Universidad de las Ciencias Informáticas

Facultad 1

Título: Repositorio de información científica para la biblioteca de la UCI

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yarima Montenegro Savón

Tutor: Yeneit Delgado Kios

Ciudad de la Habana, junio del 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

"[Insertar nombre(s) de autor(es)]"

_____

"[Insertar nombre(s) de tutor(es)]"

_____

DATOS DE CONTACTO

"[insertar breve curriculum e información de contacto del tutor]"

AGRADECIMIENTOS

A toda mi familia, en especial a mis padres, mi hermana y mis abuelos.

A mi novio por estar siempre apoyándome en todo.

A mis compañeras de cuarto, Elizabeth, Yanet, Yuraimy, Lisett, Jenny y Yusmary.

RESUMEN

El presente trabajo se enmarca en el campo de la investigación de los repositorios de información. Los archivos abiertos conocidos también como repositorios constituyen una alternativa de creciente importancia para la edición de documentos científicos, aprovechando las posibilidades que ofrece Internet para la difusión del conocimiento más allá de las restricciones marcadas por los intereses comerciales. Con la denominación de eprint se engloba de forma genérica a todo tipo de documentos científicos, tesis doctorales, artículos ya publicados, preprints o artículos pendientes de publicación, u otros materiales que pueden ser depositados libremente por los autores en estos repositorios.

El propósito de los repositorios es hacer visible, accesible y recuperable el texto completo de los documentos científicos sobre la especialidad para cualquier usuario potencial con acceso a Internet. Además, con este servicio, especialistas, técnicos y estudiantes, pueden difundir a la comunidad científica internacional sus ponencias y trabajos científicos, estén publicados o no, para fomentar el intercambio de conocimientos y experiencias entre ellos.

En este trabajo además se hace referencia a algunos de los diferentes archivos de acceso abierto existentes en la actualidad y se exponen las características principales de ellos, fundamentalmente de E-LIS que es un archivo de acceso abierto sobre Ciencias de la Información y Documentación.

PALABRAS CLAVE

Acceso abierto, archivo abierto, autoarchivo, repositorios de información.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

El acceso a las fuentes de información científico-técnica se ha convertido en uno de los temas más complejos, atractivos y polémicos de los últimos años en nuestro campo, además constituye la piedra angular de los procesos de generación de nuevos conocimientos científicos, y factor clave en el desarrollo de la sociedad del Siglo XXI. La eliminación de las barreras económicas y de reproducción que supone la restricción del acceso a la información por parte de los principales grupos editoriales de difusión científica, constituye un reto para la comunidad científica internacional.

Para muchos especialistas, el acceso abierto a la literatura científica, específicamente a las publicaciones seriadas, ha de ser una meta a lograr en el ámbito académico, con
vistas a facilitar el acceso a las fuentes de información y a las personas que la generan, así como contribuir a la disminución de la brecha tecnológica existente entre los países desarrollados y aquéllos que no logran, por falta de recursos, desarrollar políticas en ciencia y tecnología con éxito.

Es por ello que numerosas instituciones universitarias han desarrollado estrategias alternativas en aras de lograr una divulgación más rápida e inmediata de su producción científica, y garantizar un mayor impacto de la misma sobre sus correspondientes círculos o comunidades de especialistas afines. Una de estas alternativas es la creación de archivos abiertos (libres de cualquier tipo de restricción), también conocidos como repositorios, donde los científicos, a través del denominado auto archivo, pueden depositar sus artículos incluso previamente antes de su publicación, para que puedan ser difundidos con mayor rapidez y discutidos por la comunidad científica.

Un repositorio de información tiene como objetivos principales garantizar la visibilidad de los autores, facilitar el contacto entre ellos, favorecer la discusión de los trabajos depositados, y contribuir al aumento de las citaciones, y por ende del impacto de los trabajos en la comunidad científica internacional.

Por estas razones, y por ser además herramientas novedosas en pleno desarrollo, es que la comunidad científica debe reconocer la utilidad de los archivos abiertos y no perder la oportunidad de integrarlos a sus procesos investigativos. Su utilización como fuente de información, como vía de divulgación y como

herramienta docente-educativa, necesariamente contribuirá a la unificación de los científicos del mundo entero en comunidades virtuales cada vez más interconectadas.

En el contexto de la Web semántica y bajo el paradigma de la sociedad del conocimiento, las iniciativas para el acceso abierto a la información científica constituyen uno de los principales pasos que la comunidad científica internacional ha dado en aras de la eliminación de la brecha tecnológica existente entre los países ricos y los países pobres del mundo, y en aras también del rescate del sentido humanista de la investigación en Ciencia y Tecnología.

El presente trabajo consta de introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. En el Capítulo 1 Fundamentación teórica, se aborda de forma general los aspectos teóricos mas importantes relacionado con los repositorios de información y sus tendencias actuales. Además se habla de las herramientas y la metodología empleada para el desarrollo de la aplicación. En el Capítulo 2 Características del sistema, se describe el flujo de trabajo de los procesos llevados a cabo para desarrollar la aplicación. Por último en el Capítulo 3 Análisis y diseño del sistema, se realiza el análisis y diseño de la aplicación dándole solución al problema de investigación planteado.

SITUACIÓN PROBLÉMICA

En la biblioteca de la Universidad de las Ciencias Informáticas muchos procesos no están automatizados y con los que se cuentan no satisfacen las necesidades del personal que en él trabajan ni proporciona una interfaz amigable a los usuarios que interactúan con el sistema.

Una de las necesidades actuales de la biblioteca de la universidad es la implementación de un repositorio de información científica que permita a los usuarios de la comunidad subir sus publicaciones al mismo así como consultarlas, que estas publicaciones pueden ser trabajos de diplomas, trabajos científicos, etc. y que además otros usuarios fuera de la comunidad también puedan acceder a las mismas.

PROBLEMA DE INVESTIGACIÓN

¿Cómo realizar un repositorio de información científica para la automatización de este proceso en la biblioteca de la Universidad de las Ciencias Informáticas?

OBJETO DE ESTUDIO

El objeto de estudio de este trabajo son los procesos que se desarrollan en la biblioteca de la Universidad de las Ciencias Informáticas para llevar a cabo un control y uso eficiente de los materiales bibliográficos que se encuentran en la misma.

OBJETIVO GENERAL

Realizar el análisis y diseño del repositorio de información científica para la automatización de este proceso en la biblioteca de la Universidad de las Ciencias Informáticas.

OBJETIVOS ESPECÍFICOS

1. Analizar aspectos teóricos conceptuales de los repositorios de información científica.
2. Realizar un estudio de algunos de los repositorios de información científica existentes.
3. Hacer el análisis y diseño del repositorio.

Con vistas al cumplimiento de los objetivos se propone la realización de las siguientes tareas de investigación:

- Estudio y descripción de algunos de los repositorios de información existentes en la actualidad.
- Selección de la metodología que facilite la creación y garantice la calidad del sistema.
- Selección de las herramientas en la que se desarrollará la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se dará una visión de algunos de los elementos relacionados con los repositorios de información científica, de manera que se responderá la interrogante de ¿qué es un repositorio de información? Se mencionan además algunas de las iniciativas relacionadas con la propuesta de este trabajo. Por último se explica el por qué de la selección de la metodología para el desarrollo de sistemas informáticos y las herramientas empleadas en el presente trabajo.

1.1 Repositorios de información

Los repositorios surgen de la necesidad de compartir recursos que provienen de distintas fuentes y de organizar su almacenamiento de manera que permita potenciar su reutilización.

El tipo de componentes albergados en un repositorio deben tener sus propias identidades y ser por lo tanto localizables, son tan variados como gráficos, imágenes textos, videos, documentos e integración de ellos como capítulos de un curso o hasta cursos completos.

Los repositorios son archivos digitales accesibles a través de Internet que reúnen la producción intelectual de una disciplina o de una institución. Una de las características fundamentales de los repositorios es su carácter abierto e interoperable con otros sistemas.

Los repositorios de información constituyen una de las alternativas más promovidas por la OAI, con el fin de crear espacios donde los científicos, a través de técnicas de autoarchivo, depositen sus artículos de investigación incluso antes de su publicación, para que puedan ser difundidos con mayor rapidez y discutidos por la comunidad científica.

Básicamente, un repositorio es una base de datos accesible libremente a través de Internet mediante la cual se accede a artículos de investigación a texto completo, e implementada bajo un protocolo que posibilita el intercambio de información entre repositorios y maximiza el impacto de los propios documentos en la red [1]. Las características principales de estos repositorios, y que los diferencian ostensiblemente de las bases de datos tradicionales, son las siguientes:

- Los documentos almacenados pueden tener la forma de pre-prints (antes de pasar cualquier proceso de arbitraje) o post-prints (documentos revisados por pares y aceptados, publicados o en proceso de publicación), y pueden ser artículos de revistas, comunicaciones en congresos, capítulos para libros o cualquier otra forma de comunicación científica.

- Los documentos disponibles en los repositorios de información se encuentran a texto completo y el acceso es gratuito, libre de cualquier tipo de restricciones.

- Los propios autores son los responsables de la introducción de los documentos en el repositorio.

Los factores que provocan el crecimiento de un repositorio de información son:

- Accesibilidad: la disponibilidad a texto completo supone que los documentos sean mucho más accesibles, se eliminen las barreras monetarias o de tiempo y se simplifique el caos que algunas veces supone localizar documentos en la red.

- Visibilidad: los documentos se hacen mucho más visibles para la comunidad científica internacional al formar parte de una misma base de datos.

- Diseminación: los trabajos pueden estar en repositorios incluso antes de ser publicados, por lo que la difusión es inmediata, y la reacción de la comunidad científica, de existir, ocurrirá en un plazo breve de tiempo.

- Citación: las tres primeras ventajas conllevan a otra ventaja aún más interesante para un autor. Si un documento de investigación, además de ser interesante o novedoso, está visible en Internet, es de fácil accesibilidad y ha sido difundido con inmediatez, inevitablemente tendrá más probabilidad de ser citado que otro con las mismas características de contenido pero con poca visibilidad o acceso restringido.

Existen en la actualidad diferentes iniciativas que han implementado archivos abiertos dirigidos a los profesionales de las Ciencias de la Información y Documentación.

CCSD: @rchiveSIC
CCSD: MémSIC

Se trata de repositorios desarrollados mediante la colaboración de diferentes instituciones francesas y sus documentos son en francés. @rchiveSIC dispone de 528 documentos. MémSIC de 48 documentos.

DLIST, Digital Library for Information Science and Technology

Iniciativa de la Universidad de Arizona. Sólo acepta documentos en inglés y en la actualidad contiene 354 documentos.

University of North Carolina School of Library and Information

Repositorio dirigido a los estudiantes que realizan las tesis de licenciatura. En la actualidad contiene 110 documentos.

TDR (Tesis Doctorales en Red)

Es un repositorio cooperativo que contiene, en formato digital, las tesis doctorales leídas en las universidades de Cataluña y de otras comunidades autónomas.

Permite la consulta remota a través de Internet del texto completo de las tesis, así como realizar búsquedas por autor/a, director/a, título, tema de la tesis, universidad y departamento donde se ha leído, año de defensa, etc. Los objetivos de este repositorio, son:

- Difundir, por todo el mundo y a través de Internet, los resultados de la investigación universitaria.
- Ofrecer a los autores de las tesis una herramienta que incremente el acceso y la visibilidad de su trabajo.
- Mejorar el control bibliográfico de las tesis.

- Incentivar la creación y el uso de la producción científica propia.

La consulta de las tesis es libre y no necesita ninguna clave de entrada al sistema. Los derechos del autor de la tesis quedan protegidos mediante un contrato. La integridad del texto se garantiza también mediante las opciones de seguridad que incorpora el formato de almacenamiento utilizado: PDF.

El repositorio TDR usa el protocolo de interoperabilidad de la Open Archives Initiative (OAI), lo que permite incrementar la visibilidad de las tesis al ofrecerlas conjuntamente con otros repositorios de tesis internacionales.

E-LIS

E-LIS es el mayor archivo de acceso abierto sobre Bibliotecología y Ciencias de la Información. Su propósito es hacer visible, accesible y recuperable el texto completo de los documentos científicos sobre la especialidad para cualquier usuario potencial con acceso a Internet. Además, con este servicio, especialistas, técnicos y estudiantes, pueden difundir a la comunidad científica internacional sus ponencias y trabajos científicos, estén publicados o no, para fomentar el intercambio de conocimientos y experiencias entre ellos.

La gestión de un repositorio con las características de E-LIS no hubiera tenido éxito si no se hubieran tenido en cuenta aspectos de vital importancia, como sus propias normas de implementación. Las más importantes, están claramente definidas en la interface Web del repositorio:

- Misión: sus metas y objetivos, qué es, hacia dónde se dirige, sus destinatarios y su internacionalización.

- Normas de depósito: quién deposita y cómo lo debe hacer.

- Política de copyright: fundamental para cualquier archivo abierto. Todo trabajo depositado en E-LIS continúa siendo propiedad exclusiva del autor. Los autores que depositen trabajos deben asegurarse de que sobre los documentos autoarchivados no pese ninguna restricción que impida su distribución electrónica.

- Modelo organizativo: lo que supone la esencia de E-LIS y determina su naturaleza disciplinar.

E-LIS ha experimentado un crecimiento exponencial de visitas a partir de su creación en el 2003, y sus más de 4 000 documentos a texto completo especializados en Bibliotecología y Ciencias de la Información creados por más de 1 600 autores de 70 países, reciben más de 100 000 visitas mensuales, por lo que puede afirmarse que la comunidad académica y científica en Bibliotecología y Ciencias de la Información acepta y reconoce el acceso abierto a la literatura científica como una alternativa atractiva e interesante de desarrollo, y por este motivo participan de E-LIS.

1.2 Herramientas para el desarrollo de la aplicación

El software libre cada día da grandes pasos en el mundo de la informática. Dentro del mundo de la Web se encuentran diversos productos como Zope, servidor de aplicaciones Web, Python, lenguaje de programación interpretado e interactivo, capaz de ejecutarse en gran cantidad de plataformas, en el ámbito de los sistemas gestores de bases de datos se encuentra PostgreSQL que está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo y Plone como Sistema de Gestión de Contenido basado en Zope y programado en Python.

1.2.1 Servidores Web

En informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios [2]. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos.

Los archivos para cada sitio de Internet se almacenan y se ejecutan en el servidor. Hay muchos servidores en Internet y muchos tipos de servidores, pero comparten la función común de proporcionar el acceso a los archivos y servicios.

Un servidor sirve información a los ordenadores que se conecten a él. Cuando los usuarios se conectan a un servidor pueden acceder a programas, archivos y otra información del servidor.

Ahora un servidor Web es un programa que corre sobre el servidor, que escucha las peticiones HTTP que le llegan y las satisface [3]. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

Además un servidor Web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

1.2.1.1 Apache

Apache es el servidor Web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

La licencia Apache es una descendiente de la licencias BSD, no es GPL. Esta licencia te permite hacer lo que quieras con el código fuente siempre que les reconozcas su trabajo.

Ahora te preguntarás el por qué de la popularidad de este software libre grandemente reconocido en muchos ámbitos empresariales y tecnológicos, pues aquí algunas razones:

- Corre en una gran multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un modulo para realizar una función determinada.
- Apache trabaja con gran cantidad de lenguajes como Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza parte del pastel de Perl tanto con soporte CGI

como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.

- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor.

El resto de características importantes de Apache son:

Soporte del último protocolo HTTP 1.1

Apache es uno de los primeros servidores Web en integrar el protocolo HTTP 1.1. Es totalmente compatible con el nuevo estándar HTTP 1.1 y al mismo tiempo sigue siendo compatible con HTTP 1.0. Apache esta preparado para todas las novedades del nuevo protocolo. Por ejemplo, antes de HTTP 1. 1, un navegador Web tenia que esperar una respuesta del servidor Web antes de poder emitir otra petición. Con el surgimiento de HTTP 1. 1, esto ha dejado de ser así. Un navegador Web puede enviar solicitudes en paralelo, las cuales ahorran ancho de banda dejando de trasmitir las cabeceras HTTP en cada solicitud. De algún modo estamos recibiendo un estímulo del lado del usuario final porque los archivos solicitados en paralelo aparecerán antes en el navegador.

Sencillo, con la configuración basada en un poderoso archivo

El servidor Apache no posee una interfaz de usuario gráfica para su administración. Se trata de un sencillo archivo de configuración llamado httpd. conf que se puede utilizar

para configurar Apache. Únicamente necesita su editor de texto favorito. Sin embargo, es lo suficientemente flexible para permitirle repartir la configuración de su host virtual en múltiples archivos para no sobrecargar un único archivo httpd. conf con toda la gestión de las múltiples configuraciones de servidores virtuales.

Soporte para CGI (Common Gateway Interface)

Apache soporta CGI utilizando los módulos mod cgi y mod cgid. Es compatible con CGI y aporta características extendidas como personalización de las variables de entorno y soporte de reparación de errores o debugging, que son difíciles de encontrar en otros servidores Web.

Soporte de FastCGI

No todo el mundo escribe sus CGI en Perl. ¿Cómo pueden hacer sus aplicaciones CGI más rápidas? Apache también tiene una solución para esto. Utilice el modulo mod _f cgi para implementar un entorno FastCGI dentro de Apache y haga que sus aplicaciones FastCGl arranquen rápidamente.

Soporte de host virtuales

Apache es además uno de los primeros servidores Web en soportar tanto host basados en IP como host virtuales.

Soporte de autentificación HTTP

Apache soporta autentificación básica basada en la Web. Está también preparado para autentificación basada en la digestión de mensajes, que es algo que los navegadores Web populares ya han implementado. Apache puede implementar autentificación básica utilizando tanto archivos estándar de contraseña como los DBM, llamadas a SQL o llamadas a programas externos de autentificación.

Perl integrado

Perl se ha convertido en el estándar para la programación de scripts CGI. Apache es seguramente uno de los factores que hacen de Perl un lenguaje de programación CGI tan popular. Apache se encuentra más cerca de Perl que nunca. Puede bajar un script CGI basado en Perl a la memoria utilizando su modulo mod perl, y reutilizarlo tantas veces como necesite. Este proceso elimina las desventajas del arranque que se encuentran asociadas a menudo con los lenguajes de interpretación como Perl.

Soporte de scripts PHP

Este lenguaje de script ha comenzado a ser muy utilizado y Apache tiene un amplio soporte para PHP utilizando el modulo mod php.

Soporte de servlets de Java

Los servlets de Java y las Java Server Pages (JSP) se están convirtiendo en algo muy común en los sitios Web dinámicos. Puede ejecutar servlets de Java utilizando el premiado entorno Tomcat con Apache.

Servidor Proxy integrado

Apache puede convertirse en un servidor Proxy caché. Sin embargo, la implementación actual del modulo opcional de Proxy no soporta HTTP Proxy o el ultimo protocolo HTTP 1.1. Se esta planeando actualizar este modulo muy pronto.

Estado del servidor y adaptación de registros

Apache le da una gran cantidad de flexibilidad en el registro y la monitorización del estado del servidor. El estado del servidor puede monitorizarse mediante un navegador Web. Además, puede adaptar sus archivos de registro a su gusto.

Soporte de Server Side Includes (SSI)

Apache ofrece un conjunto de Server Side Includes que añade una gran flexibilidad para el desarrollador del sitio Web.

Soporte de Secured Sockect Layer (SSL)

Puede crear fácilmente un sitio Web SSL utilizando OpenSSL y el modulo mod ssl de Apache.

1.2.1.2 Selección del servidor Web a utilizar

Zope es un servidor de aplicaciones Web orientado a objetos, desarrollado en el lenguaje de programación Python. Puede ser manejado casi totalmente usando una interfaz de usuario basada en páginas Web.

Zope publica en la Web, objetos Python, los cuales persisten usualmente en una base de datos orientada a objetos denominada ZODB; esta base de datos almacena objetos ordenados en un sistema similar a un sistema de ficheros, pero cada objeto tiene propiedades, métodos y puede contener a su vez otros objetos. Objetos básicos como documentos, imágenes, templates, se encuentran disponibles para que el usuario los pueda crear y administrar vía Web. Objetos especiales como wikis, blogs, galería de fotos son provistos como agregados de terceros, denominados productos.

Como se mencionaba anteriormente un sitio Web de Zope está compuesto de objetos en lugar de archivos, como es usual con la mayoría de los otros sistemas de servidores Web. Las ventajas de usar objetos en lugar de archivos son:

- Combinan el comportamiento y los datos en una forma más natural que los archivos de texto plano.
- Alientan el uso de componentes estándares que se ocupan de una parte particular de las que forman una aplicación Web, permitiendo flexibilidad y buena descomposición.
- Posibilitan procesos automáticos de gestión de información.

Además lo que Zope nos proporciona es una plataforma de desarrollo Web robusta, claramente orientada a objetos, y que soluciona gran parte de los problemas cotidianos de los desarrolladores de aplicaciones Web, facilitando enormemente su trabajo. Así, Zope garantiza automáticamente, por citar algunos ejemplos: integridad en las bases de datos, persistencia, control de acceso y herramientas de búsqueda integradas.

A continuación se exponen algunas ventajas de Zope:

- Interfaz de usuario muy sencilla.
- (Casi) todo integrado en uno.
- Fácil instalación y configuración.
- No requiere herramientas de desarrollo específicas ni propietarias.
- Fácil escalabilidad en cuanto número de usuarios y desarrolladores.
- Código abierto.
- Ofrece un modelo de seguridad que permite gestionar cómodamente usuarios, privilegios y delegación de la gestión de contenidos.
- Proporciona persistencia a través de la ZODB (Zope Object DataBase).

