiento

Tabla 3.12. Acoplamiento

Acoplamiento	Cantidad de clases	Promedio
Ninguno	9	52,94117647
Bajo	1	5,882352941
Medio	5	29,41176471
Alto	2	11,76470588

- Cantidad de pruebas

Tabla 3.14. Cantidad de pruebas

Cantidad de Pruebas	Cantidad de clases	Promedio
Baja	16	88,88888889
Media	2	11,11111111
Alta	0	0

Reutilización

Tabla 3.15. Reutilización

Reutilización	Cantidad de clases	Promedio
Baja	0	0
Media	2	11,11111111
Alta	16	88,88888889

Según los valores obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que el diseño propuesto se encuentra dentro de los niveles aceptables de calidad, mostrando que el 94% de las clases poseen 3 o menos relaciones entre ellas. Los atributos de calidad fueron evaluados satisfactoriamente para el 88% de las clases, confirmando la elevada reutilización y bajo acoplamiento, complejidad y cantidad de pruebas en el diseño propuesto.

3.6 Conclusiones parciales

En este capítulo se realizó la validación de los requerimientos funcionales del sistema, utilizando para ello métricas como la correctitud, completitud y consistencia, de las cuales se obtuvo un resultado favorable. Se aplicaron también técnicas como los prototipos de interfaz de usuario lográndose una mayor comunicación con el cliente durante la captura de requisitos logrando mayor claridad en las posibles respuestas del sistema. Las métricas usadas en la validación del diseño, demostraron que el mismo posee un bajo acoplamiento de las clases y una alta reutilización de las mismas; lo que lo hace sencillo y fácil implementar. Queda así constatada de manera general la calidad del análisis y diseño. Las métricas aplicadas a los elementos del diseño permitieron comprobar la calidad de los mismos y además posibilitó tener una visión de la complejidad que tendrá el proceso de desarrollo del sistema.

Conclusiones generales

Se arribó a las siguientes conclusiones:

- Se realizó un estudio de las tendencias en las disciplinas a desarrollar, las herramientas a utilizar para modelar, la metodología a tener en cuenta, los patrones de casos de uso y de diseño, así como las métricas para evaluar la calidad de la solución propuesta. Esto permitió la preparación teórica que sustenta el desarrollo del trabajo.
- El estudio de los principales conceptos y aspectos relacionados con la plataforma tecnológica para el observatorio, junto a la interacción directa con los clientes del sistema y la aplicación de algunas técnicas para capturar los requisitos, permitió que se obtuvieran resultados satisfactorios en la identificación de las funcionalidades que debe cumplir el sistema.
- La elaboración de los artefactos del modelo del sistema posibilitó un entendimiento común entre desarrolladores y clientes en cuanto a las funcionalidades que el sistema debe brindar.
- La construcción de los artefactos del modelo de diseño facilitó obtener los elementos que deberán ser implementados para que el sistema cumpla con los requisitos especificados.
- Se evaluó la calidad de los principales artefactos generados, los requisitos, los casos de uso del sistema y el diseño, mediante métricas definidas por autores reconocidos y la revisión por parte del proyecto de calidad de la facultad, lo que permitió confirmar la realización confiables y con calidad de dichos artefactos, los cuales garantizaran un entendimiento de las funcionalidades a los desarrolladores permitiendo la futura implementación.

Referencias Bibliográficas

Alanis, Silvana. 2008.*Introducción al gobierno electrónico.* 2008.

Boost Productivity with Innovative and Intuitive Technologies. 2007. Boost Productivity with Innovative and Intuitive Technologies. [En línea] 2007. http://www.visual-paradigm.com/aboutus/10reasons.jsp.

Cortizo Pérez, José Carlos, Expósito Gil, Diego y Ruiz Leyva, Miguel. 2004.*eXtreme Programming.* España : s.n., 2004.

2006. Dirección del Trabajo Gobierno de Chile. *Dirección del Trabajo Gobierno de Chile.* [En línea] 2006. http://www.dt.gob.cl/1601/w3-article-90304).

HispaNetwork Publicidad y Servicios, S.L. 2006. Glosario.net. *Glosario.net.* [En línea] 2006. www.glosario.net.

Jacobson, Ivar, Booch, G y Rumbaugh, J. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid, España : s.n., 2000.

Kendall, K. 1997.*Análisis y Diseño de Sistemas.* 1997.

RATIONAL. www.rational.com. [En línea] http://www.rational.com.

SPARX System Pty L. 2000-2007. SPARX System. [En línea] 2000-2007. http://sparxsystems.com.ar/products/ea.html.

Torres, Francisco. 2006. [En línea] 2006. http://giga4.es/archivos/raiz/moviendoarchivos/2008/febrero/presentacion_tac.pdf.