Ejemplos de páginas Web realizadas con Zope:
http://www.euskaraz.net
La estructura y desarrollo de la Web se basa en la plataforma Zope, que ha permitido construir una Web dinámica y flexible, con grandes facilidades para su actualización.

http://www.aditel.org

La Web hace uso intensivo de muchas de las funcionalidades que proporciona Zope: programación en ZPT, acceso a base de datos relacionales, integración de productos, catalogación automática del contenido, entre otras.

http://www.gacetadelinux.com

http://www.deli.deusto.es

http://www.guellconsulting.com

1.2.2 Lenguajes de programación para la Web

Lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas [4].

Debe distinguirse de lenguaje informático, que es una definición más amplia, puesto estos incluyen otros lenguajes como son el HTML o PDF que dan formato a un texto y no es programación en sí misma.

El programador es el encargado de utilizar un lenguaje de programación para crear un conjunto de instrucciones que, al final, constituirá un programa o subprograma informático.

Los lenguajes de programación pueden clasificarse según el paradigma que usan en: procedimentales, orientados a objetos, funcionales, lógicos, híbridos, etc.

Son ejemplos de lenguajes de programación: php, prolog, ASP, ActionScript, ada, python, pascal, c, Basic, JAVA, JavaScript, etc.

1.2.2.1 PHP

Es el acrónimo de Hypertext Preprocessor. Es un lenguaje de programación del lado del servidor gratuito, de código abierto, e independiente de plataforma, muy rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML. Su sintaxis es muy parecida a la del lenguaje C, por lo que para cualquier programador que esté familiarizado con dicha sintaxis será fácil aprender a programar en PHP.

El lenguaje PHP tiene la característica de poder mezclarse con el lenguaje HTML. PHP, al contrario que este último, se interpreta y ejecuta directamente en el servidor en el que está albergada la página Web, con lo que el visitante únicamente recibe el resultado buscado por el código en el que está escrito.
La creación y desarrollo de PHP ha estado ligada siempre al proyecto GNU, por lo que al igual que en GNU/Linux o MySQL, el desarrollo de este lenguaje depende de millones de programadores de todo el mundo. Casi la totalidad de servicios ofrecidos por nuestra empresa incluyen el soporte para lenguaje de programación PHP, en sus últimas versiones estables.

Los principales usos de PHP son los siguientes:

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK, lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas de PHP

- Es un lenguaje multiplataforma.
- Muy sencillo de aprender.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- Posee una amplia documentación en su página oficial.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.

- Permite crear los formularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Siempre que se habla de PHP lo primero que se hace es presentar el gran número de gestores de bases de datos a los que puede acceder:

- Adabas D
- dbm
- dBase
- filePro
- Hyperwave
- Informix
- InterBase
- LDAP
- Microsoft SQL Server
- mSQL
- MySQL
- ODBC
- Oracle
- PostgreSQL
- Solid
- Sybase

Pero si este aspecto resulta impresionante no menos el soporte para:

- Acceso a servidores IMAP
- Envío de correo con SMTP
- Acceso a servidores de FTP
- Acceso a SNMP para gestión de redes y equipos
- Generación dinámica de gráficos y documentos PDF
- Análisis de documentos XML

- Corrector de ortografía
- Generación de datos en WDDX (Intercambio Web de Datos Distribuidos)

1.2.2.2 Perl (Practical Extraction and Report Language)

Perl es un lenguaje creado en 1987 por Larry Wall. El objetivo para el cuál se diseña Perl es el disponer de un lenguaje de propósito general que permita simplificar la mayoría de las tareas de administración de un sistema tipo UNIX.

Perl es multiplataforma, podemos encontrar versiones de Perl disponibles para un amplio número de sistemas operativos como Linux, UNIX, MVS, VMS, MS/DOS, Macintosh, OS/2, Amiga.

Perl provee herramientas muy sencillas de uso para la generación de páginas dinámicas vía procesamiento de información de entrada/salida a través del Web. Este hecho masificó el uso de Perl en la creación de scripts para la Web así como un sinnúmero de otras aplicaciones. La utilidad de Perl en la Web ha ido aumentando notoriamente con el tiempo. Su afinidad con las aplicaciones necesitadas en Internet ha hecho sumar esfuerzos en la creación de distintos módulos y bibliotecas orientadas a distintos temas Web.

Algunas de las ventajas del uso del lenguaje Perl son las siguientes:

- Construcción de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas.
- Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2, etc. sin realizar cambios de código, siendo únicamente necesario la introducción del interprete Perl correspondiente a cada sistema operativo.
- También es uno de los lenguajes mas utilizados en la programación de CGI scripts, que son guiones o scripts que utilizan el interfase CGI (Common Gateway Interface), para intercambio de información entre aplicaciones externas y servicios de información.
- El mantenimiento y depuración de un programa en Perl es mucho más sencillo que la de cualquier programa en C.
- El desarrollo de aplicaciones es muy rápido.
- Perl es gratuito. Mucho más que eso, es software libre. Esto quiere decir que el código fuente está disponible para que cualquiera lo pueda ver o modificar.

1.2.2.3 Selección del lenguaje de programación a utilizar

Python es un lenguaje de scripts independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, incluso, páginas Web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

El lenguaje se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.
- La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.
- Además, Python es gratuito, incluso para propósitos empresariales.

Entre las características del lenguaje tenemos:

Propósito general

Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la Web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

Multiplataforma

Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

Interpretado

Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

Interactivo

Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

Orientado a Objetos

La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y librerías

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip

Sintaxis clara

Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

1.2.3 Sistemas Gestores de Bases de Datos (SGBD)

Los sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta [5].

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos.

Los objetivos que deben cumplir los SGBD:

- Abstracción de la información: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- Control de la concurrencia: En la mayoría de entornos lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- Tiempo de respuesta: Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Ventajas:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.
- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- Integridad referencial el terminar los registros.

1.2.3.1 MySQL

MySQL es el sistema de gestión de bases de datos SQL Open Source más popular.
MySQL es un servidor multi-hilos de bases de datos de código abierto, confiable, rápido, compacto, poderoso y multiplataforma podemos hacer las bases de datos a código abierto.
Esta base de datos, lo desarrolla, distribuye y soporta MySQL AB. MySQL AB es una compañía comercial, fundada por los desarrolladores de MySQL. Es una compañía Open Source de segunda generación que une los valores y metodología Open Source con un exitoso modelo de negocio, una gran ventaja es que se puede utilizar gratis y su código fuente siempre esta disponible.

Además de que MySQL es la base de datos de código fuente abierto más usada del mundo, su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del software y una aproximación minimalística para producir características funcionalmente ricas, ha dado lugar a un sistema de administración de la base de datos incomparable en velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del core server del manejador de tablas, permite funcionar a MySQL bajo control estricto de transacciones o con acceso a disco no transaccional ultrarrápido.

MySQL es un sistema de administración de bases de datos

Una base de datos es una colección estructurada de datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto monto de información en una red corporativa. Para agregar, acceder y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos

Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta

Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir que puede hacer y que no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GLP o requiere introducir código MySQL en aplicaciones comerciales, usted pude comprar una versión comercial licenciada.

A continuación se describen algunas de las características más importantes de MySQL**:**

Interioridades y portabilidad

- Escrito en C y en C++
- Probado con un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiples CPUs si están disponibles.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Seguridad

- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites

- Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros. También se conocen usuarios que usan MySQL Server con 60.000 tablas y acerca de 5.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna char, varchar, blob, o text.

Conectividad

- Los clientes pueden conectarse con el servidor MySQL usando sockets TCP/IP en cualquier plataforma.

1.2.3.2 SQL Server

SQL Server es un sistema de gestión de bases de datos relacionales basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Entre sus características figuran:

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos

1.2.3.3 Selección de SGBD a utilizar

Como ya se había comentado anteriormente, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofrece control de concurrencia multi-versión, 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 (que incluye C, C++, Java, perl, tcl y python). PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características:

DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transactions, optimización de consultas, herencia, y arrays.

Altamente-Extensible

PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

Soporte-SQL-Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectarse a PostgreSQL.

Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso

de caída, ya que cualquier cambio que no fue escrito en la base de datos pueden ser recuperado usando el dato que fue previamente registrado. Una vez que el sistema ha quedado restaurado, el usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

Otras características aportan potencia y flexibilidad adicional son:

Restricciones (Constraints)
Disparadores (triggers)
Reglas (rules)
Integridad transaccional

Ahora se va a establecer una comparación entre PostgreSQL y el también gestor de base de datos MySQL.

MySQL

- Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
- Otra característica importante es que consume muy pocos recursos, tanto de CPU como de memoria.

Ventajas:

- Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects.
- Mejores utilidades de administración (backup, recuperación de errores, etc.)
- Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.
- El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro (Barrapunto.com) y de buscadores de aplicaciones (Freshmeat.net).
- No hay límites en el tamaño de los registros.
- Mejor control de acceso, en el sentido de que usuarios tienen acceso a que tablas y con que permisos.

- MySQL se comporta mejor que PostgreSQL a la hora de modificar o añadir campos a una tabla.

Inconvenientes:

- No soporta transacciones, roll-backs ni subselects.
- No considera las claves ajenas. El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
- No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

PostgreSQL

PostgreSQL intenta ser un sistema de bases de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase.

Ventajas:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
- Implementa el uso de roll-backs, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.
- Tiene mejor soporte para triggers y procedimientos en el servidor.
- Soporta un subconjunto de SQL92 mayor que el que soporta MySQL. Además, tiene ciertas características orientadas a objetos.

Inconvenientes:

- Consume gran cantidad recursos y carga más el sistema.

- Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
- Es de 2 a 3 veces más lento que MySQL.
- Menos funciones en PHP.

Como conclusión a la comparación entre MySQL y PostgreSQL, parece aceptado que MySQL junto con Apache y PHP forman un buen equipo para servir páginas Web con contenido dinámico, discusiones, noticias, etc. En general, sistemas en los que la velocidad y el número de accesos concurrentes sea algo primordial, y la seguridad no sea muy importante puede bastar con hacer backups periódicos que se restaurarán tras una caída del servidor. En cambio, para sistemas más serios en los que la consistencia de la base de datos sea fundamental, PostgreSQL es una mejor opción pese a su mayor lentitud.

Además ninguno de estos dos gestores son totalmente perfectos, por lo que no hay que obcecarse en una elección única, como se suele hacer en muchos casos. Simplemente se trata de escoger el más conveniente en cada caso.

1.2.4 Sistema de Gestión de Contenido

¿Qué son exactamente los contenidos?
Cuando se habla de contenido se refiere a la información de cualquier tipo publicada en una Web [6]. Los contenidos atraen visitantes hacia una Web. Está más que comprobado que la inserción y/o renovación periódica de nuevos contenidos provocan un aumento de las visitas a una Web. Las Web de mayor éxito son las que ofrecen contenidos interesantes y atractivos que se renuevan con frecuencia. Los visitantes de dichas Web vuelven con frecuencia a visitarlas porque tienen la expectativa de encontrar nuevos contenidos de interés en cada nueva visita.

Entonces un Sistema de Gestión de Contenido (Content Management System, en inglés, abreviado CMS) permite la creación y administración de contenidos principalmente en páginas Web [7].
Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo

clásico es el de editores que cargan el contenido al sistema y otro de nivel superior que permite que estos contenidos sean visibles a todo público.

Los Sistemas de Gestión de Contenidos permiten renovar los contenidos de una Web de una forma muy fácil a usuarios que desconozcan el lenguaje XHTML. También hacen posible de forma extremadamente sencilla la carga y descarga de archivos de texto o imagen. Estos cambios en los contenidos se realizan accediendo a una página en la que tras introducir un nombre de usuario y una contraseña se encontrará con un menú de opciones muy sencillo e intuitivo (Insertar Modificar, Borrar...) a través del cuál podrá cambiar los contenidos de su Web en cuestión de segundos.

Los Sistemas Gestión de Contenidos ofrecen la posibilidad de realizar las siguientes funciones de forma extremadamente fácil:

- Añadir nuevos contenidos.
- Editar contenidos.
- Borrar contenidos antiguos.
- Mantener una sección de noticias.
- Subir imágenes al servidor.
- Cargar archivos en el servidor (pdf, doc...) y ponerlos a disposición de los usuarios para su descarga.
- Insertar enlaces hacia otras páginas.
- El sistema puede ser mantenido por varias personas que pueden acceder de forma independiente al administrador de contenidos.

Las ventajas de los sistemas de gestión de contenidos son:

- La capacidad de manejar eficientemente gran cantidad de páginas Web.
- Trabajar en un ambiente de páginas Web interactivas, es decir, que se generan según las peticiones de los usuarios.
- Controlar el acceso de los usuarios al sistema, no sólo mediante su contraseña, sino mediante los permisos asignados a cada uno y la información que incluye, tanto en calidad como en cantidad, que posibilita el perfecto crecimiento y desempeño del sistema.
- Orden en el sistema, al existir la posibilidad de asignar, por parte de la herramienta, un mismo estilo a todas las páginas generadas.

1.2.4.1 OpenCms

OpenCms es una herramienta open source profesional, surgida en 1999, que permite la creación de sistemas de gestión de contenidos de forma fácil y por personas sin conocimientos sobre HTML. Como programa open source, OpenCms se encuentra completamente libre de costos y se desarrolla bajo licencia GPL. Dispone de una gran variedad de empresas alrededor del mundo que desarrollan soluciones y ofrecen soporte técnico. Esta herramienta se basa en Java, pero es compatible con casi todas las infraestructuras tecnológicas actuales; además corre perfectamente bien en sistemas operativos y bases de datos open source, como Linux y MySQL, servidores como Apache y en otros privados como Windows NT, Microsoft IIS y Oracle, entre otros.

OpenCms se instala en un servidor Web y los usuarios pueden utilizarlo en sus puestos de trabajo con sus navegadores. Es posible asignar a los usuarios o grupos de trabajo diferentes funciones, controlar el flujo de trabajo del sitio, así como la seguridad de la información que circula. Las páginas y nuevos contenidos también están sujetos a estados que posibilitan su gestión. Existen estados que permiten trabajar con páginas y documentos sin que otros usuarios los vean y hacerlos públicos sólo cuando estén listos; así es posible realizar cambios sin tener que detener el funcionamiento del sitio. Igual proceso puede realizarse con la publicación de noticias y eventos, a los cuales puede agregarse la fecha de publicación y en la que expira, entre otros atributos.

Los contenidos disponibles en un sitio, creado con OpenCms, pueden presentar su fecha de aparición, el creador u otros datos, permitir la discusión y recibir los comentarios realizados por el resto de los usuarios. Este debate en línea, permite evaluar y obtener una retroalimentación sobre los distintos documentos o tareas publicadas.

El entorno de trabajo amigable y fácil que provee OpenCms, además de sus funcionalidades posibilita una eficiente gestión de la información que circula en el sitio. OpenCms se puede trabajar en varios idiomas, entre los que está el español, a partir de la instalación de un programa que se ocupa de la traducción.

1.2.4.2 Typo3

Typo3 es una herramienta para la creación de sistemas de gestión de contenidos open source, distribuido bajo licencia GPL y libre de costo; que permite construir y mantener sitios Web en forma ágil y sencilla. Este programa puede instalarse en servidores Web, Apache o ISS y tiene incluido un sistema de base de datos en MySQL, aunque se puede agregar otras, por medio de extensiones como son Oracle, ODBC,

etcétera. Windows o Mac son los sistemas operativos necesarios para que Typo3 funcione y los usuarios pueden ver los sitios en sus computadoras por medio de los navegadores Internet Explorer, Netscape, Firefox y Opera, entre otros.

El sistema está preparado para asimilar un motor de búsqueda que realiza operaciones, tanto dentro del sitio como en los documentos que se anexen, sea en Word o PDF y que viene con el módulo de instalación. Igualmente, sucede con la posibilidad de publicar noticias, comentarios y discusiones respecto a los contenidos del sitio.

La interfaz de presentación es muy sencilla y permite que cada usuario configure su página principal, según sus gustos y necesidades. Los usuarios tienen funciones asignadas y distintos permisos para manipular la información en el sitio; así, puede gestionarse el flujo de trabajo con total control, tanto de los usuarios individuales y de los grupos como de los documentos que circulan en el sistema y garantizar la seguridad y confiabilidad del sitio. Typo3 permite la configuración en varios idiomas, además de la incorporación de metadatos a los recursos, sea de forma global o para cada página y en distintos idiomas.

1.2.4.3 Drupal

Drupal es un sistema de gestión de contenido para sitios Web. Permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web incluido en el producto.

Características generales de Drupal:

Ayuda online: Un robusto sistema de ayuda online y páginas de ayuda para los módulos del núcleo, tanto para usuarios como para administradores.

Búsqueda: Todo el contenido en Drupal es totalmente indexado en tiempo real y se puede consultar en cualquier momento.

Código abierto: El código fuente de Drupal está libremente disponible bajo los términos de la licencia GNU/GPL. Al contrario que otros sistemas de blogs o de gestión de contenido propietarios, es posible extender o adaptar Drupal según las necesidades.

Módulos: La comunidad de Drupal ha contribuido en muchos módulos que proporcionan funcionalidades como páginas de categorías, autenticación mediante jabber, mensajes privados, etc.

Personalización: Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo las preferencias definidas por el usuario.

Autenticación de usuarios: Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal u otro sitio Drupal. Para su uso en una Intranet, Drupal se puede integrar con un servidor LDAP.

Permisos basados en roles: Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un rol y agrupar los usuarios por roles.

Control de versiones: El sistema de control de versiones de Drupal permite seguir y auditar totalmente las sucesivas actualizaciones del contenido qué se ha cambiado, la hora y la fecha, quién lo ha cambiado, y más. También permite mantener comentarios sobre los sucesivos cambios o deshacer los cambios recuperando una versión anterior.

Enlaces permanentes: Todo el contenido creado en Drupal tiene un enlace permanente asociado a él para que pueda ser enlazado externamente sin temor de que el enlace falle en el futuro.

Objetos de Contenido (Nodos): El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios o no sobre cada objeto.

Plantillas (Templates): El sistema de temas de Drupal separa el contenido de la presentación permitiendo controlar o cambiar fácilmente el aspecto del sitio Web. Se pueden crear plantillas con HTML y/o con PHP.

Sindicación del contenido: Drupal exporta el contenido en formato RDF/RSS para ser utilizado por otros sitios Web. Esto permite que cualquiera con un agregador de noticias, tal como NetNewsWire o Radio UserLand visualice el contenido publicado en la Web desde el escritorio.

Agregador de noticias: Drupal incluye un potente agregador de noticias para leer y publicar enlaces a noticias de otros sitios Web. Incorpora un sistema de caché en la base de datos, con temporización configurable.

Soporte de Blogger API: La API de Blogger permite que un sitio Drupal sea actualizado utilizando diversas herramientas, que pueden ser herramientas Web o herramientas de escritorio que proporcionen un entorno de edición más manejable.

Independencia de la base de datos: Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una capa de abstracción de base de datos que actualmente está implementada y mantenida para MySQL y PostgreSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.

Multiplataforma: Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor Web y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable.

Múltiples idiomas y Localización: Drupal está pensado para una audiencia internacional y proporciona opciones para crear un portal multilingüe. Todo el texto puede ser fácilmente traducido utilizando una interfaz Web, importando traducciones existentes o integrando otras herramientas de traducción como GNU ettext.

Administración vía Web: La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún software adicional.

Análisis, Seguimiento y Estadísticas: Drupal puede mostrar en las páginas Web de administración informes sobre enlaces entrantes, popularidad del contenido, o de cómo los usuarios navegan por el sitio.

Registros e Informes: Toda la actividad y los sucesos del sistema son capturados en un registro de eventos, que puede ser visualizado por un administrador.

Comentarios enlazados: Drupal proporciona un potente modelo de comentarios enlazados que posibilita seguir y participar fácilmente en la discusión sobre el comentario publicado. Los comentarios son jerárquicos, como en un grupo de noticias o un foro.

Encuestas: Drupal incluye un módulo que permite a los administradores y/o usuarios crear encuestas online totalmente configurables.

Foros de discusión: Drupal incorpora foros de discusión para crear sitios comunitarios dinámicos.

Libro colaborativo: Esta característica es única de Drupal y permite crear un proyecto o libro a ser escrito y que otros usuarios contribuyan con el contenido. El contenido se organiza en páginas cómodamente navegables.

Control de congestión: Drupal incorpora un mecanismo de control de congestión que permite habilitar y deshabilitar determinados módulos o bloques dependiendo de la carga del servidor. Este mecanismo es totalmente configurable y ajustable.

Sistema de caché: El mecanismo de caché elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

1.2.4.4 Selección del Sistema de Gestión de Contenido a utilizar

Plone es un Sistema de Gestión de Contenidos o CMS por sus siglas en inglés (Content Management System), basado en Zope y programado en Python. Plone ha sido diseñado para ser extensible. Puede ser empleado para construir portales, sitios Web corporativos, sitio de noticias, servidor de extranet o intranet, sistema de publicación y repositorio de documentos, herramienta groupware, e-commerce, etc.

Las principales características y ventajas de Plone son:

- Fácil de instalar
- Fácil de usar

- Internacional (interfaz en más de 40 idiomas)
- Estandarizado (cumple con los estándares de usabilidad y accesibilidad)
- Open Source (GNU General Public License)
- Existe soporte técnico.
- Extensible.
- Tecnológicamente neutro (compatible con distintos SO, bases de datos…)

Otras ventajas que Plone les proporciona a sus usuarios:

- Producción muy rápida
- Acento en los contenidos y no en la tecnología
- Diseño adaptado por el Web
- Gestión de contenido de localizado
- Edición de las páginas en tiempo real
- Colaboración fácil
- Enfoque centrado en el usuario
- Localización de la interfase en modo nativo

Un sitio Plone posee:

- Distintos tipos de contenido
- Herramientas para administración de usuarios
- Workflows
- Layout y templates predefinidos y personalizables
- Interfaces de administración
- Style sheets
- Buscador en tiempo real
- Soporte multilenguaje
- Políticas de seguridad

Entre los tipos de contenidos provistos por Plone tenemos:

- Documento: presenta información estática al usuario. Es el tipo más común y es muy similar a la típica página Web.

- Ítem de noticia: es un documento que posee campos especiales, como fecha y que se muestra automáticamente dentro del tab de noticias del sitio.

- Enlace: es un link a una url, posee los campos título, descripción y url, el cual puede ser un enlace interno o externo.

- Imagen: imágenes que pueden estar en los formatos digitales, tales como los archivos gif o jpg.

- Eventos: un evento a realizarse, como reuniones, conferencias, etc.

- Carpeta: similar a las de un sistema de archivos, es decir, una carpeta para guardar contenido y proveer un mecanismo para organizar el mismo.

- Archivo: permite almacenar contenidos tales como una película, sonido, texto, hoja de cálculo, archivo comprimido o cualquier otra cosa que se quiera subir al sito Plone.

- Carpetas inteligentes: similares a las carpetas pero se diferencian porque en vez de permitir almacenar contenido dentro de ellas, muestran contenidos resultantes de una búsqueda cuyo criterio es previamente definido.

1.3 Desarrollo de la aplicación en 3 capas

En este trabajo se hará uso de una arquitectura en 3 capas que estas son la capa de presentación, capa de negocios y capa de datos; por ejemplo al conectarnos a Internet estamos navegando en 3 capas, al abrir un formulario Web de inscripción (capa de presentación), después de enviar la información esta es

verificada (capa de negocios) y finalmente la información es grabada en una base de datos (capa de datos).

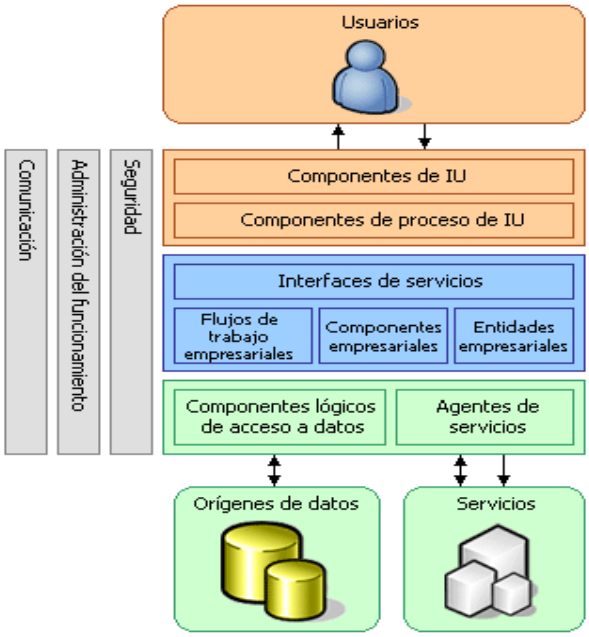

Figura 1. 1 Arquitectura en 3 capas

Capa de Presentación

La capa de presentación es la encargada de interactuar con el usuario y se corresponde con lo que tradicionalmente se conoce como interfaz de usuario. Aquí es donde la aplicación presenta información a los usuarios y acepta entradas o respuestas del usuario para usar por su programa. Idealmente, la interfaz de usuario no desarrolla ningún procesamiento de negocios o reglas de validación de negocios. Por el contrario, la interfaz de usuario debería relegar sobre la capa de negocios para manipular estos asuntos. Esta capa se comunica únicamente con la capa de negocio.

Una de las mayores dificultades y factores importantes cuando desarrollamos aplicaciones cliente/servidor es mantener una separación completa entre la presentación, la lógica de negocios y los servicios de datos. Es muy tentador para los desarrolladores mezclar una o más capas; poniendo alguna validación u otro proceso de negocios dentro de la capa de presentación en vez de en la capa de negocios.

Capa de Negocios

Toda aplicación tiene código para implementar reglas de negocios, procesos relacionados a los datos o cálculos y otras actividades relativas a los negocios. Colectivamente este código es considerado para formar la capa de negocios. Otra vez, uno de los principios del diseño lógico cliente/servidor, la lógica de negocios debe mantenerse separada de la capa de presentación y de los servicios de datos. Esto no significa necesariamente que la lógica de negocios está en cualquier parte, por el contrario, esta separación es en un sentido lógico.

Hay muchas formas de separar la lógica de negocios. En términos orientados a objetos, se debería encapsular la lógica de negocios en un conjunto de objetos o componentes que no contienen presentación o código de servicios de datos. Teniendo separada lógicamente su lógica de negocios de ambas, la capa de presentación y servicios de datos, se ganará en flexibilidad en término de que se puede almacenar físicamente la lógica de negocios. Por ejemplo, puede seleccionarse almacenar la lógica de negocios sobre cada estación de cliente, u optar por ejecutar la lógica de negocios sobre un servidor de aplicaciones, permitiendo a todos los clientes acceder a un recurso centralizado.

Los objetos de negocios son diseñados para reflejar o representar sus negocios. Ellos se convierten en un modelo de sus entidades de negocios e interrelaciones. Esto incluye tanto objetos físicos como conceptos abstractos. Estos son algunos ejemplos de objetos del mundo real: un empleado, un cliente, un producto, una orden de compra.

Todos estos son objetos en el mundo físico, y la idea en su totalidad detrás de usar objetos de negocios de software, es crear una representación de los mismos objetos dentro de la aplicación. Las aplicaciones pueden hacer que estos objetos interactúen unos con otros como ellos lo hacen en el mundo real. Por ejemplo, un empleado puede crear una orden de compra a un cliente que contiene una lista de productos. Siguiendo esta lógica se puede crear objetos de negocios de una orden conteniendo el código necesario para administrarse a si mismo, así nunca necesitará replicar código para crear ordenes, solo usará el objeto. Similarmente, un objeto cliente contiene y administra sus propios datos. Un buen diseño de un objeto cliente contiene todos los datos y rutinas necesitadas para representarlo a través del negocio completo, y puede ser usado a través de toda la aplicación de ese negocio.

No toda la lógica de negocio es la misma. Alguna lógica de negocio es un proceso intensivo de datos, requiriendo un eficiente y rápido acceso a la base de datos. Otras no requieren un frecuente acceso a los datos, pero es de uso frecuente por una interfaz de usuario robusta para la validación en la entrada de campos u otras interacciones de usuarios.

Capa de Datos

La capa de datos es donde residen los datos. Está formada por uno o más gestores de bases de datos que realiza todo el almacenamiento de los datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no sería lo normal), lo más usual es que haya una multitud de ordenadores donde resida la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o mas ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos.

1.4 Metodología para el desarrollo de la aplicación

Influenciado por el gran desarrollo de la computación y el uso creciente de Internet se construyen software más grandes y complejos. Se requiere de un software construido en el menor tiempo posible y que sea eficiente. El Proceso Unificado de Desarrollo es una solución al problema del software.

El Proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software [8].

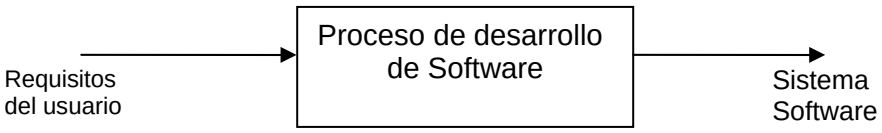

Figura 1. 2 Un proceso de desarrollo de software

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software. De hecho, UML es una parte esencial del Proceso Unificado.

No obstante las características fundamentales del Proceso Unificado son: está dirigido por casos de uso, está centrado en la arquitectura y es iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

Proceso dirigido por casos de uso

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

En RUP los casos de uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los casos de uso constituyen un elemento integrador y una guía del trabajo como se muestra en la figura 1.3.

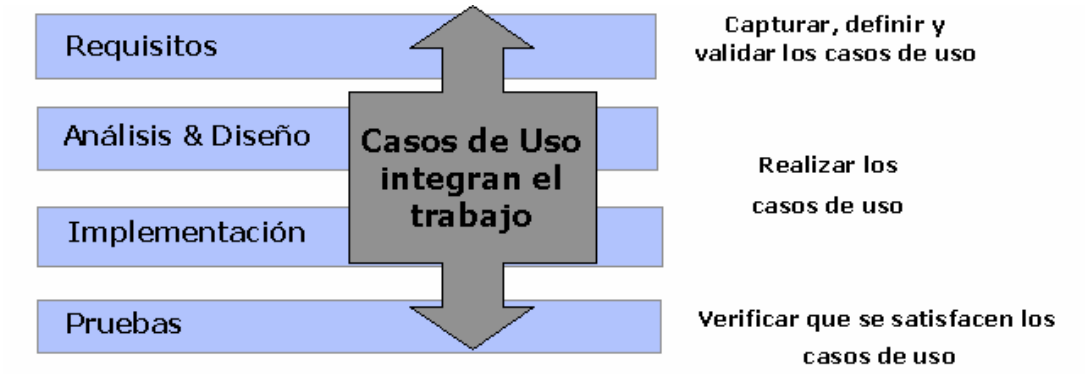

Figura 1. 3 Los casos de uso integran el trabajo

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

Basándonos en los casos de uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada caso de uso. Todos los modelos deben estar sincronizados con el modelo de casos de uso.

Proceso centrado en la arquitectura

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

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

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

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

En la figura 1.4 se ilustra la evolución de la arquitectura durante las fases de RUP. Se tiene una arquitectura más robusta en las fases finales del proyecto. En las fases iniciales lo que se hace es ir

consolidando la arquitectura por medio de una línea base y se va modificando dependiendo de las necesidades del proyecto.

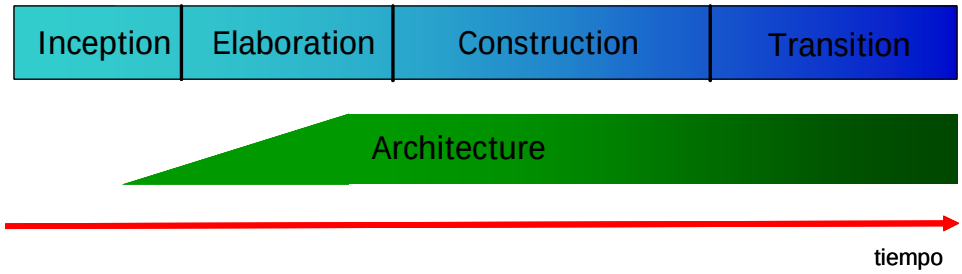

Figura 1. 4 Evolución de la arquitectura del sistema

Proceso iterativo e incremental

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

Una iteración puede realizarse por medio de una cascada como se muestra en la figura 1.5. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

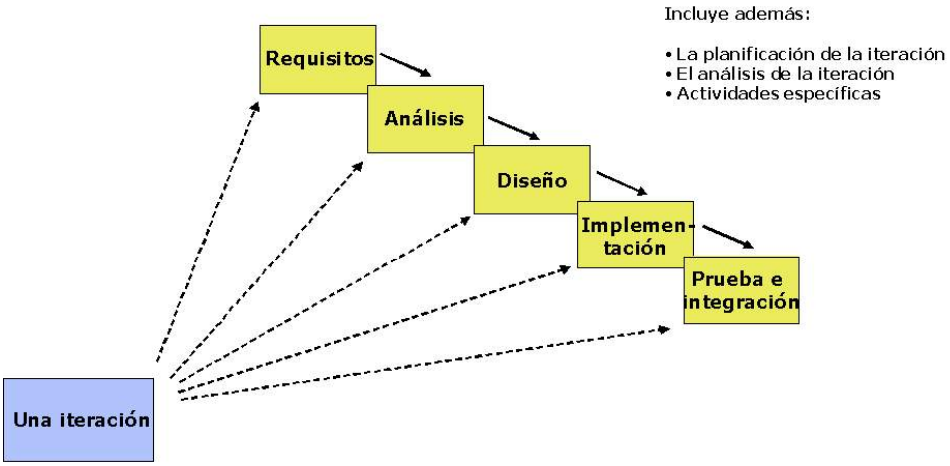

Figura 1. 5 Una iteración RUP

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

1.5 Lenguaje empleado para el desarrollo de la aplicación

Como lenguaje de modelado se utilizara UML que es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.
La decisión de utilizar UML como notación para la aplicación se debe en primer lugar a que la metodología utilizada en nuestro proyecto es RUP, por lo que ambos poseen una estrecha relación; por otra parte se ha convertido en un estándar de facto que tiene las siguientes características:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).

- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Cubre las cuestiones relacionadas con los tamaños propios de los sistemas complejos y críticos.
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.
- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

1.6 Herramienta CASE de modelado con UML

Las herramientas CASE (Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero [9].

A medida que los sistemas que hoy se construyen se tornan más y más complejos, las herramientas de modelado con UML ofrecen muchos beneficios para todos los involucrados en un proyecto, por ejemplo, administrador del proyecto, analistas, arquitectos, desarrolladores y otros. Las herramientas CASE de modelado con UML nos permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraernos del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Cuanto más grande es un proyecto, es más importante utilizar una herramienta CASE. Al usar las herramientas CASE:

- Los Analistas de Negocio/ Sistemas pueden capturar los requisitos del negocio/sistema con un modelo de casos de uso.

- Los Diseñadores/Arquitectos pueden producir el modelo de diseño para articular la interacción entre los objetos o los subsistemas de la misma o de diferentes capas (los diagramas UML típicos que se crean son los de clases y los de interacción).
- Los Desarrolladores pueden transformar rápidamente los modelos en una aplicación funcionando, y buscar un subconjunto de clases y métodos y asimilar el entendimiento de cómo lograr interfaces con ellos.

1.6.1 Visual Paradigm

Es una herramienta CASE que utiliza UML como lenguaje de modelado.
Características:
Producto de calidad.
Soporta aplicaciones Web.
Las imágenes y reportes generados, no son de muy buena calidad.
Varios idiomas.
Generación de código para Java y exportación como HTML.
Fácil de instalar y actualizar.
Compatibilidad entre ediciones.

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

1.6.2 Herramienta CASE a utilizar

Rational Rose es una herramienta para el modelado visual, que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida de desarrollo de software.
Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Desarrollo Iterativo
Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.
Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Trabajo en Grupo
Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.
También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Generador de Código
Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Ingeniería Inversa
Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

Características del Rational Rose:

- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.
- Genera la documentación automáticamente.
- Generación de código a partir de los modelos.
- Ingeniería inversa (crear modelo a partir de código).

Conclusiones del capítulo

En este capítulo fue expuesto el concepto asociado al dominio del problema. Se hizo un análisis de algunos de los repositorios de información existentes. Por otro lado, se expusieron las características principales de las herramientas propuestas para el desarrollo de la aplicación, así como las ventajas y los inconvenientes de las mismas. De igual forma, las características del Proceso Unificado demuestran las ventajas de elegir esta metodología.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Un sistema, sin importar el tamaño que posea, tiende a ser complejo, por lo que se hace necesario dividirlo en piezas para un mejor entendimiento. Esas piezas pueden ser representadas a través de modelos que permitan abstraer sus características esenciales. Es por esta razón que en el campo del software también resulta sumamente útil la creación de modelos que organicen el trabajo y brinden una idea de lo que se quiere lograr. En este capítulo se describirá el flujo de trabajo de los procesos llevados a cabo para desarrollar el sistema, dándole solución al problema de investigación planteado. Primero se realiza un modelo del dominio, donde se capturan las clases más importantes en el contexto del sistema. Se hace el levantamiento de los requisitos funcionales y no funcionales, donde los primeros se estructuran mediante los casos de uso del sistema, de los cuales se ofrece una descripción textual.

2.1 Modelo del dominio

Hay por lo menos dos aproximaciones para expresar el contexto de un sistema en una forma utilizable para los desarrolladores de software: modelado del dominio y modelado del negocio. Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que estén o los eventos que suceden en el entorno en el que trabaja el sistema.
Muchas de las clases del dominio pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio.

2.1.1 Descripción del problema del dominio

El sistema repositorio de información científica permite disponer de un lugar común donde profesores y estudiantes almacenen sus trabajos de investigación. Se desea que el sistema sea accesible a través de un entorno Intranet.
El sistema presenta en su pantalla principal un mensaje de bienvenida describiendo los servicios que se ofrecen al usuario junto con la opción de registrarse, o si ya esta registrado, poder utilizar el sistema. Este acceso se da por medio de la inserción de un nombre de usuario y una contraseña previamente escogidos y que deben validarse.
Una vez registrado el usuario, y después de haberse validado su acceso en el sistema, puede seleccionar las siguientes actividades:

Subir información

Permite a los usuarios subir información científica especificando la categoría en la que incluye esta información.

Se debe depositar el documento para que pueda ser publicado en el repositorio. Los documentos depositados no aparecerán hasta que hayan sido revisados por el editor del archivo.

Actualizar información

Los usuarios pueden actualizar la información que han aportado al repositorio. El sistema brinda la posibilidad de cambiarle la categoría, especificar si están disponibles o no para el resto de los usuarios, así como borrarlos definitivamente.

Buscar información

Asiste a los usuarios en la búsqueda de información, que puede restringirse especificando una serie de criterios con los que deben cumplir la información que espera encontrar.

El sistema ofrecerá además a los usuarios la posibilidad de emitir juicios valorativos, ya sea cualitativa o cuantitativamente acerca de la información consultada.

2.1.2 Diagrama de clases del dominio.

Ver Anexo I.

2.2 Requisitos

2.2.1 Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

1. Registrar usuario
2. Autenticar usuario
3. Actualizar usuario
4. Subir información
5. Buscar información
6. Consultar información
7. Actualizar información
8. Emitir comentario de información
9. Evaluar información
10. Publicar información

2.2.2 Requisitos 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, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto.

Requerimientos de hardware

- Conectividad Red de Área Local(LAN)
- Respaldo eléctrico(UPS)
- Microprocesador 200 MHz
- 32 MB de memoria RAM
- 4 GB de disco duro

Apariencia o interfaz externa

Se requiere de una interfaz agradable, legible, sencilla de usar, autoritaria para que los usuarios se sientan confiados y profesional.

Usabilidad

Dentro del marco de los usuarios que necesitarán del uso de este sistema se encuentran estudiantes que comienzan en primer año de la carrera pero que aún no poseen suficiente experiencia en el manejo de la computadora, los empleados de la biblioteca y otra serie de trabajadores del centro que no necesariamente dominan al detalle el funcionamiento de las mismas, por esta razón se trata que la interfaz sea amigable y clara para lograr una mejor comprensión del sistema.

Rendimiento

Se desea un sistema eficiente con un gran nivel de precisión, de respuestas rápidas, que procese la información a una velocidad que cumpla las expectativas del cliente y que esté disponible las 24 horas del día.

Soporte

Tanto la instalación como el mantenimiento del sistema deben efectuarse de manera sencilla, sin mayores complicaciones.

Portabilidad

El sistema será implementado sobre Linux, pero tendrá la posibilidad de migrar hacia otras plataformas sin presentar mayores complicaciones para lograrlo.

Seguridad

Se ha concebido un sistema que aporte confidencialidad**,** es decir**,** que la información manejada por el mismo esté protegida de acceso no autorizado y divulgación, ya que todos los usuarios no pueden acceder a las distintas funcionalidades del sistema. Debe brindar además disponibilidad, lo cual es similar a decir que a los usuarios autorizados les será garantizado el acceso a la información pertinente y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para

obtener los datos deseados en un momento dado. La información manejada será tratada con severa minuciosidad para no corromper su integridad.

Político-culturales

El producto podrá ser utilizado por aquellas empresas o entidades con relaciones con la universidad, dada la aprobación de esta para su uso.

Legales

El software cumple con las leyes vigentes en la Universidad y en el país. Este documento y la aplicación pertenecen al Proyecto UCI-Ciudad Digital, específicamente a las Áreas que comprende la biblioteca de la misma.

Requerimientos de Confiabilidad

Se deben realizar periódicas salvas de la base de datos para garantizar que siempre esté disponible el acceso a esta.

Interfaz de software

El análisis y el diseño del sistema se realizará haciendo uso de la metodología RUP, el lenguaje de modelación UML y como herramienta Rational Rose. El lenguaje de programación que se ha usado es el Python, y como gestor de base de datos se utiliza el PostgreSQL

2.3 Actores del sistema

Los actores pueden ser personas, software o hardware; el término actor representa el rol genérico de usuario del sistema. Identificar los actores nos permite: definir los límites del sistema (qué forma parte del sistema y qué no) y desarrollar un sistema orientado al usuario que contemple todas las funcionalidades esperadas por los diferentes actores. Ver Anexo II.

2.4 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. Ver Anexo II.

2.5 Descripción de los casos de uso

Ver Anexo II.

Conclusiones del capítulo

En este capítulo se mostraron las principales clases del dominio del problema para un mayor entendimiento del negocio y se describió la solución que se le da al problema de investigación, se mostró la vista lógica y física de la aplicación así como los datos que se representan en ella.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se detallan las características del análisis y diseño del sistema, así como la interacción que existe entre todos sus componentes finales.

3.1 Clases de análisis

Las clases de análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación que pudieran ser de agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en clases de interfaz, de control o de entidad.

Las clases de entidad modelan información que posee larga vida y que es a menudo persistente.

Figura 3. 1 Clase entidad

Las clases de interfaz modelan la interacción entre el sistema y sus actores.

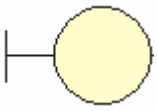

Figura 3. 2 Clase interfaz

Las clases de 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.

Figura 3. 3 Clase controladora

En una aplicación cliente/servidor de tres capas, en la capa de usuario aparecen fundamentalmente clases interfaz ya que allí se ejecutan las aplicaciones del cliente. En la capa intermedia están las clases de control ya que en ella se agrupan los servicios que son compartidos por múltiples aplicaciones. En la tercera capa estarían las clases entidad porque allí se tiene la base de datos.

3.1.1 Modelo de clases de análisis

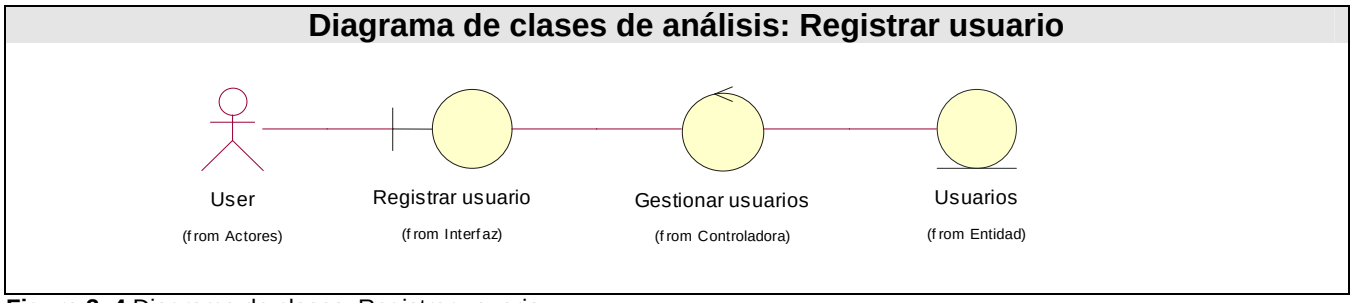

Figura 3. 4 Diagrama de clases: Registrar usuario

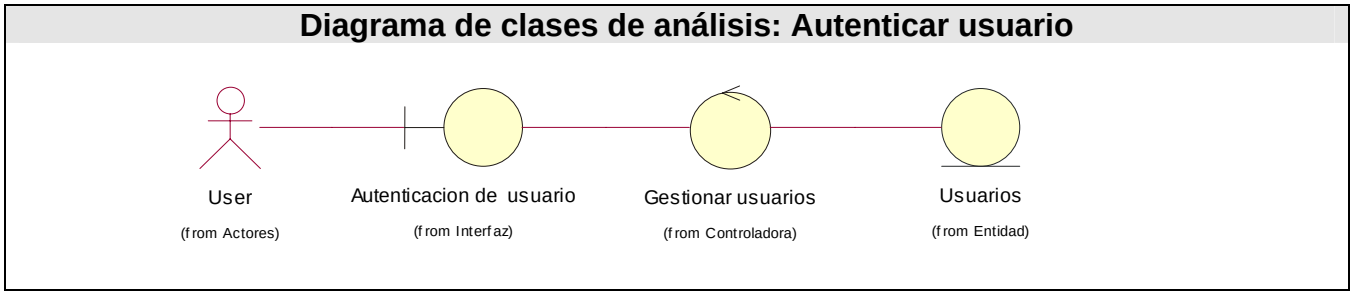

Figura 3. 5 Diagrama de clase: Autenticar usuario

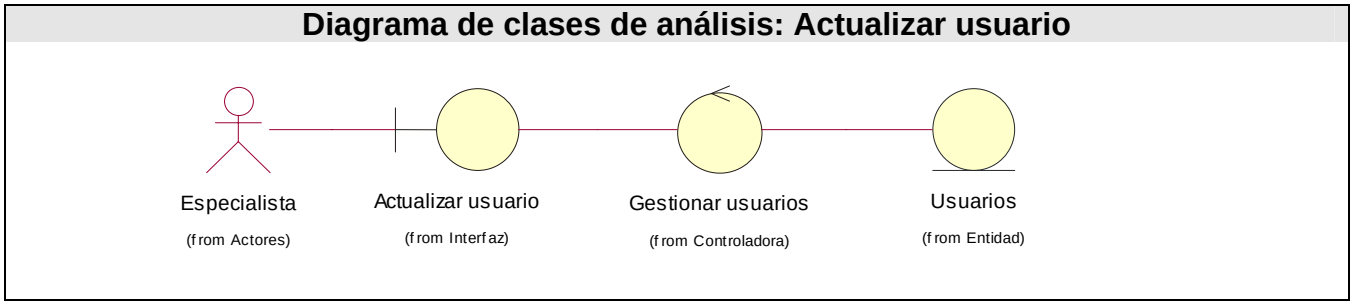

Figura 3. 6 Diagrama de clase: Actualizar usuario

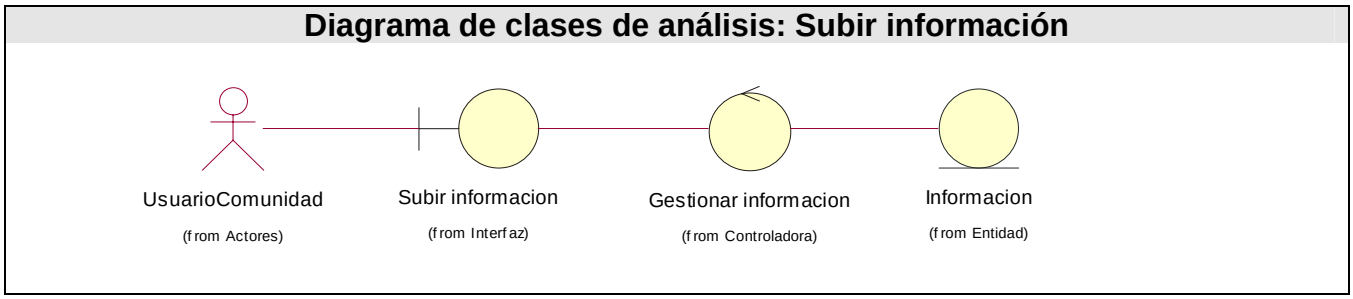

Figura 3. 7 Diagrama de clase: Subir información

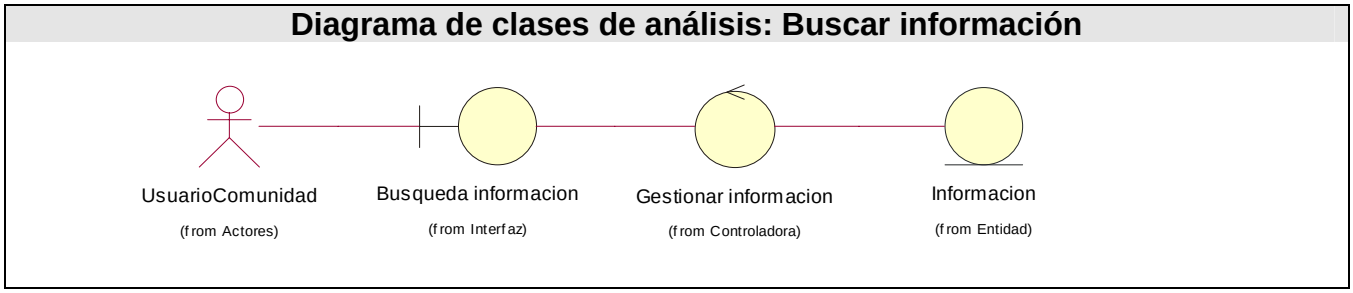

Figura 3. 8 Diagrama de clase: Buscar información

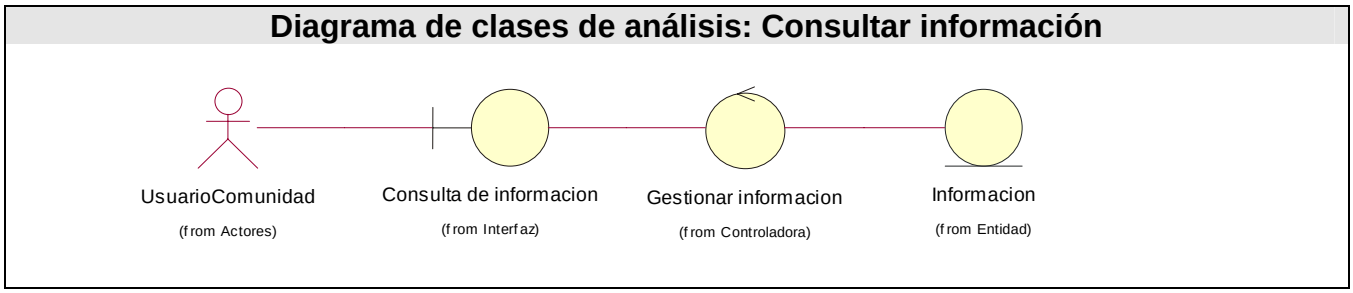

Figura 3. 9 Diagrama de clase: Consultar información

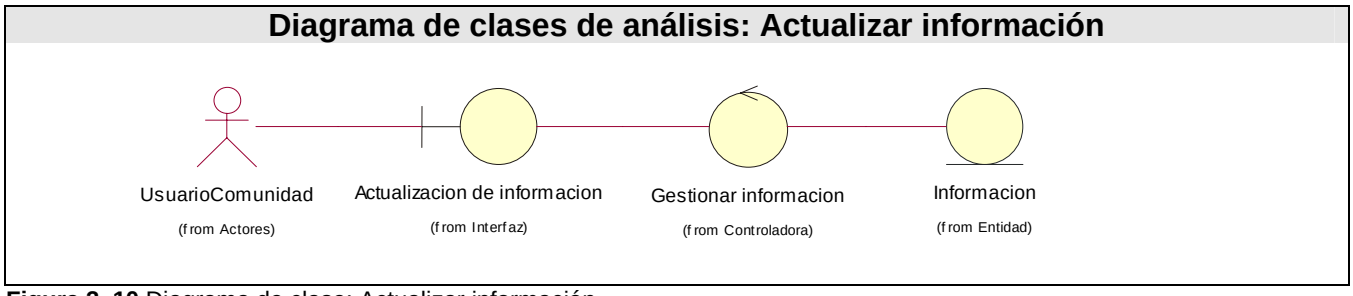

Figura 3. 10 Diagrama de clase: Actualizar información

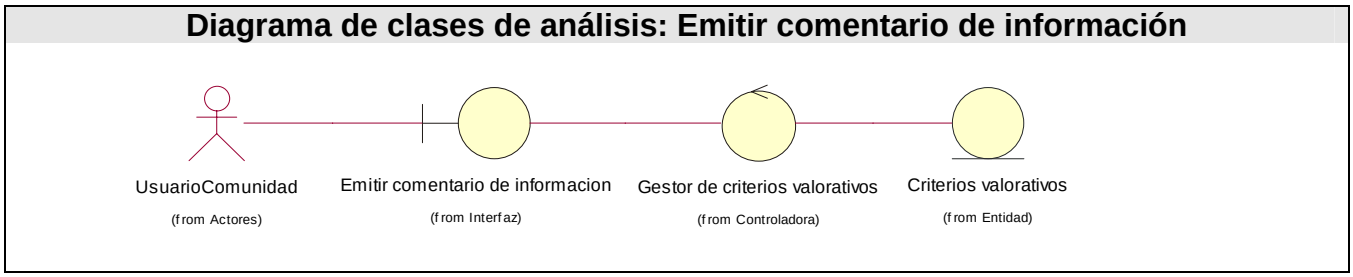

Figura 3. 11 Diagrama de clase: Emitir comentario de información

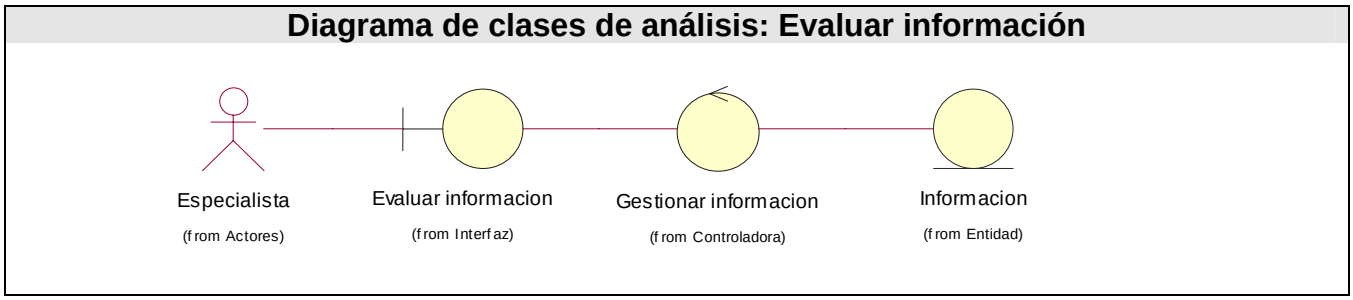

Figura 3. 12 Diagrama de clase: Evaluar información

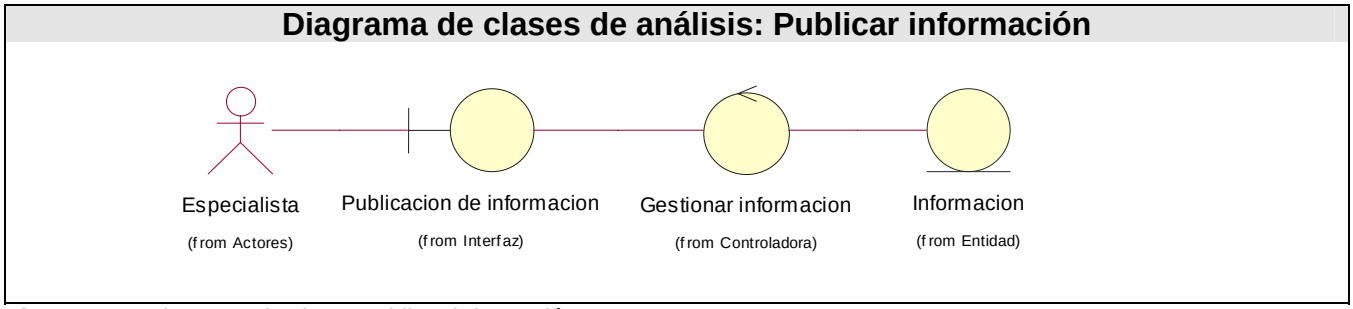

Figura 3. 13 Diagrama de clase: Publicar información

3.1.2 Diagramas de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Los diagramas de interacción pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso.

Los diagramas de interacción no son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa.

Los diagramas de interacción están compuestos por diagramas de colaboración y diagramas de secuencia, en este caso se tendrán en cuenta los diagramas de colaboración. Un diagrama de colaboración destaca la organización de los objetos que participan en una interacción. Un diagrama de colaboración se construye colocando en primer lugar los objetos que participan en la colaboración como nodos del grafo. A continuación se representan los enlaces que conectan esos objetos como arcos del grafo. Por último, estos enlaces se adornan con los mensajes que envían y reciben los objetos.

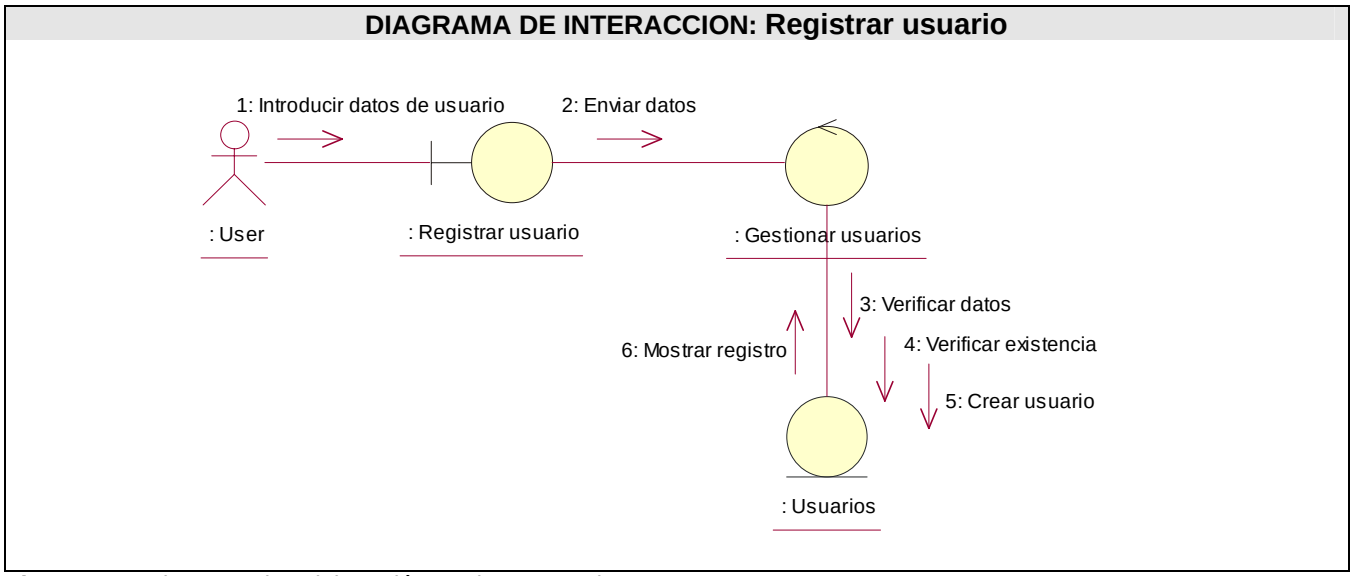

Figura 3. 14 Diagrama de colaboración Registrar usuario

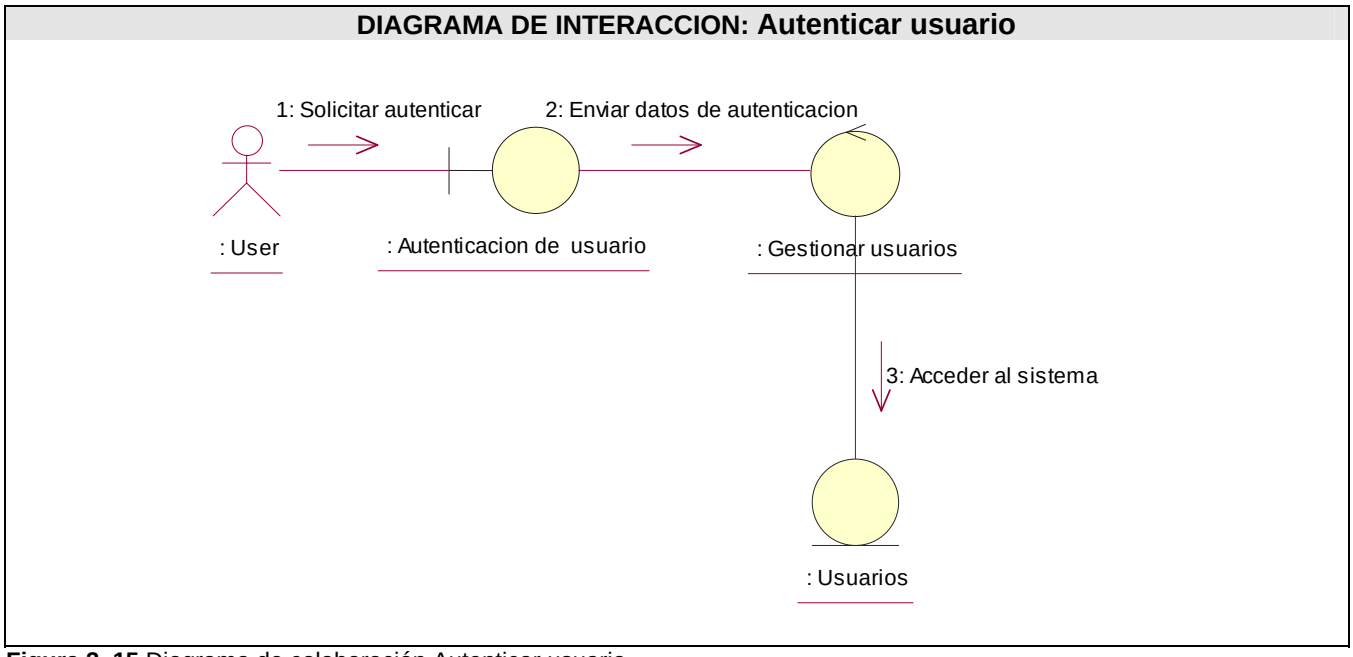

Figura 3. 15 Diagrama de colaboración Autenticar usuario

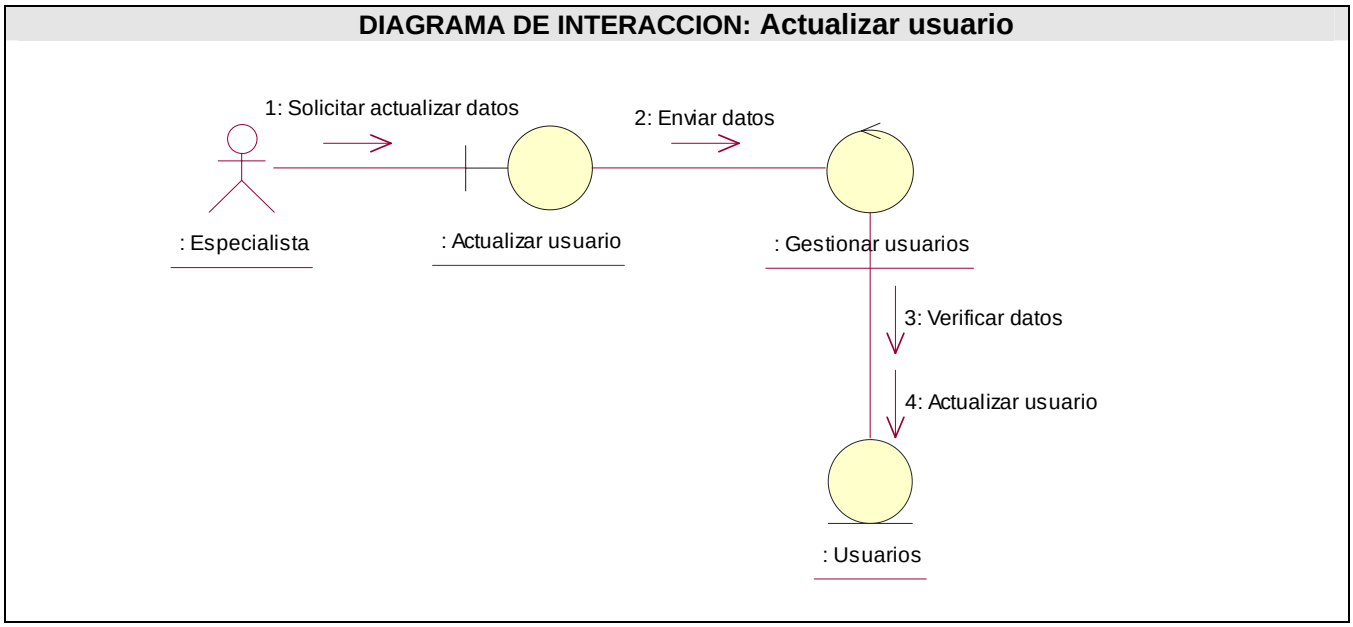

Figura 3. 16 Diagrama de colaboración Actualizar usuario

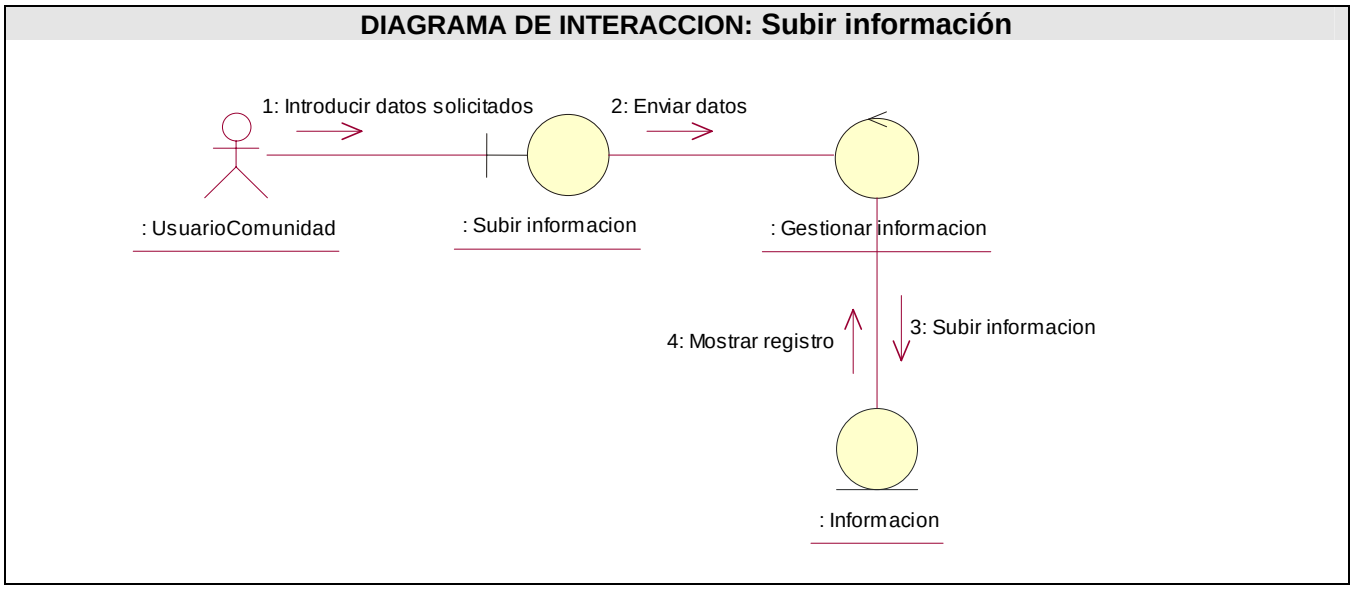

Figura 3. 17 Diagrama de colaboración Subir información

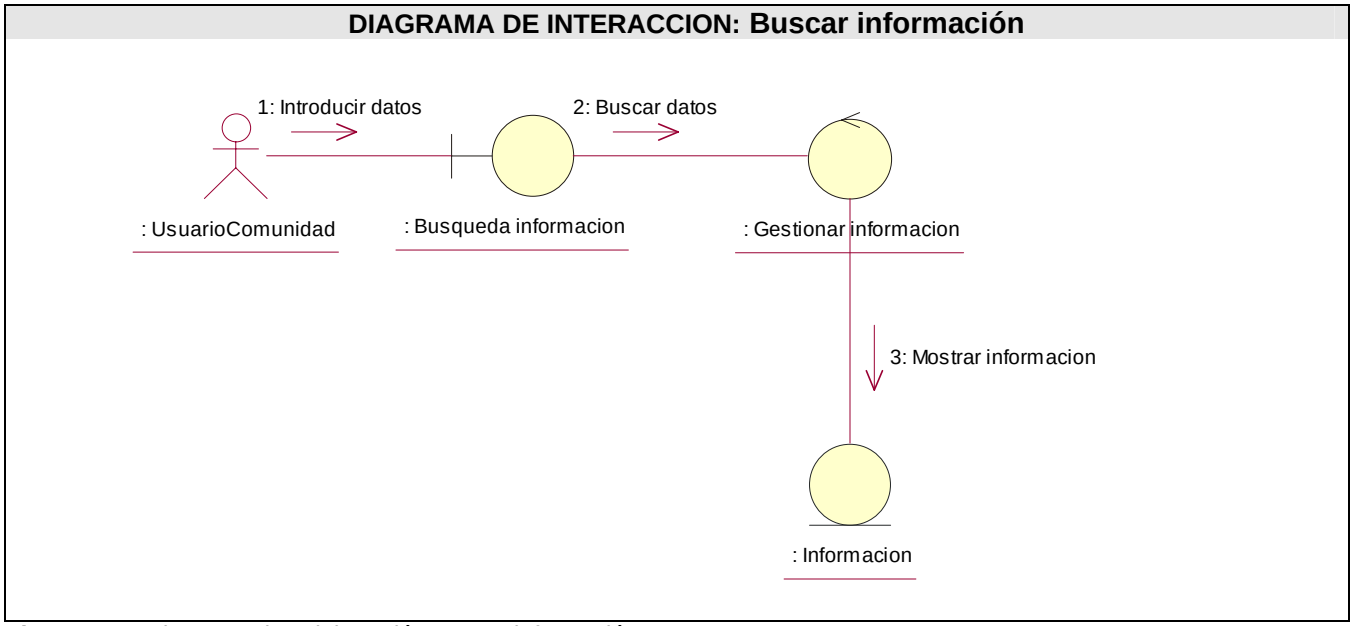

Figura 3. 18 Diagrama de colaboración Buscar información

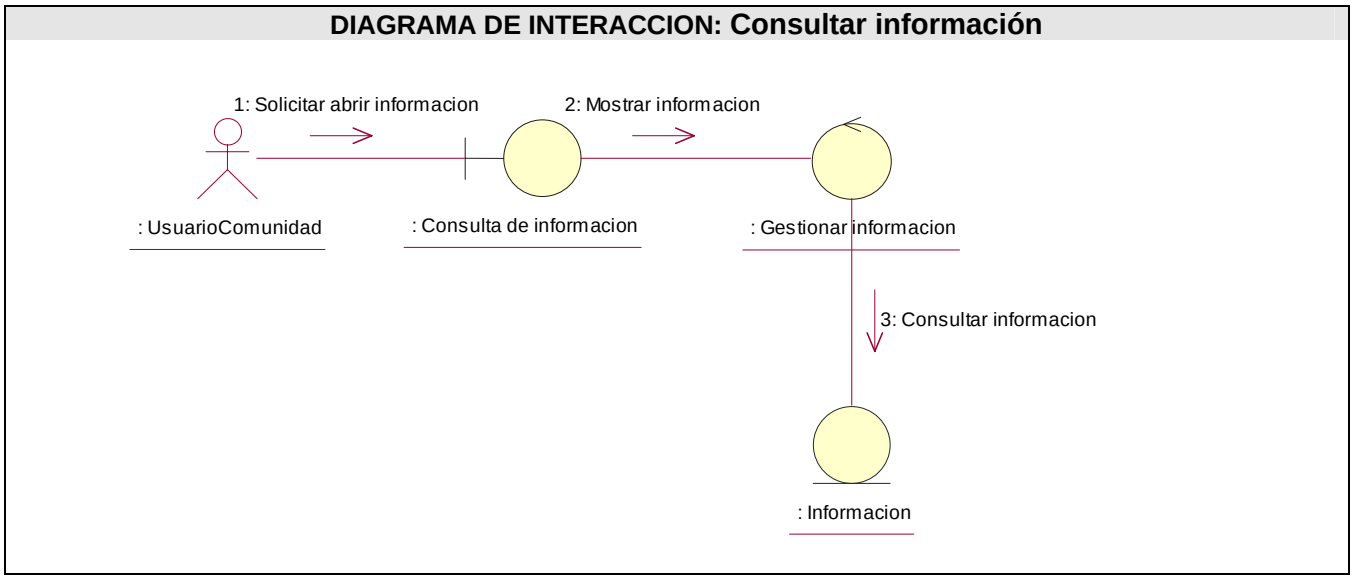

Figura 3. 19 Diagrama de colaboración Consultar información

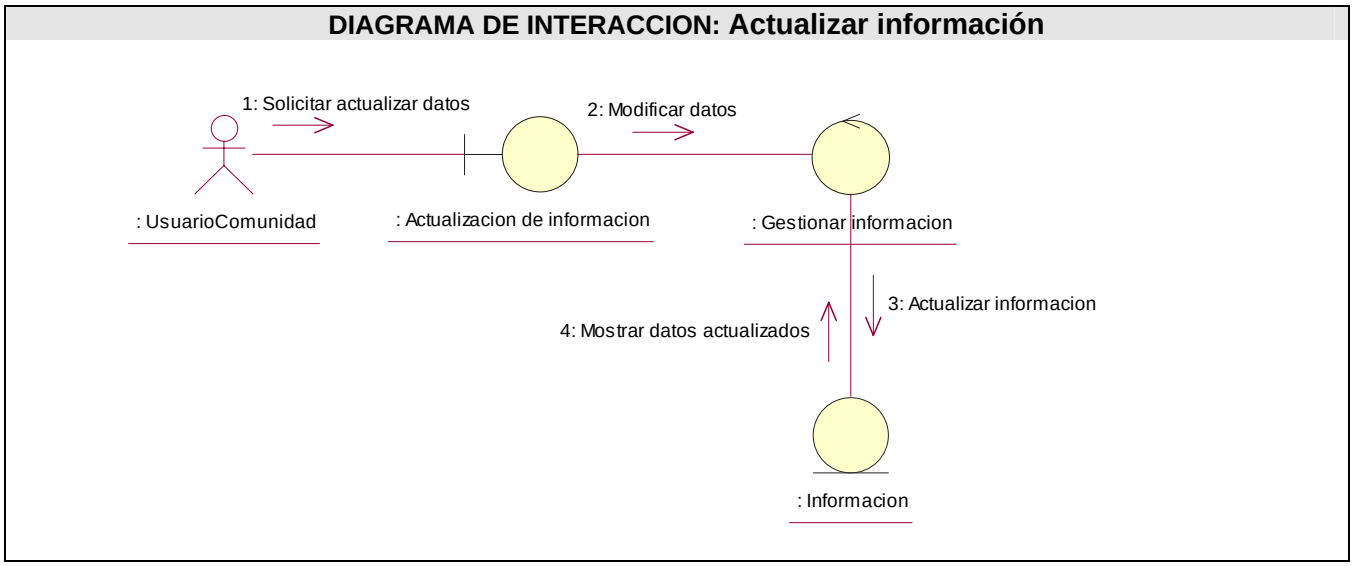

Figura 3. 20 Diagrama de colaboración Actualizar información

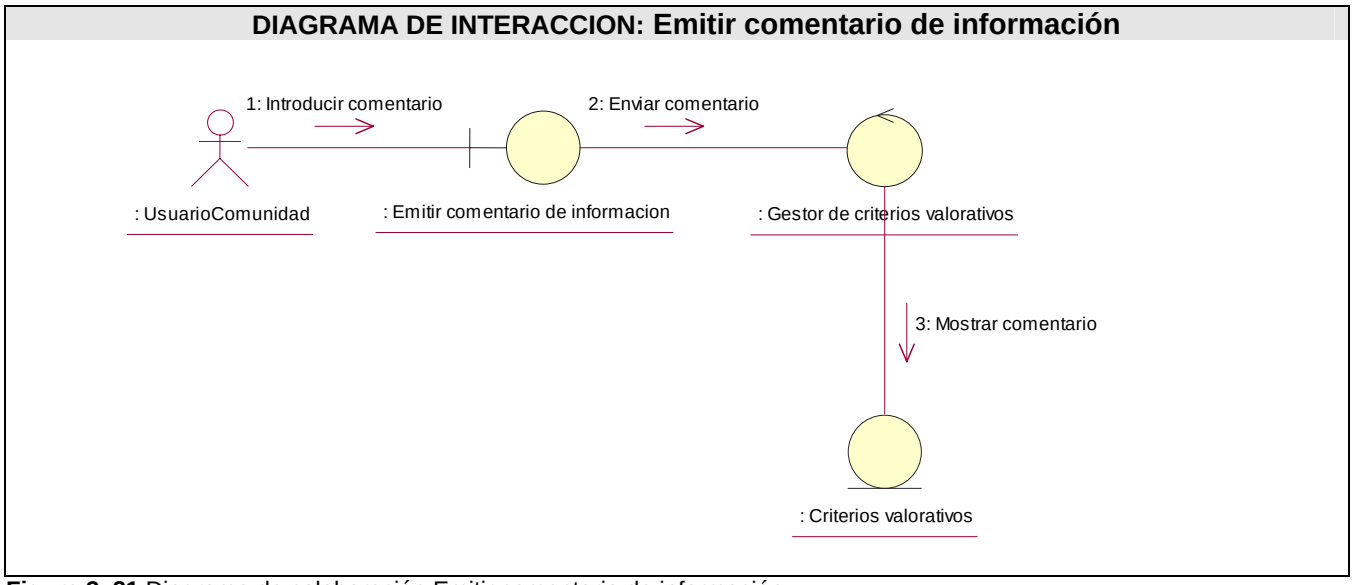

Figura 3. 21 Diagrama de colaboración Emitir comentario de información

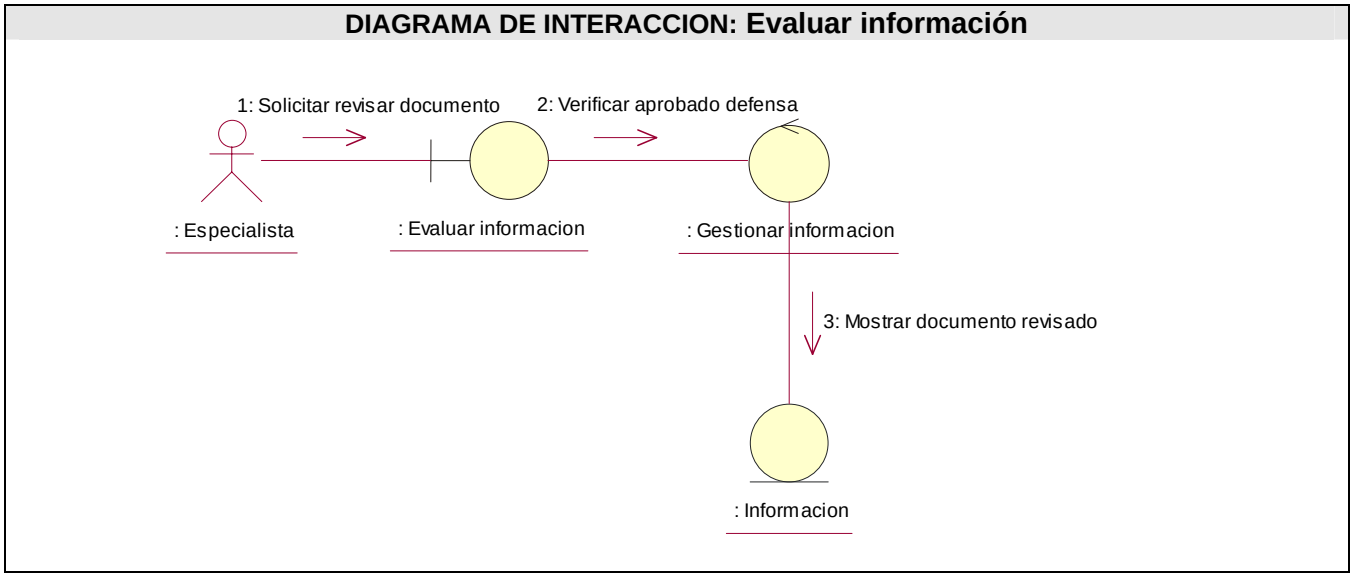

Figura 3. 22 Diagrama de colaboración Evaluar información

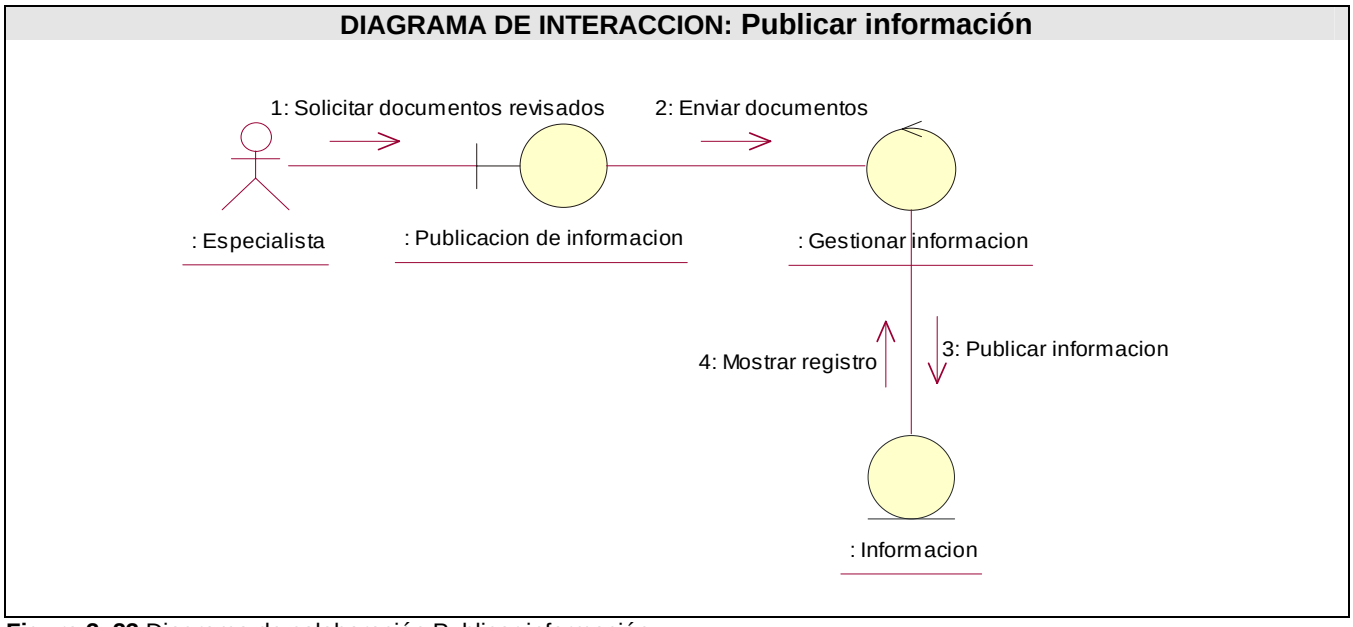

Figura 3. 23 Diagrama de colaboración Publicar información

3.1.3 Diseño

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema.

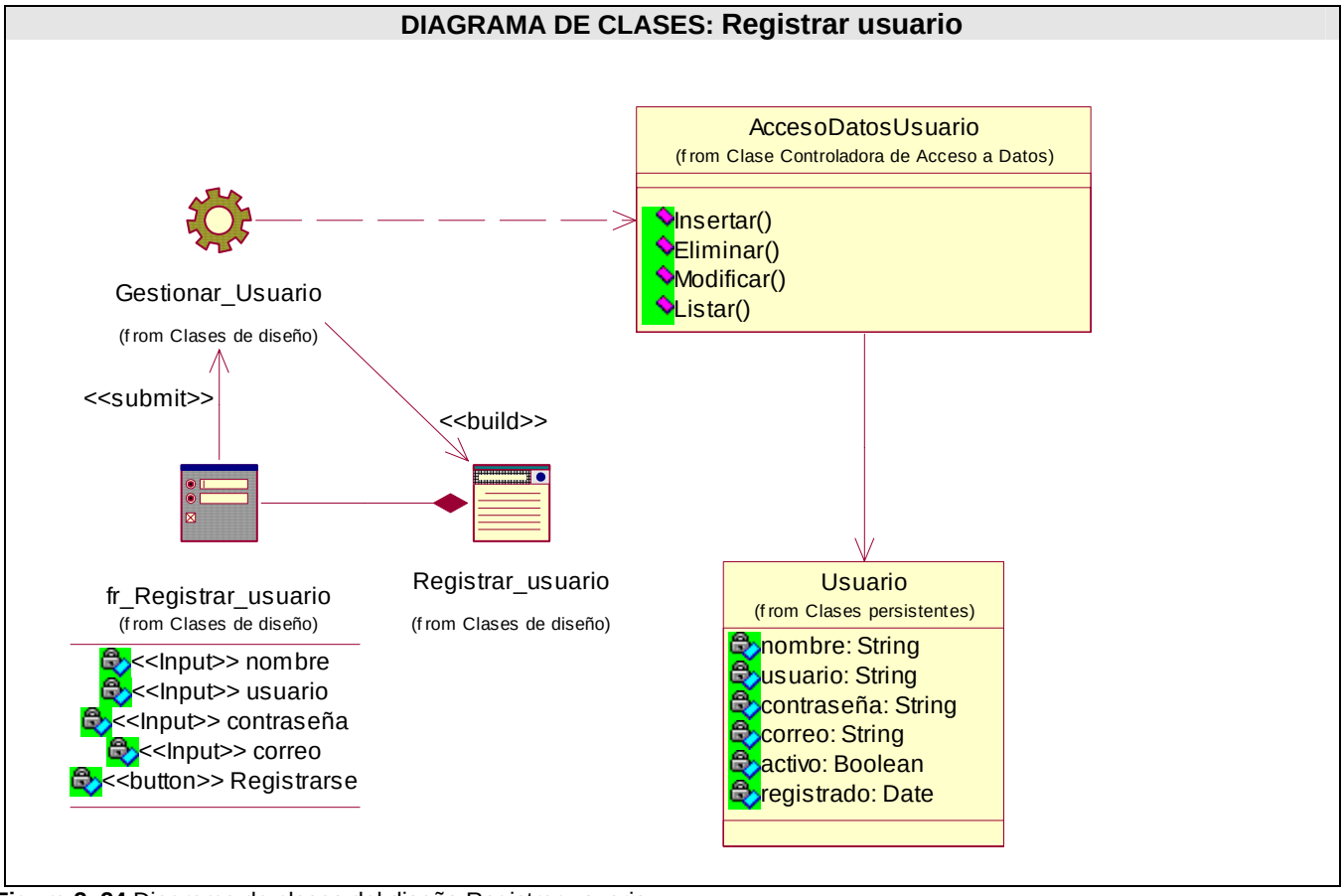

Figura 3. 24 Diagrama de clases del diseño Registrar usuario

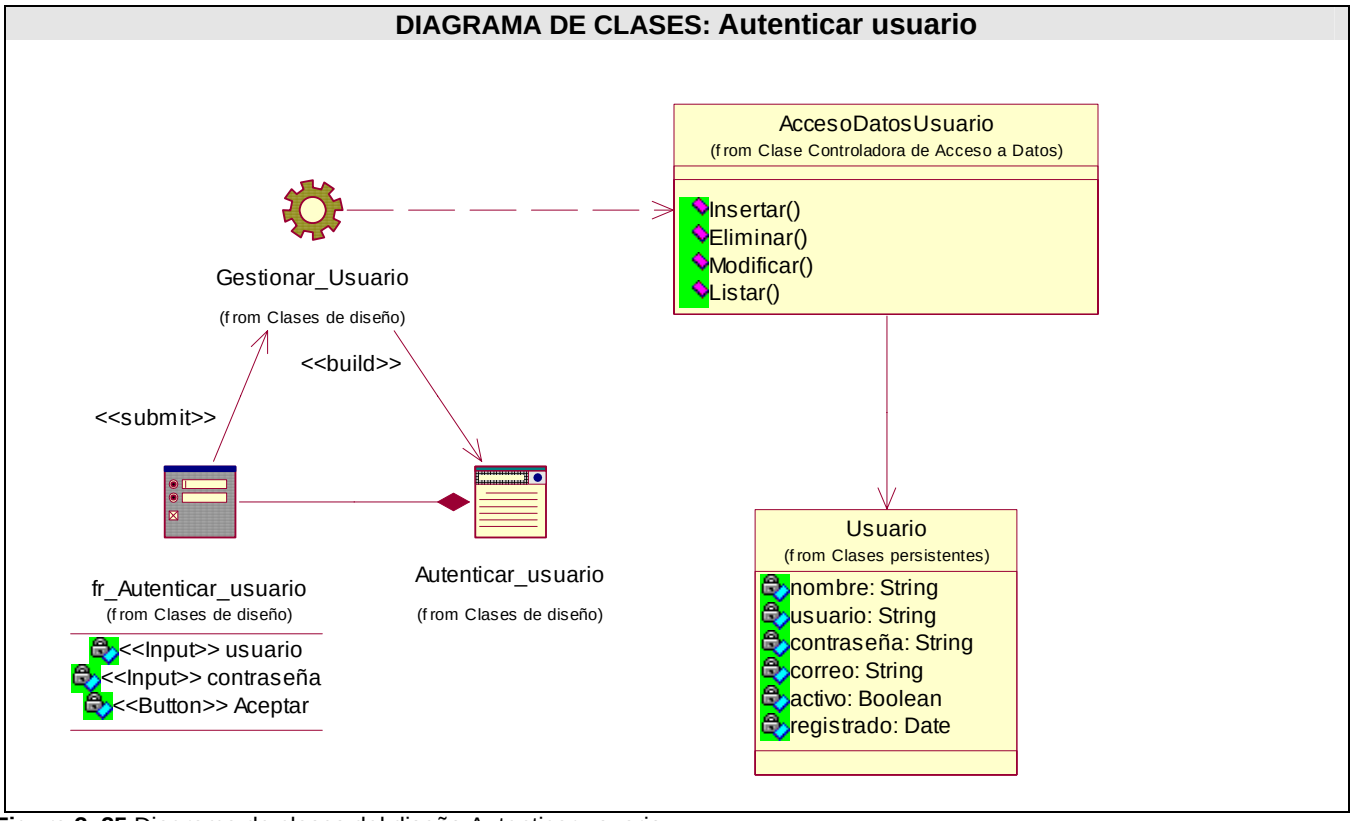

Figura 3. 25 Diagrama de clases del diseño Autenticar usuario

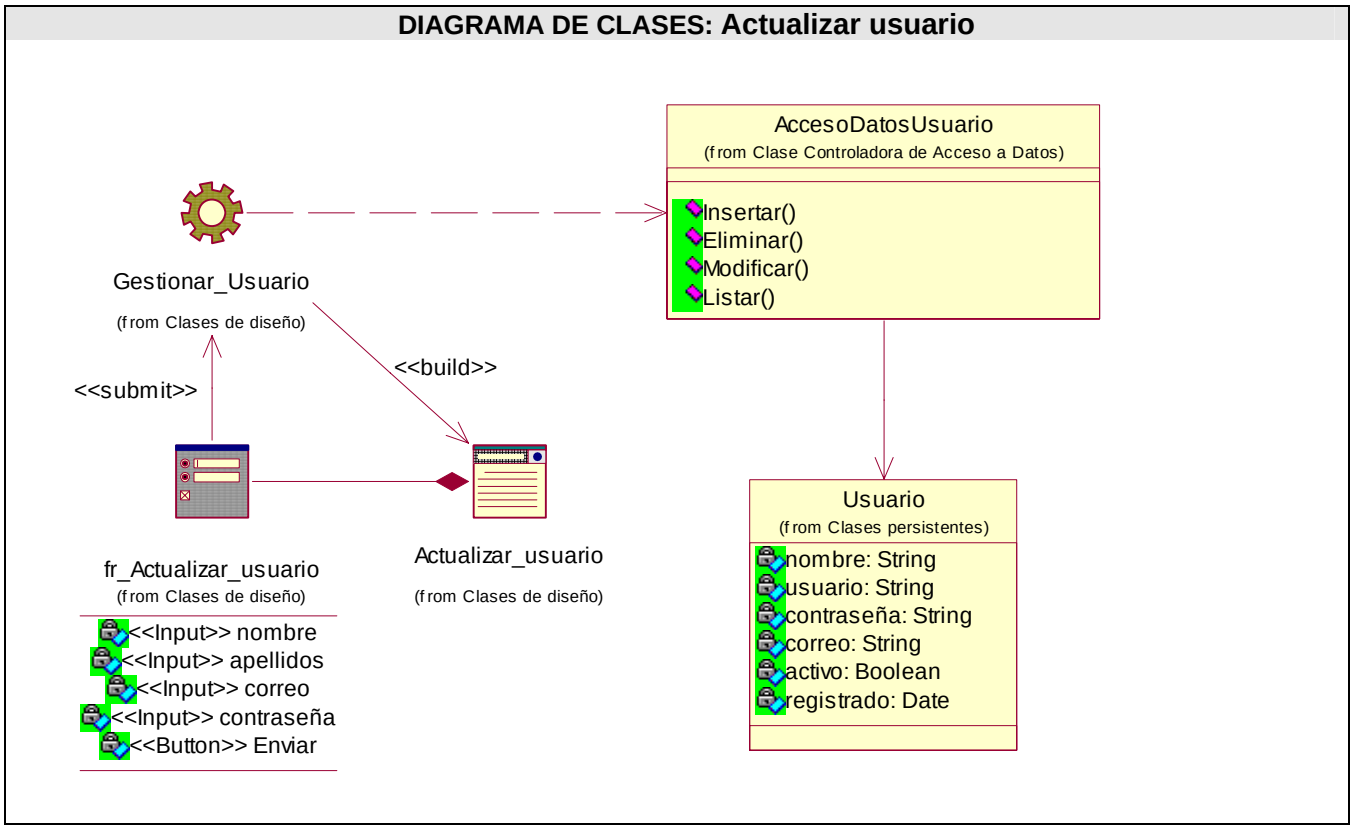

Figura 3. 26 Diagrama de clases del diseño Actualizar usuario

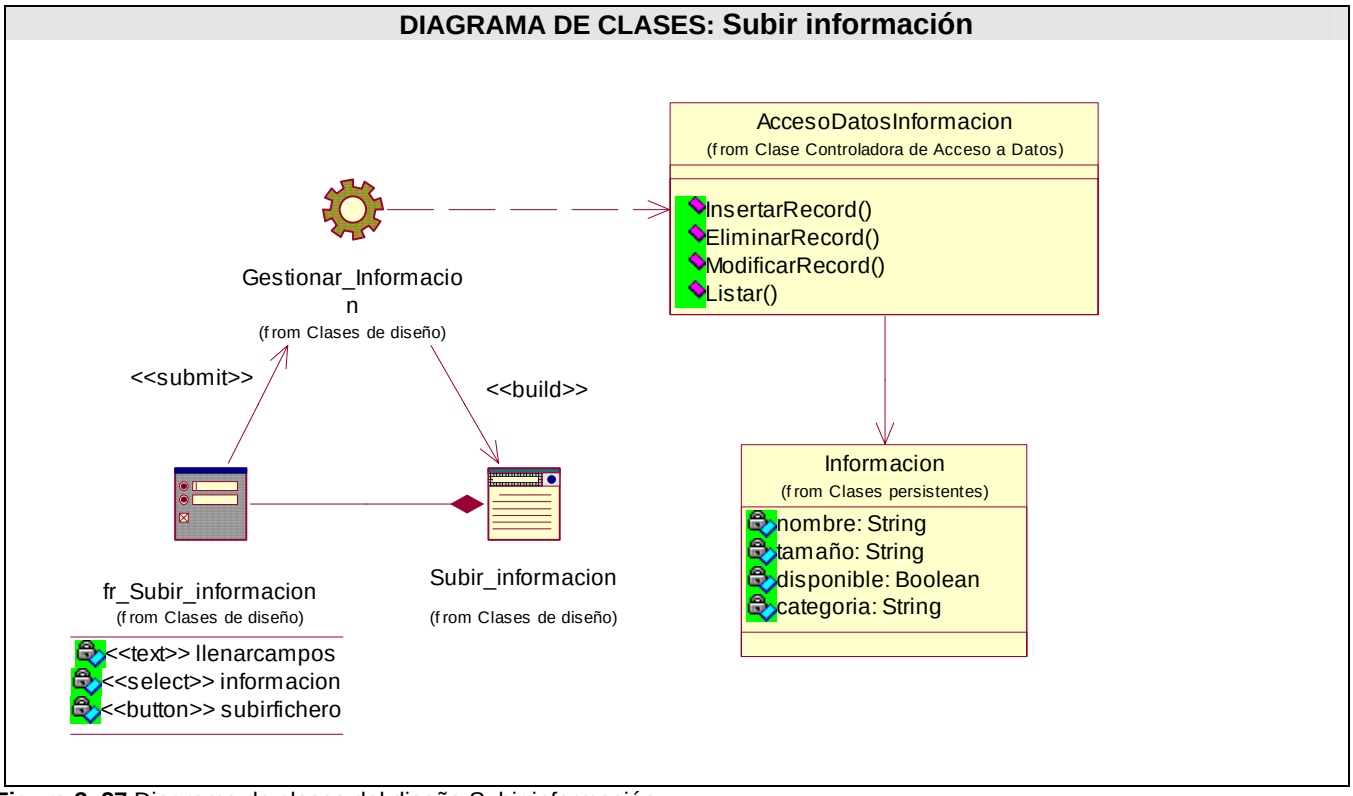

Figura 3. 27 Diagrama de clases del diseño Subir información

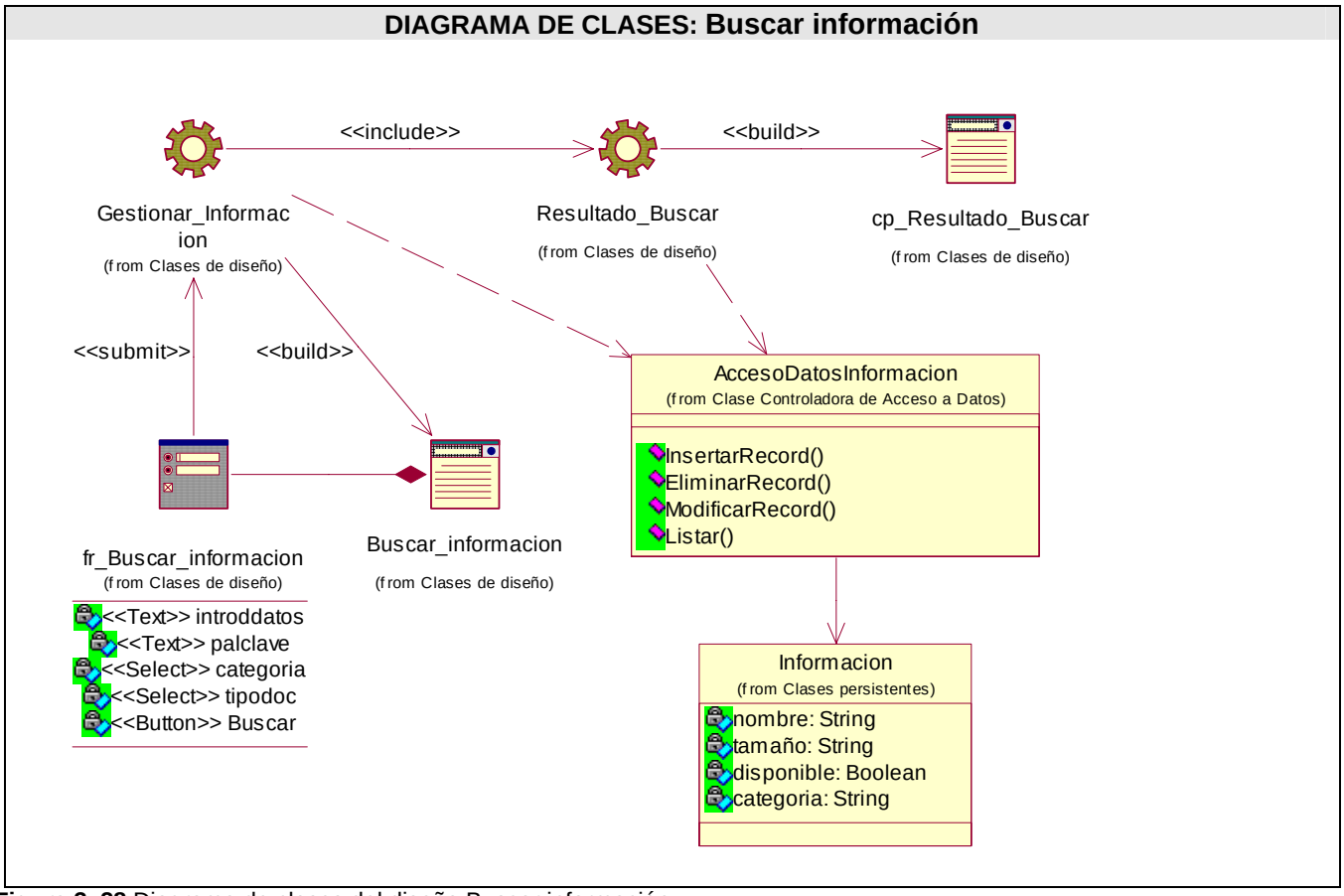

Figura 3. 28 Diagrama de clases del diseño Buscar información

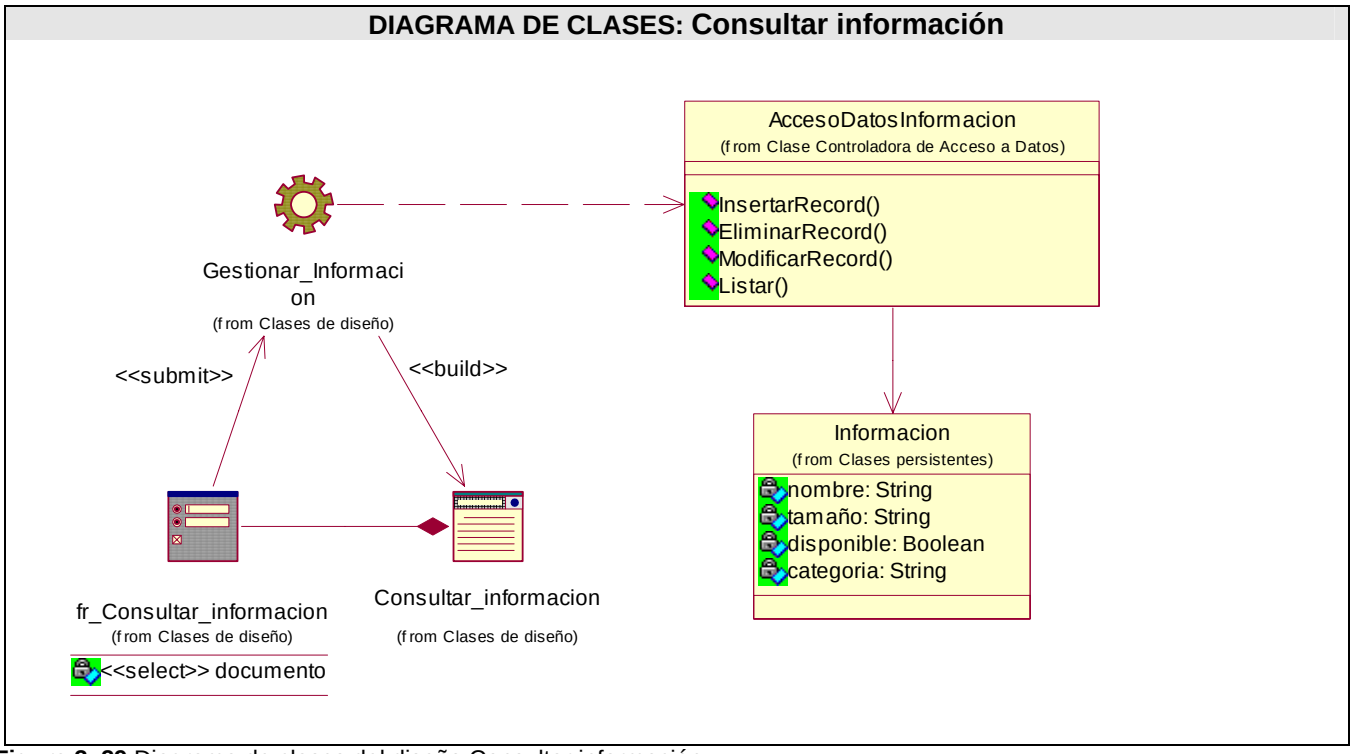

Figura 3. 29 Diagrama de clases del diseño Consultar información

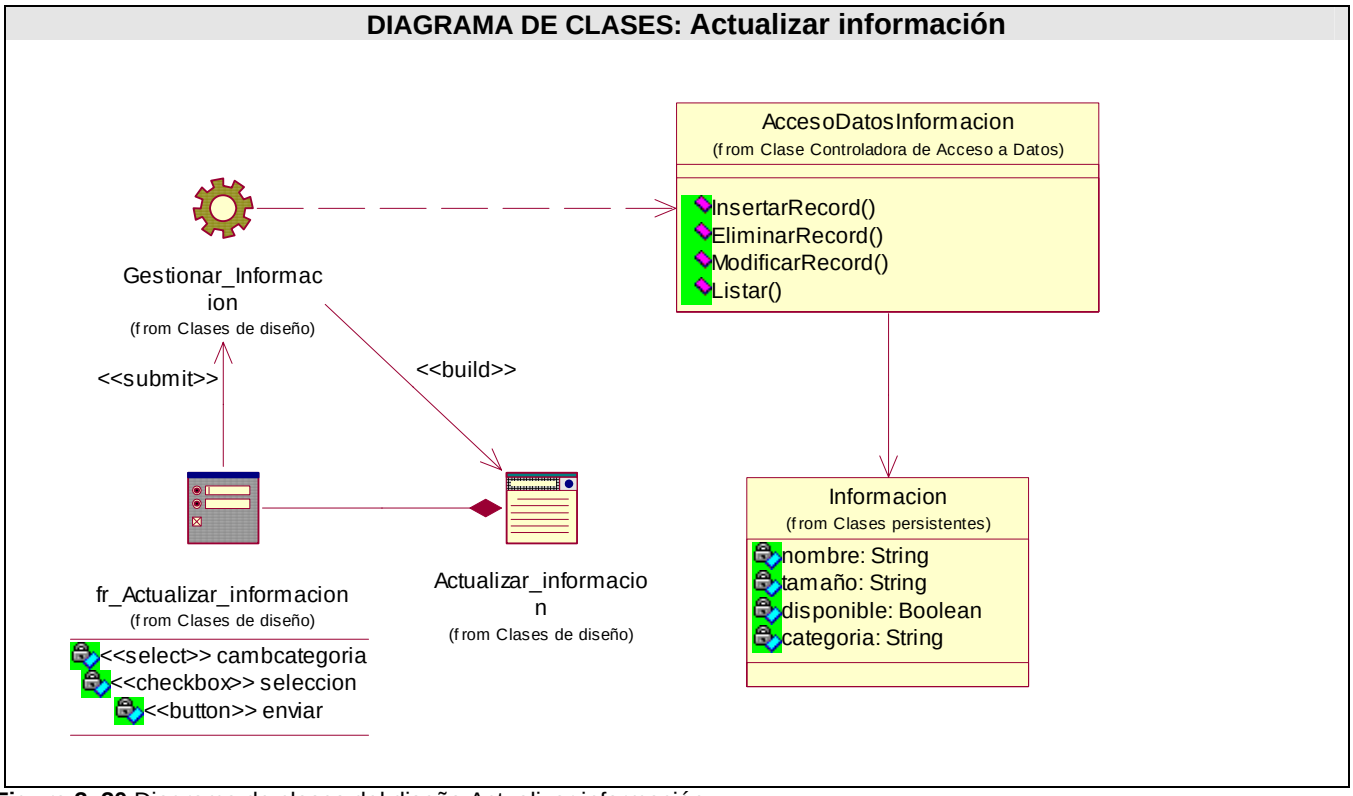

Figura 3. 30 Diagrama de clases del diseño Actualizar información

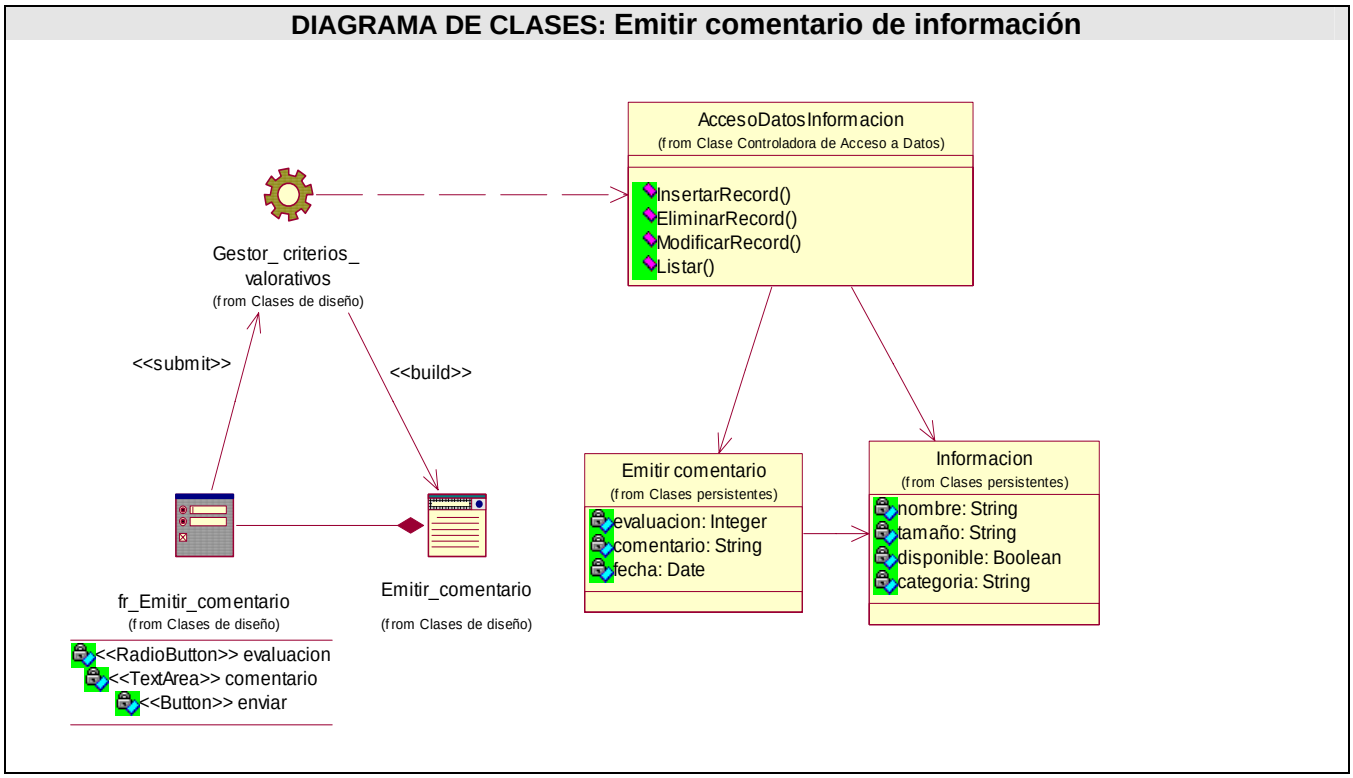

Figura 3. 31 Diagrama de clases del diseño Emitir comentario de información

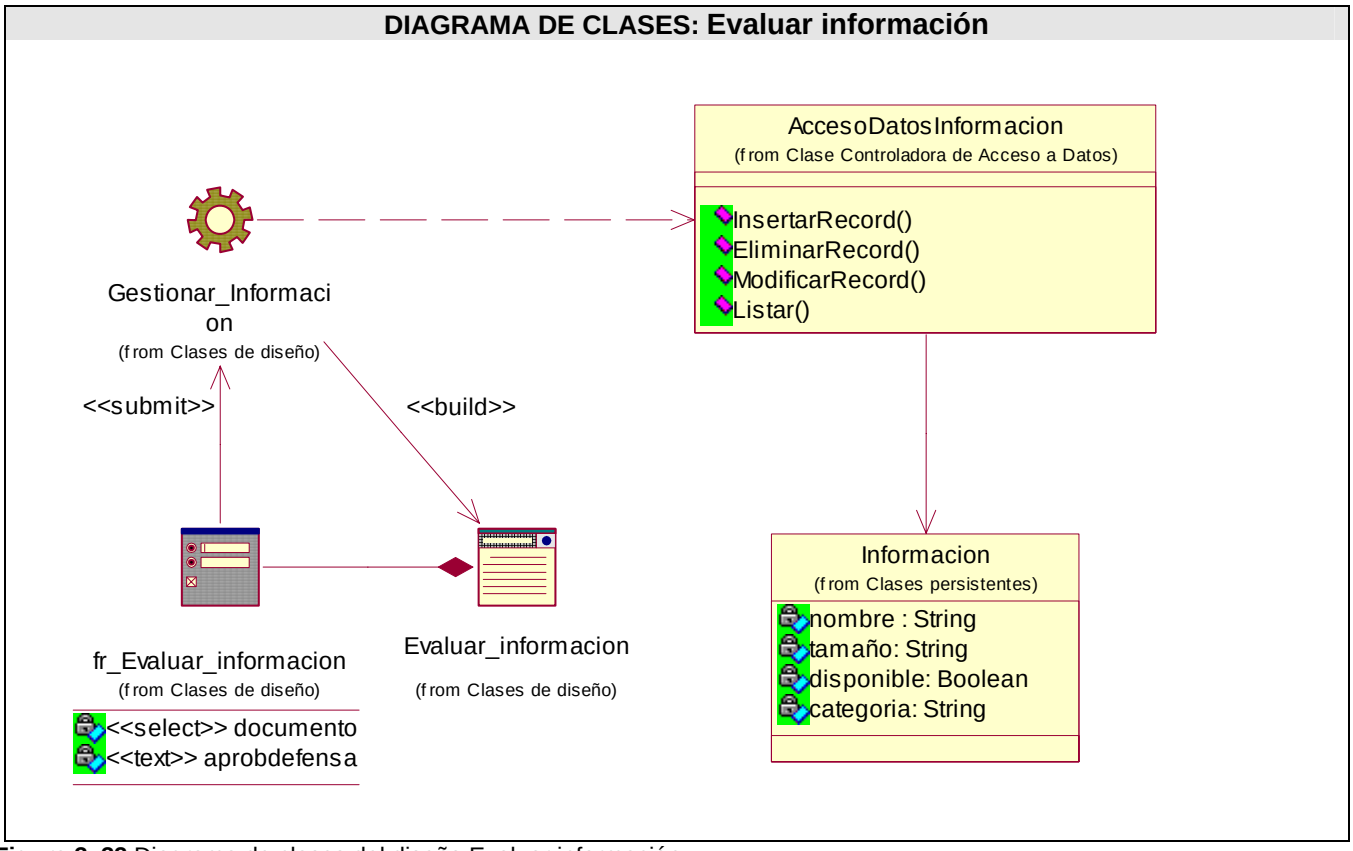

Figura 3. 32 Diagrama de clases del diseño Evaluar información

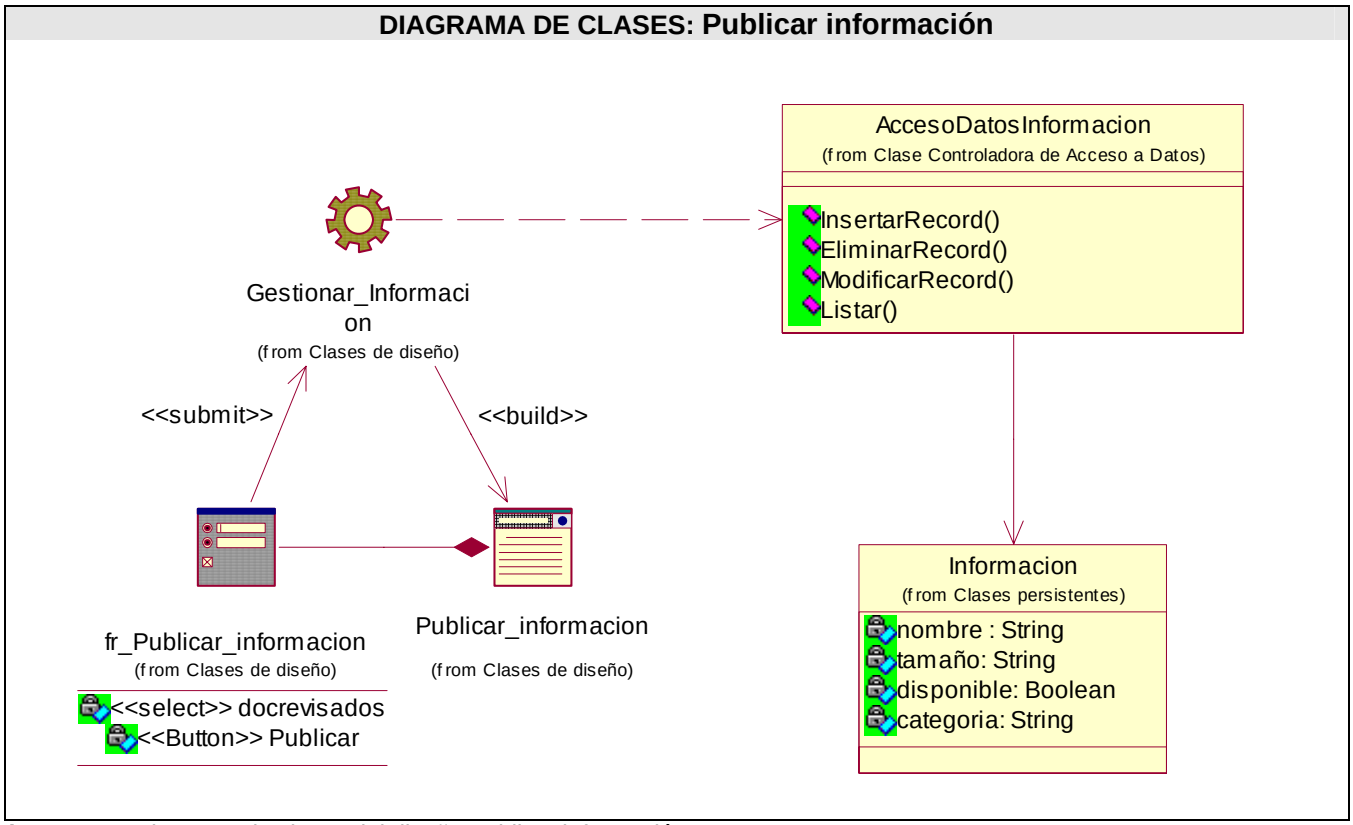

Figura 3. 33 Diagrama de clases del diseño Publicar información

Descripción de las clases

Nombre: Usuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre	String
nomb_usuario	String
contraseña	String
correo	String
Para cada responsabilidad:	
Nombre:	construct(nombre, nomb_usuario, contraseña, correo)
Descripción:	Esta función construye los atributos de la clase.
Nombre:	set(pvalor)
Descripción:	Asigna valor al atributo. Existe una función set para cada atributo de la clase.
Nombre:	get()
Descripción:	Recoge el valor al atributo. Existe una función get para cada atributo de la clase.

Tabla 1. Descripción de clases: Usuario

Nombre: Informacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre	String
tamaño	String
disponible	Boolean
categoria	String
Para cada responsabilidad:	
Nombre:	construct(nombre, tamaño, disponible, categoria)
Descripción:	Esta función construye los atributos de la clase.
Nombre:	set(pvalor)
Descripción:	Asigna valor al atributo. Existe una función set para cada atributo de la clase.
Nombre:	get()
Descripción:	Recoge el valor al atributo. Existe una función get para cada atributo de la clase.

Tabla 2. Descripción de clases: Información

Nombre: Emitir comentario	
Tipo de clase: Entidad	
Atributo	**Tipo**
evaluacion	Integer
comentario	String
fecha	Date
Para cada responsabilidad:	
Nombre:	construct(evaluacion, comentario, fecha)
Descripción:	Esta función construye los atributos de la clase.
Nombre:	set(pvalor)
Descripción:	Asigna valor al atributo. Existe una función set para cada atributo de la clase.
Nombre:	get()
Descripción:	Recoge el valor al atributo. Existe una función get para cada atributo de la clase.

Tabla 3. Descripción de clases: Emitir comentario

3.1.4 Diseño de la base de datos

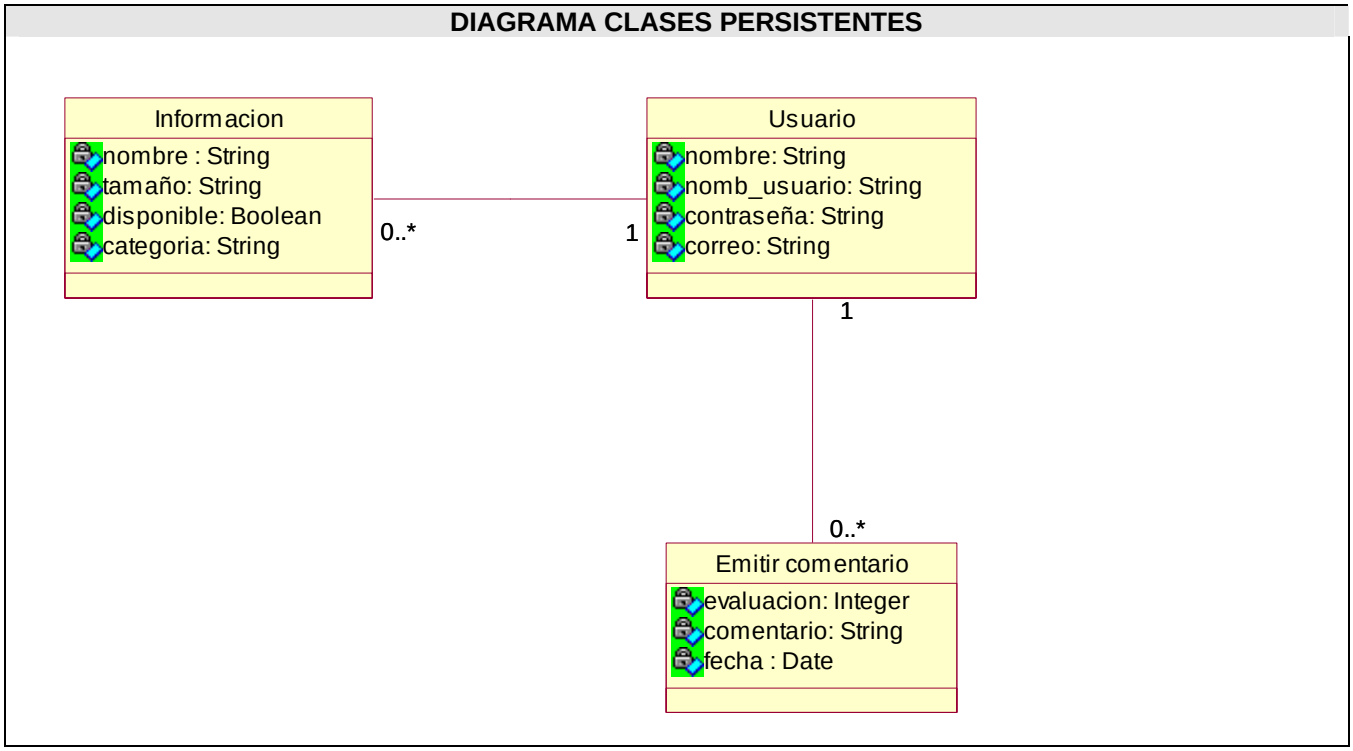

Figura 3. 34 Diagrama de clases persistentes

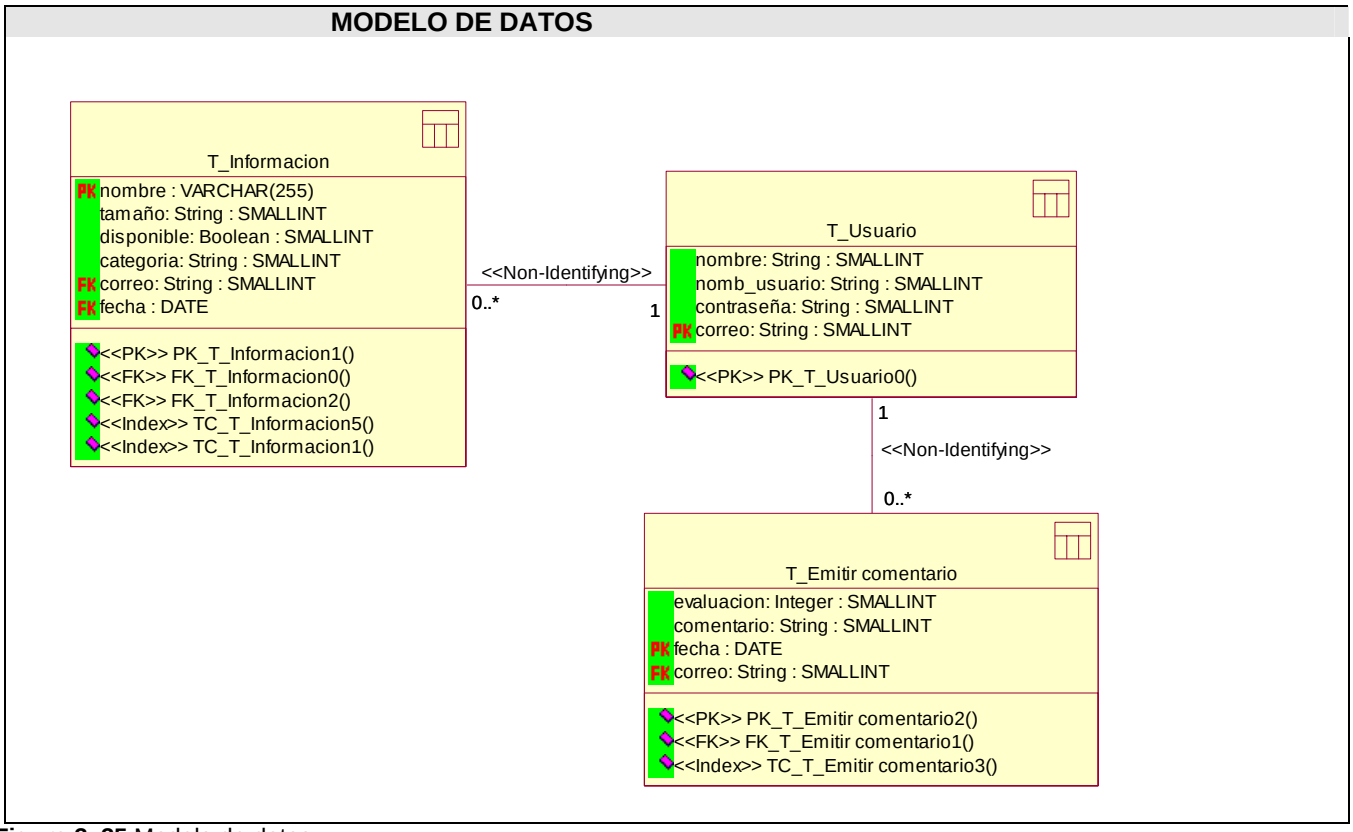

Figura 3. 35 Modelo de datos

3.2 Análisis de costo-beneficio

Esta aplicación cuando se llegue a implementar no requerirá una gran inversión en software pues casi todas las herramientas que se proponen son libres y además multiplataformas, lo que permite que la misma pueda ser usada en diferentes ambientes como por ejemplo Windows, Linux, etc. Además se beneficiarán los usuarios de la comunidad al contar con un sistema que les permita publicar así como consultar sin ninguna restricción los documentos del repositorio de información.

Conclusiones del capítulo

En este capítulo se mostró todo lo referente al análisis y diseño de la aplicación se vieron los modelos de clases del análisis, los diagramas de colaboración, las clases del diseño así como la descripción de las clases y el modelo de datos y por último se analizó la factibilidad del trabajo.

CONCLUSIONES

Con la realización de este trabajo se ha dado cumplimiento a los objetivos planteados en la introducción del mismo. Se realizó un análisis de los aspectos teóricos de los repositorios de información dándose a conocer que son los mismos, los factores que provocan su crecimiento y lo que le aporta a una institución tener un repositorio de información. Luego se vieron las características de algunos de estos repositorios de información, enfocándose más en las características de E-LIS y se llegó a la conclusión de que era necesario realizar el diseño de un sistema completamente nuevo dado a las características particulares de la Universidad de las Ciencias Informáticas debido a que los sistemas que se analizaron no son en español y ende los trabajos que se albergan en estos repositorios no son de habla hispana, excepto el de TDR que como su nombre lo indica es para tesis doctorales y también por un problema de seguridad se decide diseñar un sistema nuevo. Por último se realiza el análisis y diseño del repositorio de información para la biblioteca de la universidad.

RECOMENDACIONES

- Se sugiere continuar con la implementación del sistema ya que el beneficio que aportará a los usuarios de la comunidad de la universidad de las ciencias informáticas es de gran valor, al permitirle a los mismos publicar, consultar y descargar trabajos investigativos.
- Se sugiere implementar el protocolo OAI-PMH para que en un futuro este repositorio de información pueda ser accesible a través de Internet.
- Si en el ambiente en el que se va a trabajar es netamente software libre se recomienda utilizar para el modelado visual una herramienta de software libre también como ArgoUML, Umbrello, etc.

BIBLIOGRAFÍA

Reyero, José.Características de Drupal.http://softwarelibre.apif.info/caracteristicas_drupal

04/06/2006

Cuerda, Xavier; Minguillón, Julia.Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto.http://mosaic.uoc.edu/articulos/cms1204.html

29/11/2004

Sarduy, Yanetsys; Urra, Pedro.Sistemas de gestión de contenidos: En busca de una plataforma ideal.http://bvs.sld.cu/revistas/aci/vol14_4_06/aci11406.htm

04/10/2006

Plone: intranets con código abierto.http://eltituloesloultimo.wordpress.com/2005/12

30/12/2005

López, César.Ejemplo de desarrollo software utilizando la metodología RUP.http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/index.html

28/07/2003

Letelier, Patricio.Introducción a Rational Unified Process(RUP).http://www.dsic.upv.es/~letelier/pub/p16.ppt

02/04/2004

Sánchez, Carlos.Aplicaciones en capas.http://oness.sourceforge.net/proyecto/html/ch03s02.html

28/09/2004

Reynoso,Carlos;Kicillof,Nicolás.http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp

03/2004

Pecos, Daniel.PostGreSQL vs. MySQL.http://www.netpecos.org/docs/mysql_postgres/index.html

Aguilar, Vicente; Suau, Pablo.MySQL vs. PostgreSQL.http://www.fedora-es.com/node/189

Pérez, José M. ¿Qué es MySQL?.http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm
16/08/2005
Moreno, Gerardo.Ingenieria de Software UML.http://www.monografias.com/trabajos5/insof/insof.shtml
Copyright © 2001 INDUDATA LTDA.Rational Rose.http://www.indudata.com/1rational_rose.htm
26/05/2007
Del Castillo, Alvaro.El servidor de Web Apache: Introducción práctica.http://acs.barrapunto.org/svn/articulos/trunk/LinuxActual/Apache/html/index.html
Del Castillo, Alvaro.Manual PHP. Características de PHP.http://www.lawebera.es/manuales/php/2-2.php
Alvarez, Miguel A.Qué es Python.http://www.desarrolloweb.com/articulos/1325.php
Ibáñez, Juan D.; Marzal, Andrés.Un sistema de información académica universitaria.http://es.tldp.org/Presentaciones/200002hispalinux/conf-20/20-html/ponencia.html
Benito, Carlos.Depositar artículos en repositorios acelera su difusión.http://weblogs.madrimasd.org/openaccess/archive/2006/11/24/52260.aspx
24/11/2006
López, Pablo; Pérez, Guillermo.ZOPE: MUCHO MÁS QUE UN SERVIDOR WEB.http://jungla.dit.upm.es/~joaquin/las/trabajos/2001/ZOPE.pdf.gz
Arencibia, Ricardo.Las iniciativas para el acceso abierto a la información científica en el contexto de la Web semántica.http://www.bibliosperu.com/articulos/26/26_06.pdf
12/2006
Klibanski, Monica.ELIS y el paradigma del acceso abierto a la información.http://www.a-abierto.blogspot.com
24/05/2007

Rodríguez, Luis.Seminario Archivos de eprints: nuevo campo de trabajo para bibliotecas universitarias y científica.http://www.sedic.es/p_boletinclip42_sedicabierto2.htm
13/12/2004

Romero, Santiago.ProQuest y la Biblioteca Virtual en Ciencias de la Salud.http://www.usal.es/~sabus/site%20med/descargas/proquest_santiago.pdf
09/2006

GLOSARIO

API: Una API es una Interfaz de Programación de Aplicaciones (Application Programming Interface), un conjunto de funciones o métodos usados para acceder a ciertas funcionalidades.

Blogs: Un blog, también conocido como weblog o cuaderno de bitácora, es un sitio Web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente.

Interoperabilidad: es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.

Metadatos: son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos, llamado recurso.

OAI: La Iniciativa de Archivos Abiertos (OAI) es un movimiento con diferentes corrientes, cuyo mayor logro ha sido establecer pautas que garanticen la interoperabilidad entre diferentes repositorios y sistemas de recuperación.

OAI-PMH: Protocolo para la transmisión de contenidos en Internet, solamente es una interfaz sumamente sencilla para acceder a la información bibliográfica disponible en un archivo o repositorio.

Open Source: Calificación de software que cumple una serie de requisitos, principalmente aquel que permite una libre redistribución, distribuye el código fuente, y permite modificaciones y trabajos derivados.

Web semántica: es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida.

ANEXOS

ANEXO I

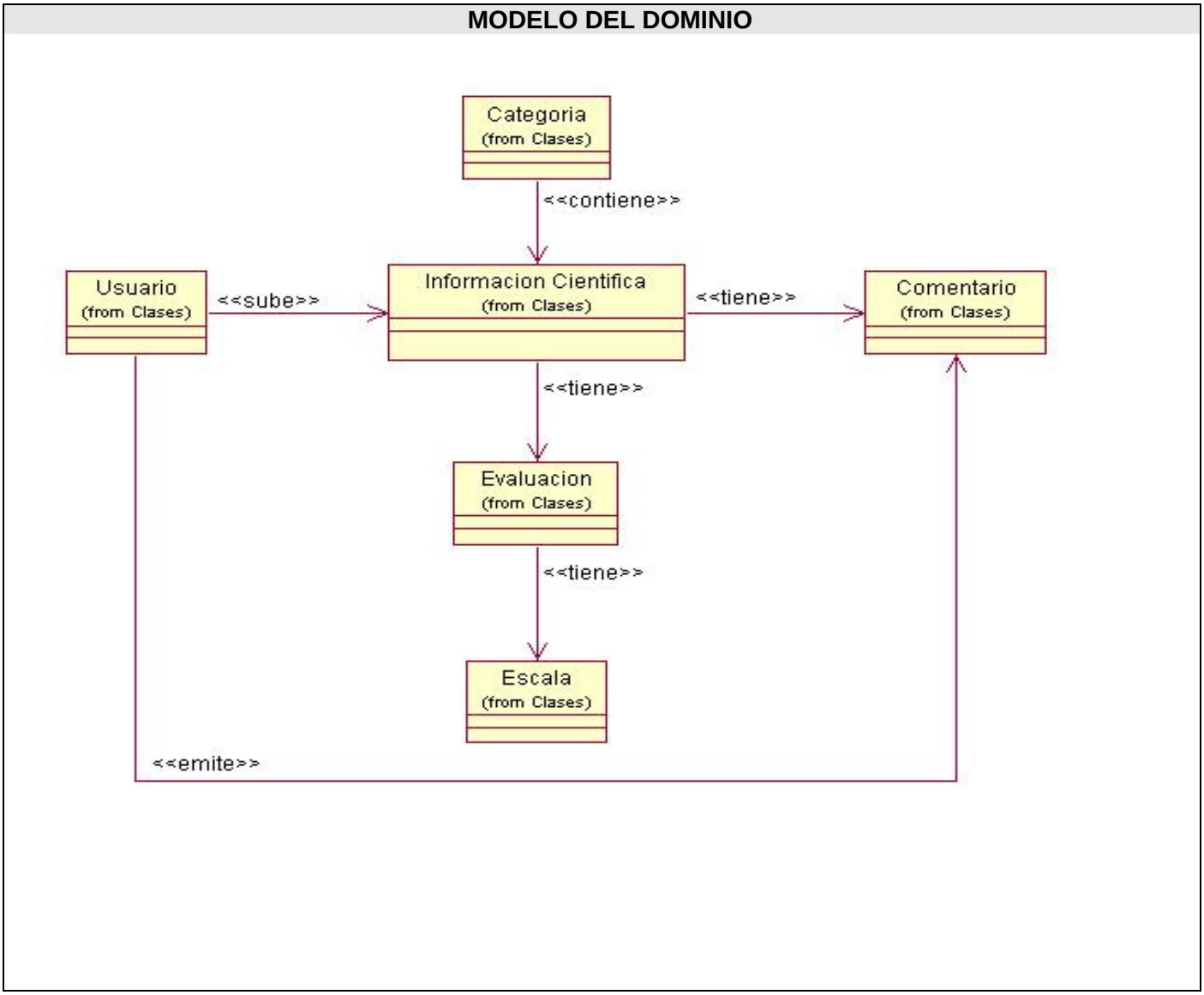

Figura 2. 1 Modelo del dominio

ANEXO II

Actores	Justificación
User	Representa una persona que aun no forma parte del sistema y esta a punto de registrarse o se ha registrado pero no se ha autenticado en el sistema.
UsuarioComunidad	Representa una persona cuyos datos fueron registrados en el sistema, por tanto puede aportar nuevos recursos así como descargar y ofrecer criterios valorativos.
Especialista	Representa una persona que actualiza los usuarios registrados y además es la encargada de publicar la información que aporta el usuario una vez esta haya sido evaluada por esta misma persona.

Tabla 4. Descripción de los actores

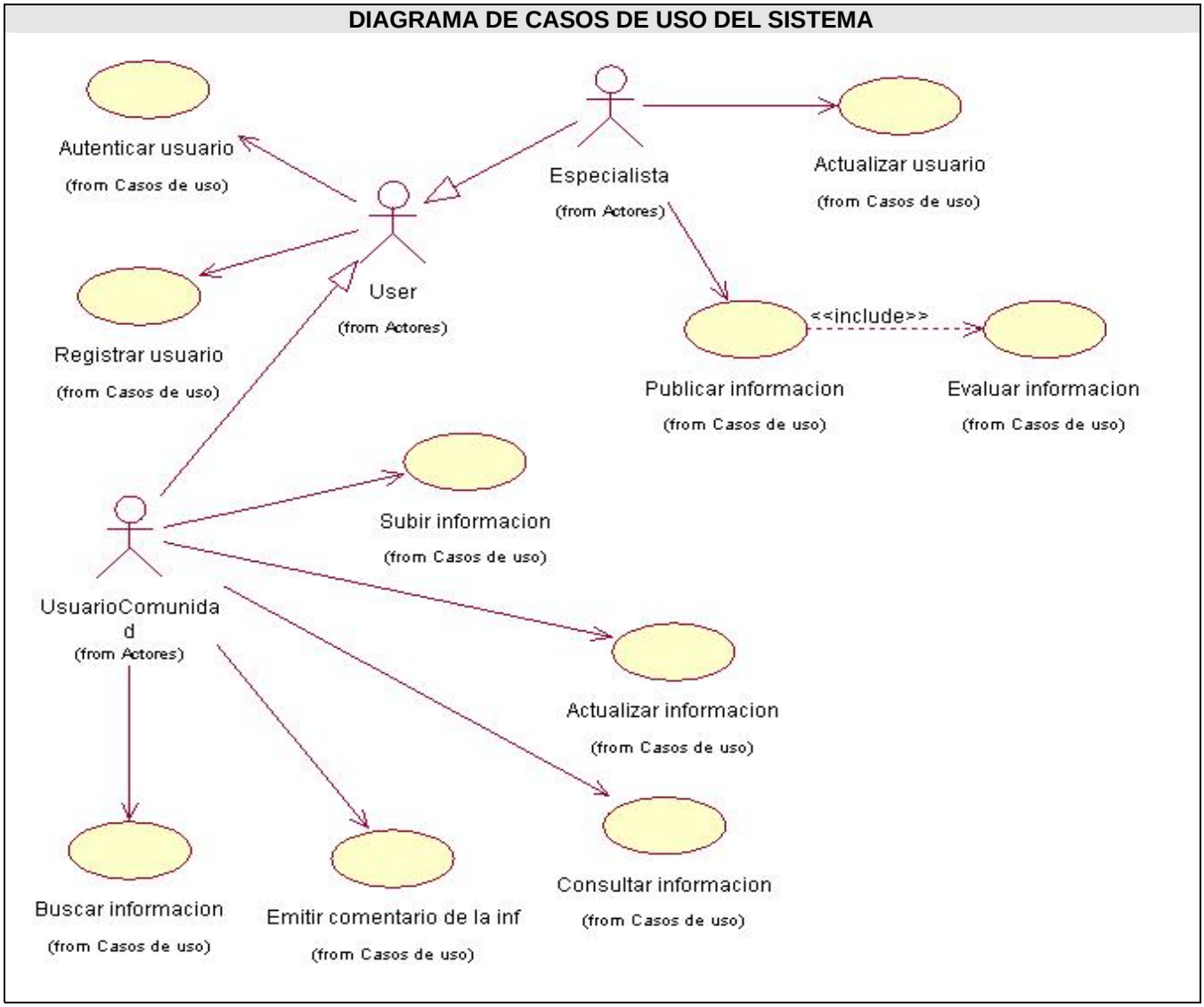

Figura 2. 2 Diagrama de casos de uso del sistema

Nombre del Caso de Uso	**Registrar usuario**
Actores	User (inicia)
Propósito	Permitir registrarse.
Resumen	El caso de uso inicia cuando el usuario decide crear una cuenta de acceso al sitio. El caso de uso finaliza cuando el sistema ha creado la cuenta de usuario.

Referencias	R1
Precondiciones	No debe existir un usuario con el alias que el user proporcionó
Poscondiciones	La cuenta de usuario ha sido creada.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario introduce los datos que le son pedidos.	1.1 El sistema comprueba la validez de los datos del usuario. 1.2 Verifica que el usuario no este registrado. 1.3 En caso de ser correcto crea cuenta de usuario.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.3 En caso de que los datos sean incorrectos se le envía un mensaje de aviso.
Prioridad:	

Tabla 5. Descripción textual del CU Registrar usuario

Nombre del Caso de Uso	**Autenticar usuario**
Actores	User (inicia)
Propósito	Permitir autenticarse.
Resumen	El Caso de Uso se inicia cuando el usuario introduce los datos que se le piden para acceder a la aplicación, estos se verifican y finaliza dándole los permisos y habilitándole la entrada.
Referencias	R2
Precondiciones	El usuario debe estar registrado.
Poscondiciones	Se habilitan las funcionalidades según lo privilegios.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario entra Usuario y Contraseña	1.1 El sistema encripta la contraseña. 1.2 Busca el usuario y compara la contraseña. 1.3 En caso de ser correcto se le asignan los permisos.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.3 En caso de no existir se envía un mensaje de aviso.
Prioridad:	Crítico

Tabla 6. Descripción textual del CU Autenticar usuario

Nombre del Caso de Uso	Actualizar usuario
Actores	Especialista (inicia)
Propósito	Permitir al usuario actualizar sus datos.
Resumen	El caso de uso inicia cuando el usuario decide actualizar sus datos personales. El sistema edita los datos del usuario, el cual hace los cambios que desee. El caso de uso finaliza cuando se han actualizado los datos.
Referencias	R3
Precondiciones	El usuario debe estar autenticado.
Poscondiciones	Los datos del usuario quedan actualizados.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario solicita actualizar sus datos personales.	1.1 El sistema comprueba que los datos no coinciden con los de otro usuario. 1.2 Comprueba validez de los datos. 1.3 En caso de ser correcto actualiza usuario.

Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.3 En caso de coincidencia o incorrección en los datos se envía un mensaje de aviso.
Prioridad:	Crítico

Tabla 7. Descripción textual del CU Actualizar usuario

Nombre del Caso de Uso	Subir información
Actores	UsuarioComunidad (inicia)
Propósito	Permitir que el usuario pueda subir al repositorio su trabajo.
Resumen	El caso de uso inicia cuando el usuario decide subir una información. El sistema solicita el archivo, lo sube al servidor y chequea que su formato sea valido. El caso de uso finaliza cuando la información queda almacenada.
Referencias	R4
Precondiciones	El usuario debe estar autenticado.

Poscondiciones	La información del usuario queda almacenada y se actualizan sus datos.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario introduce los datos del documento a depositar.	1.1 El sistema comprueba que los campos marcados como obligatorios sean llenados. 1.2 En caso de ser correcto permite subir el fichero.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.2 En caso de no cumplir con lo establecido se envía un mensaje de aviso.
Prioridad:	

Tabla 8. Descripción textual del CU Subir información

Nombre del Caso de Uso	**Buscar información**
Actores	UsuarioComunidad (inicia)
Propósito	Permitir que el usuario haga una búsqueda de la información que necesita.
Resumen	El caso de uso inicia cuando el usuario decide buscar alguna información. El sistema muestra una serie de criterios que el usuario puede utilizar para especificar las características del contenido que busca.
Referencias	R5
Precondiciones	El usuario debe estar autenticado.
Poscondiciones	---
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario selecciona como quiere realizar la búsqueda. 2. Introduce los datos de lo que desea encontrar.	2.1 El sistema busca la información solicitada por el usuario. 2.2 Muestra la información según lo solicitado por el usuario.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	2.2 En caso de no existir lo buscado muestra un mensaje de aviso.
Prioridad:	

Tabla 9. Descripción textual del CU Buscar información

Nombre del Caso de Uso	Consultar información
Actores	UsuarioComunidad (inicia)
Propósito	Permitir al usuario consultar la información que previamente ha sido buscada.
Resumen	El caso de uso inicia cuando el usuario decide consultar una información que ha sido buscada con anterioridad.
Referencias	R6
Precondiciones	El usuario debe estar autenticado.
Poscondiciones	---
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario selecciona la información que quiere consultar. 2. El usuario consulta la información deseada.	1.1 El sistema abre el documento y muestra el contenido.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
Prioridad:	

Tabla 10. Descripción textual del CU Consultar información

Nombre del Caso de Uso	Actualizar información
Actores	UsuarioComunidad (inicia)
Propósito	Permitir al usuario actualizar la información que ha sido aportada por el
Resumen	El caso de uso inicia cuando el usuario decide actualizar una información. El sistema muestra solo aquellas que han sido aportadas por el usuario, quien selecciona una y la actualiza.
Referencias	R7
Precondiciones	El usuario debe estar autenticado y debe haber aportado alguna información.
Poscondiciones	La información queda actualizada.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario solicita actualizar información.	1.1 El sistema muestra los documentos aportados por el usuario. 1.2 Solicita al usuario que escoja el que desea actualizar.
2. El usuario selecciona el documento y modifica los datos pertinentes.	2.1 El sistema actualiza el documento.

Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.1 En caso de que el usuario no halla aportado ningún documento se muestra un mensaje de aviso.
Prioridad:	

Tabla 11. Descripción textual del CU Actualizar información

Nombre del Caso de Uso	**Emitir comentario de información**
Actores	UsuarioComunidad (inicia)
Propósito	Permitir que el usuario pueda dar una evaluación y/o opinión de la información.
Resumen	El caso de uso inicia cuando el usuario decide otorgar una evaluación y/o emitir su opinión acerca de la información.
Referencias	R8
Precondiciones	El usuario debe estar autenticado.
Poscondiciones	Se actualizan la evaluación y/o la opinión de la información.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario solicita emitir criterio sobre un documento.	1.1 El sistema muestra área de discusión.
2. El usuario da su criterio acerca de un determinado documento.	2.1 El sistema recoge todos los comentarios de los usuarios acerca de un determinado documento.

Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
Prioridad:	

Tabla 12. Descripción textual del CU Emitir comentario de información

Nombre del Caso de Uso	**Evaluar información**
Actores	Especialista (inicia)
Propósito	Permitir al especialista evaluar la información que ha sido subida al repositorio por el UsuarioComunidad.
Resumen	El caso de uso inicia cuando el especialista decide evaluar la información que quiere aportar el UsuarioComunidad al repositorio, este la evalúa y si cumple con los requisitos entonces la publica.
Referencias	R9

Precondiciones	Debe haber algún documento depositado.
Poscondiciones	Se evalúa la información.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario deposita el documento. 2. El usuario espera que el documento depositado sea aceptado.	1.1 El sistema revisa el documento depositado por el usuario.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.1 En caso de que el documento no cumpla con lo requerido se muestra un mensaje de aviso.
Prioridad:	

Tabla 13. Descripción textual del CU Evaluar información

Nombre del Caso de Uso	**Publicar información**
Actores	Especialista (inicia)
Propósito	Permitir al especialista publicar en el repositorio la información que aporta el UsuarioComunidad.
Resumen	El caso de uso inicia cuando el especialista ha evaluado la información, entonces posteriormente puede publicar la misma.
Referencias	R10
Precondiciones	El documento debe cumplir con todo lo requerido.
Poscondiciones	Se publica la información.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario tiene documentos pendientes de aceptación.	1.1 El sistema acepta el documento depositado por el usuario. 1.2 E l sistema publica el documento que ha sido aportado por el usuario.
Flujo alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.1 En caso de no ser aceptado el documento se muestra un mensaje de aviso.
Prioridad:	

Tabla 14. Descripción textual del CU Publicar información

ANEXO III

Referencias bibliográficas

[1] http://eprints.rclis.org/archive/00004475/01/redc.pdf
Consultado 2/02/2007

[2] Definición de servidor.http://www.masadelante.com/faq-servidor.htm
@2007 masadelante.com

[3] Vega, Jesús. El servidor Web.http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html
21/03/2002

[4] Definición de lenguaje de programación.
http://www.alegsa.com.ar/Dic/lenguaje%20de%20programacion.php
Consultado 18/02/2007

[5] Sistema de gestión de base de datos.
http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos
Consultado 18/02/2007

[6] Sistema de gestión de contenidos.http://www.atrioweb.com/cms/
Consultado 18/02/2007

[7] Sistema de gestión de contenidos.
http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_contenido
10/05/2007.

[8] Ivar Jacobson, Grady Booch, James Rumbaugh. El proceso unificado de desarrollo de software

[9] Herramienta CASE.http://es.wikipedia.org/wiki/CASE
8/05/2007